Strategies for Selective and Adaptive Resilience in Reconfigurable Space Systems and Apps

by

Sebastian Sabogal

M.S. Electrical and Computer Engineering, University of Florida, 2017

Submitted to the Graduate Faculty of

the Swanson School of Engineering in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2021

UNIVERSITY OF PITTSBURGH

SWANSON SCHOOL OF ENGINEERING

This dissertation was presented

by

Sebastian Sabogal

It was defended on

April 2, 2021

and approved by

Alan D. George, Ph.D., Professor
Department of Electrical and Computer Engineering

Zhi-Hong Mao, Ph.D., Professor
Department of Electrical and Computer Engineering

Jingtong Hu, Ph.D., Assistant Professor
Department of Electrical and Computer Engineering

Samuel J. Dickerson, Ph.D., Assistant Professor
Department of Electrical and Computer Engineering

Michael S. Ramsey, Ph.D., Professor
Department of Geology and Environmental Science

Dissertation Director: Alan D. George, Ph.D., Professor
Department of Electrical and Computer Engineering

Strategies for Selective and Adaptive Resilience in Reconfigurable Space Systems and Apps

Sebastian Sabogal, PhD

University of Pittsburgh, 2021

Due to ongoing advancements in sensor technology and innovations in spacecraft autonomy enabled by compute-intensive deep-learning (DL) methods, modern spacecraft increasingly require more onboard processing capabilities that address the computational demands required for future space missions. Spacecraft designers are challenged to create dependable, high-performance space computers capable of converting onboard an immense volume of raw sensor data into actionable information that can be used to formulate critical decisions autonomously. Furthermore, this space-computing challenge is further exacerbated with stringent constraints in size, weight, power, and cost (SWaP-C) and dependability requirements due to radiation effects in the harsh environment.

The proliferation of small satellites (SmallSats) has enabled a paradigm for low-SWaP-C missions that frequently employ commercial-off-the-shelf (COTS) devices, including FPGAs and hybrid system-on-chips (SoCs), to improve onboard processing capabilities. These commercial devices have numerous architectural advantages that provide superior performance, energy efficiency, and affordability compared to radiation-hardened (rad-hard) alternatives but are highly susceptible to radiation-induced single-event effects (SEEs) that can impact mission dependability. To improve dependability, hardware-redundancy techniques are frequently employed for SEE mitigation; however, these methods incur a significant overhead that can be impractical for resource-constrained systems and can limit system performance. To create space computers capable of onboard DL, it is essential to create efficient methods for SEE mitigation and dependability evaluation.

In this dissertation research, we propose both selective and adaptive strategies for efficient SEE mitigation in reconfigurable space systems and applications. We devise, evaluate, and demonstrate these approaches in reconfigurable architectures to maximize performability subject to mission availability constraints. The first is HARFT, an environmentally adaptive,

gracefully degradable system architecture that maximizes system performability in reconfigurable systems. The second is RECON, a selectively and adaptively resilient semantic-segmentation accelerator that maximizes inference performability in reconfigurable systems. Finally, we propose a methodology for evaluating the performance and dependability characteristics of FPGA-accelerated DL models, which includes a hierarchical fault-injection approach to accelerate the dependability evaluation. This methodology allows spacecraft designers to create a design tradespace and select the optimal DL solution. This dissertation research demonstrates the efficacy of these methods to enable dependable, high-performance onboard processing for next-generation missions.

Table of Contents

List of Tables

List of Figures

Preface

I dedicate this dissertation to my family and friends, especially Daniel Sabogal, whose constant support has allowed me to persevere.

This dissertation research was supported by industry and government members of the National Science Foundation (NSF) Center for Space, High-Performance, and Resilient Computing (SHREC), formerly known as the Center for High-Performance Reconfigurable Computing (CHREC), and its IUCRC Program under Grant Nos. CNS-1738783 and IIP-1161022. I wish to thank Alan George for serving as coauthor and advisor for all dissertation research.

I wish to thank the students, faculty, members, sponsors, and vendors that supported SHREC to develop novel space computers (CSP, µCSP, and SSP) and missions (STP-H5-CSP, STP-H6-SSIVP, and STP-H7-CASPR). The opportunity to develop and contribute to these platforms as part of a team has been a unique and invaluable experience. From the STP-H6-SSIVP mission development team, I especially wish to thank Christopher Wilson, Nicholas Franconi, Brad Shea, Eric Shea, Ansel Barchowsky, Thomas Cook, Patrick Gauvin, Daniel Sabogal, Evan Gretok, Antony Gillette, Kevin Glunt, Theodore Schwarz, et al. From the STP-H7-CASPR team, I especially wish to thank Noah Perryman, Thomas Cook, Justin Goodwill, Theodore Schwarz, Evan Gretok, Antony Gillette, Tyler Garrett, Seth Roffe, et al. Finally, I wish to thank the Space Test Program (STP) Houston team, especially Robert Plunkett, Paige McClung, and Thomas Zerbe, for their support on mission integration and operations.

I wish to thank the Embedded Processing Group of the NASA Goddard Space Flight Center (GSFC) Science Data Processing (Code 587) Branch for their collaboration with this dissertation research, for their strong support for SHREC's space computers and missions, and for my opportunity to develop and contribute to the next generation of the NASA SpaceCube. I especially wish to thank Christopher Wilson, Gary Crum, Alessandro Geist, Nicholas Franconi, and Thomas Flatley for their mentorship and guidance for this dissertation research and my internship experiences at NASA GSFC.

Finally, this research was performed, in part, at the Los Alamos Neutron Science Center (LANSCE), an NNSA User Facility operated for the U.S. Department of Energy by Los Alamos National Laboratory (Contract 89233218CNA000001). I especially wish to thank Stephen Wender and Kranti Gunthoti for their support and guidance on using the 4FP30R/ICE-II instrument at LANSCE.

1.0 Introduction

Due to ongoing innovations in both sensor technology and spacecraft autonomy, spacecraft designers are challenged to create dependable, high-performance space computers that address the computational demands required for future space missions [58]. Modern spacecraft increasingly require high-performance computers to compress vast volumes of raw sensor data into actionable information to overcome bandwidth limitations in downlink. Spacecraft also increasingly require real-time capabilities to formulate and execute critical spacecraft maneuvers and operations autonomously. This space-computing challenge is further exacerbated with stringent constraints in size, weight, power, and cost (SWaP-C) and dependability requirements for harsh environments (e.g., radiation, thermal, vibration, and vacuum) often considered in space missions.

Simultaneously, machine learning (ML), particularly deep learning (DL), continues to proliferate in space applications to enhance mission capabilities in onboard data analysis and spacecraft autonomy [18, 65]. DL can enable a variety of complex mission tasks for both science and defense missions such as remote sensing [24], constellation management [90], and terrain-relative navigation [36]. However, despite these advantages, DL models are computationally intensive and often impractical for deployment on traditional radiation-hardened (rad-hard) space processors.

To address these challenges, space missions continue to adopt small satellites (SmallSats), including CubeSats, as low-SWaP-C platforms enabled by the miniaturization of electronics, sensors, and instruments [63]. Furthermore, to improve onboard processing capabilities, SmallSat missions frequently employ systems developed with solely commercial-off-the-shelf (COTS) technology or with a mix of commercial and rad-hard devices, often also including commercial FPGAs and hybrid system-on-chips (SoCs) [28]. Hybrid SoCs synergize multiple distinct computing architectures within one device to attain the architectural advantages of each. FPGA-based hybrid SoCs combine dedicated fixed-logic CPUs with reconfigurable-logic FPGAs. Commercial FPGAs and SoCs provide numerous architectural advantages and offer superior performance, energy efficiency, affordability, and capability compared

to rad-hard alternatives but are highly susceptible to radiation-induced single-event effects (SEEs) that can affect the dependability of the system and application [56]. To improve dependability, hardware-redundancy techniques such as triple-modular redundancy (TMR) are frequently employed for SEE mitigation. However, TMR incurs significant overhead in the area, power consumption, and timing-critical path which can be impractical for resource-constrained systems and can also limit the performance and energy-efficiency potential of a system. To create dependable, high-performance systems capable of onboard DL, it is essential to develop efficient approaches for SEE mitigation. Furthermore, it is also crucial to devise accurate and efficient methods of evaluating the dependability of system designs to demonstrate the effectiveness of novel dependability techniques.

In this dissertation, we propose three contributions to advance the state-of-the-art in dependable, high-performance onboard computing with commercial FPGAs and SoCs. First, we propose Hybrid, Adaptive, Reconfigurable Fault Tolerance (HARFT), a reconfigurable framework for environmentally adaptive resilience in hybrid and heterogeneous SoCs and systems for space applications. HARFT consists of a runtime-reconfigurable system architecture and a methodology for evaluating environmentally adaptive and gracefully degradable systems in near-Earth radiation environments. By adapting system resources between performance and dependability modes in response to the environmental condition, HARFT can maximize system performability subject to mission availability constraints. Next, we propose Reconfigurable ConvNet (RECON), a reconfigurable framework for dependable, high-performance semantic segmentation for space applications. RECON consists of a runtime-reconfigurable semantic-segmentation accelerator architecture and includes selective and adaptive approaches for efficient SEE mitigation. Using both approaches, RECON can maximize inference performability subject to mission availability constraints. Finally, we propose a methodology for evaluating FPGA-accelerated DL models and analyzing their performance and dependability tradeoffs. With an emphasis on the dependability evaluation, we also propose a hierarchical fault-injection approach to accelerate the characterization of fault susceptibility in DL solutions. Using this methodology, we evaluate, analyze, and compare the tradeoffs of multiple semantic-segmentation models accelerated on multiple configurations of the Xilinx Deep-Learning Processing Unit (DPU) accelerator.

This dissertation is organized as follows. Chapter 2 provides a cursory overview of SmallSats missions and autonomy, commercial FPGAs and SoCs for space applications, radiation effects and dependable computing, and DL to introduce relevant background for the dissertation research. Chapter 3 describes HARFT, including the adaptive system architecture and evaluation methodology. Chapter 4 describes RECON, including the reconfigurable accelerator architecture, efficient SEE-mitigation approaches, and evaluation. Chapter 5 describes a methodology for evaluating FPGA-accelerated DL models and analyzing their tradeoffs, including the hierarchical approach to accelerate the fault-injection process. Finally, Chapter 6 concludes this dissertation.

2.0 Background Research

This section provides a cursory overview of space-computing trends including SmallSats, CubeSats, autonomy for space missions, and commercial hybrid and heterogeneous SoCs and systems that enable dependable, high-performance onboard processing. Next, this section provides a background in DL including neural network (NN) and convolutional NN (CNN or ConvNet) basics, semantic segmentation, inference accuracy metrics, and CNN architectures and optimizations for FPGA acceleration. Finally, this section discusses topics in dependable computing such as radiation effects on electronic devices, SEE susceptibility of commercial CPUs and FPGAs, including mitigation techniques and evaluation methods, symmetric and asymmetric multiprocessing, and the modeling of adaptive and evolvable systems for near-Earth radiation environments.

2.1 SmallSats, CubeSats, and Onboard Autonomy

The application of onboard DL for spacecraft autonomy and data analysis is rapidly trending in SmallSat and CubeSat missions. SmallSats, constrained to low size and mass under 500 kg, and CubeSats, measured in Units (U) with $10\times10\times10$ cm^3 per U, have emerged as useful, high-risk, low-SWaP-C platforms enabled by the miniaturization of electronics, sensors, and instruments, and have proliferated in both science and defense missions [63, 105, 95]. The proliferation of CubeSat missions has also enabled complementary activities that use emerging techniques in big data, such as DL, to process vast CubeSat-generated datasets. For example, Planet's constellation of Dove CubeSats generates several terabytes of high-cadence, high-resolution EO image data per day [31]. However, in 2016, the National Academies' Space Studies Board (SSB) highlighted the need for fault protection and high-performance computing for spacecraft operations and payload processing for CubeSats [63]. In 2018, the SSB issued a report for the 2017-2027 decadal strategy on Earth science and applications from space, providing recommendations to the National Aeronautics and

Space Administration (NASA), National Oceanic and Atmospheric Administration (NOAA), and U.S. Geological Survey (USGS) for future missions in EO [65]. The decadal strategy accentuated the need for advanced methodologies to analyze and convert EO image data into scientific knowledge, which can be achieved using DL methods. However, the decadal strategy also emphasized the importance of mission design tradeoffs and the crucial balance of three interrelated parameters: performance, cost, and risk, which signifies the importance of considering all three parameters for a space computer capable of onboard DL. In 2011, the SSB issued a report for the 2013-2022 decadal strategy for planetary science, which emphasized the need for reduced SWaP-C constraints in spacecraft and increased spacecraft autonomy to achieve planetary science objectives [62]. Similarly, in 2019, the SSB issued a report reviewing the planetary science aspects of NASA's lunar science and exploration initiative, which highlighted the deployment of lunar CubeSats and SmallSats to achieve lunar science objectives [66].

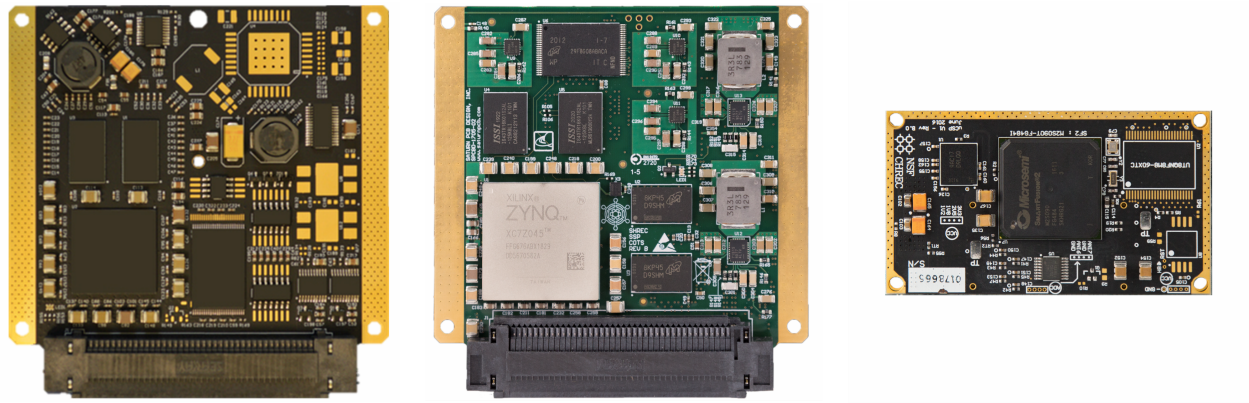
In addition to advancing science missions, SmallSat and CubeSat technology are also emerging in future defense missions. At the 2017 Small Satellite Conference keynote address, the National Geospatial-Intelligence Agency (NGA) director, Robert Cardillo, accentuated plans for the NGA to begin using immense volumes of data collected by commercial constellations of imaging SmallSats [16]. Similarly, at the 2020 Small Satellite Conference keynote address, the National Reconnaissance Office (NRO) director, Christopher Scolese, emphasized a hybrid architecture approach that combines flagship satellites with proliferated SmallSats and commercial satellites to enhance flexibility, responsiveness, resiliency, and prioritization in constellation management [90]. The Defense Advanced Research Projects Agency (DARPA) Blackjack program seeks to create a next-generation avionics unit, called Pit Boss, that will leverage both commodity and commercial technologies to enable advanced, on-orbit computing with payload-level and mission-level autonomy [18]. Blackjack aims to demonstrate that a distributed, resilient constellation of autonomous, replenishable SmallSats in low-Earth orbit (LEO) can compete with expensive, flagship spacecraft in geosynchronous orbit (GSO) [17].

2.2 Commercial Hybrid and Heterogeneous SoCs and Systems for Space Applications

To improve onboard processing capabilities that enable DL applications, SmallSat and CubeSat missions often employ commercial FPGAs and hybrid SoCs. Hybrid SoCs, such as the Xilinx Zynq-7000 SoC (Zynq-7000) [113] and Xilinx Zynq UltraScale+ MPSoC (Zynq-MPSoC) [115], combine fixed-logic CPUs with reconfigurable-logic FPGAs in a single device, with both subsystems interconnected by high-speed Advanced Microcontroller Bus Architecture (AMBA) Advanced eXtensible Interface (AXI) interfaces. The Zynq-7000 features single- or dual-core ARM Cortex-A9 APU and an Artix or Kintex 7-Series FPGA fabric interconnected by 64-bit AXI3 interfaces. The Zynq-MPSoC features a multiprocessor system, including dual- or quad-core ARM Cortex-A53 APU, dual-core ARM Cortex-R5 RPU, TMR MicroBlaze PMU, and an UltraScale+ Architecture FPGA fabric interconnected by 128-bit AXI4 interfaces. In both SoC series, the CPU and FPGA subsystems can interact over AXI for general-purpose and high-performance memory-mapped accesses. Both series also include configuration access ports (CAPs) that enable interactions with the FPGA configuration controller for FPGA reconfiguration and access to configuration memory (CRAM). These ports include the processor CAP (PCAP) and internal CAP (ICAP) which are accessible by the CPU and FPGA, respectively. In addition to full reconfiguration (FR), Xilinx FPGAs and SoCs support partial reconfiguration (PR) that allows predefined partitions, called PR regions (PRRs), to be reconfigured with compatible modules, called PR modules (PRMs) at runtime without interrupting the remainder of the system, including the CPU and logic in the static region (SR) and other PRRs. PR presents numerous advantages for space applications, including runtime reconfiguration without compromising system uptime, rotation of time-multiplexed PRMs for resource-constrained FPGAs, and reconfiguration to extend the system for post-mission operations and objectives [69].

The Microchip SmartFusion2 SoC, which combines a single-core ARM Cortex-M3 with flash-based IGLOO2 FPGA fabric interconnected by AMBA AHB-Lite or APB interfaces, is another commercial SoC suitable for low-power applications. The flash-based CRAM also improves the susceptibility of the FPGA to radiation-induced SEEs at the tradeoff of limited reconfigurability.

Space missions increasingly continue to adopt hybrid and heterogeneous SoCs and systems for onboard processing. The CHREC Space Processor (CSP) [109, 106] and SHREC Space Processor (SSP) [82] are two examples of multifaceted hybrid space computers, both illustrated in Figure 1. CSP was developed by researchers at the National Science Foundation (NSF) Center for High-Performance Reconfigurable Computing (CHREC) in collaboration with NASA Goddard Space Flight Center (GSFC), and SSP was developed at the NSF Center for Space, High-Performance, and Resilient Computing (SHREC), which superseded CHREC in 2018, at the University of Pittsburgh in collaboration with government and industry partners. CSP and SSP are both 1U compute cards that feature a Zynq-7000 (Z7020 or Z7030/Z7030/Z7045) and combine a novel mix of commercial technology (processor and memory) for performance, rad-hard technology (monitoring and managing circuits) for dependability, and supplementary dependable computing for extended reliability enhancements. µCSP, another hybrid space computer developed at CHREC, is a sub-1U system-on-module (SoM) that features a Microchip SmartFusion2 SoC (M2S090) designed for low-SWaP-C applications [107, 106]. CSP has flight heritage as part of two U.S. Department of Defense Space Test Program (STP) Houston missions to the International Space Station (ISS), including STP-H5 CHREC Space Processor (STP-H5-CSP) and STP-H6 Spacecraft Supercomputing for Image and Video Processing (STP-H6-SSIVP) [109, 106, 85]. CSP was also flown on the NASA CeREs heliophysics-science CubeSat [38] and Seeker free-flying inspector CubeSat [71], and it will be featured on the Lockheed-Martin LunIR lunar-flyby CubeSat [80], the NASA Mass Spectrometer observing lunar operations (MSolo) instrument [2], and several other planned missions. µCSP also has flight heritage as part of STP-H6-SSIVP. CSP, SSP, and µCSP are also planned for flight on the STP-H7 Configurable and Autonomous Sensor Processing Research (STP-H7-CASPR) mission to the ISS [82]. All



(a) CSP (Rev. B EM) (b) SSP (Rev. B EM) (c) µCSP (Rev. B FM)

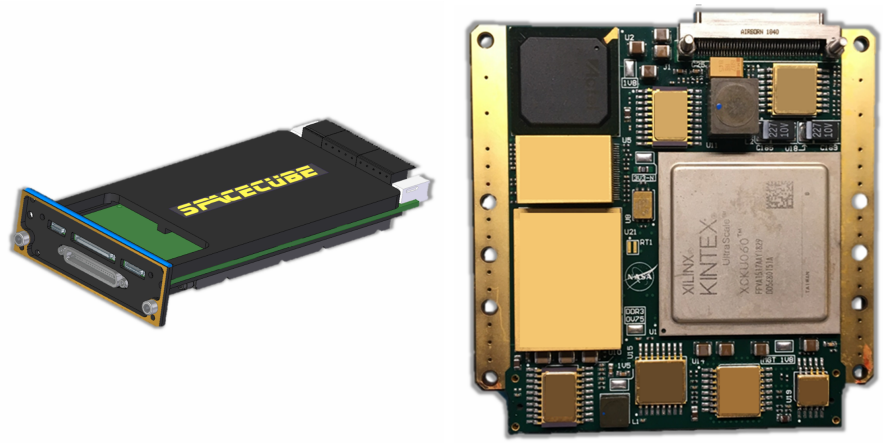
Figure 1: Hybrid space computers developed at NSF SHREC include (a) CSP, (b) SSP, and (c) µCSP. EM and FM refer to engineering and flight models, respectively.

three STP mission experiments that include CSP, SSP, or µCSP are illustrated in Figure 2. Derivatives of CSP include the SHREC Hybrid Computer (SHC) and SpaceCube Mini-Z [13], both developed at NASA and featured on many new science missions.

The Science Data Processing Branch at NASA GSFC is also developing two next-generation hybrid space computers, including the SpaceCube v3.0 VPX (SCv3VPX) [27] and SpaceCube v3.0 Mini (SCv3M) [13], both illustrated in Figure 3, as the next generation of hybrid space computers for future missions. SCv3VPX is a 3U SpaceVPX Lite card that features the Zynq-MPSoC and Kintex UltraScale FPGA (KU-FPGA) with both devices interconnected by multi-gigabit transceivers (MGTs) and supervised by a rad-hard Microchip RTAX FPGA. SCv3M is a 1U card that features the KU-FPGA, supervised by a Microchip RT ProASIC3, and can be paired with a processor card (e.g., SSP) for external management or self-managed with a soft-core processor (e.g., MicroBlaze or RISC-V).

(a) STP-H5-CSP (b) STP-H6-SSIVP (c) STP-H7-CASPR

Figure 2: STP mission experiments developed at NSF SHREC that feature CSP, SSP, or µCSP include (a) STP-H5-CSP, (b) STP-H6-SSIVP, and (c) STP-H7-CASPR.

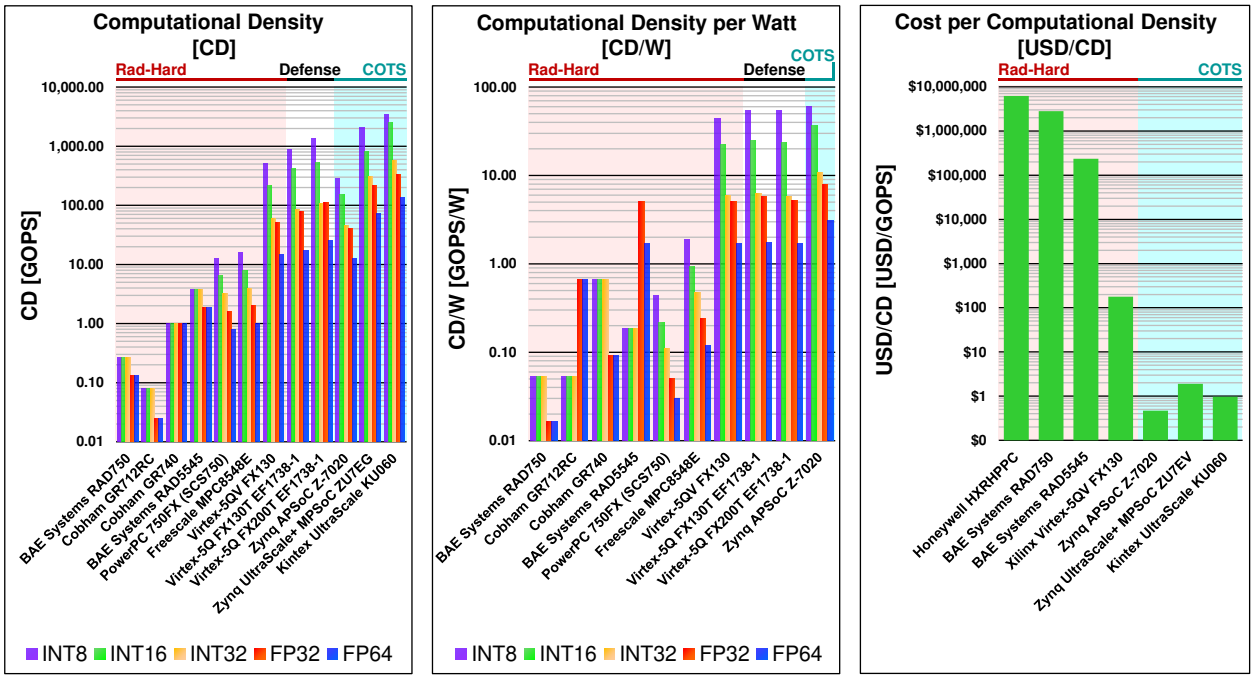


(a) SCv3VPX (b) SCv3M

Figure 3: Hybrid space computers developed at NASA GSFC include (a) SCv3VPX and (b) SCv3M.

Alternative examples of single-board computers featuring a Zynq-7000 include the Innoflight Compact Flight Computer (CFC-300), GomSpace Nanomind Z7000, and Xiphos Q7. Alternative examples that feature a Zynq-MPSoC include the Innoflight CFC-400 and Xiphos Q8.

Lovelly et al. [55] developed a methodology using device metrics to analyze and compare the performance and energy-efficiency of select commercial, defense-grade (improved packaging and parts qualification), and rad-hard processors for onboard computing. These device metrics include the *computational density* (CD), measured in giga-operations per second (GOPS), and *computational density per Watt* (CD/W), which approximate the potential performance and energy-efficiency, respectively, for a processor. Using this methodology, the commercial Zynq-7000, Zynq-MPSoC, and KU-FPGA all demonstrated substantial improvements over state-of-the-art rad-hard processors in both metrics, in addition to cost-to-performance efficiency in U.S. Dollars per CD (USD/CD), highlighting the advantages of using commercial and hybrid architectures for space applications.



(a) CD (b) CD/W (c) USD/CD

Figure 4: Comparison of space processors in terms of (a) CD[a], (b) CD/W, and (c) USD/CD[b] [55, 27, 13].

[a]ZU7EG and KU060 metrics are estimates extrapolated from existing data.
[b]Cost based on average estimated price over survey data (2020).

2.3 Radiation Effects

Radiation sources are numerous and include galactic cosmic rays (GCRs), solar particle events (SPEs), and charged particles trapped within the Van Allen radiation belts. In the near-Earth radiation environment, GCRs are mostly composed of protons but also include alpha particles and heavy ions. SPEs include coronal mass ejections (proton-rich) and solar flares (heavy-ion-rich). The Van Allen radiation belts, which reside in the Earth's magnetic (geomagnetic) field, are primarily composed of protons and electrons. The geomagnetic field provides shielding from GCRs and solar wind. However, geomagnetic shielding is dependent upon the solar weather condition [42, 12].

The charged particles trapped in the Van Allen radiation belts are governed by three periodic motions: gyration about the magnetic field lines, bounce between mirror points near the magnetic poles, and longitudinal drift with positively charged particles drifting westward and negatively charged particles drifting eastward [39]. Collectively, these periodic motions cause charged particles to drift about a toroidal surface, called a *drift shell*. The McIlwain L-shell (L_m), from McIlwain's (B, L) geomagnetic coordinate system, labels these drift shells (set of geomagnetic-field lines) that cross the geomagnetic equator in units of Earth radii ($R_\oplus$) from the geomagnetic center. The L_m can be used to estimate the fluxes of trapped particles within the geomagnetic field and GCRs attenuated after geomagnetic shielding. Using the NASA trapped particle radiation models (AP-8/AE-8), the proton and electron fluxes are indexed using the following parameters: energy, B, and L_m.

Radiation presents several challenges for electronic devices in space. Radiation effects on electronic devices are typically categorized into long-term cumulative effects and short-term transient effects. Cumulative effects include total ionizing dose (TID) and displacement damage dose (DDD). TID refers to the ionizing-radiation dose absorbed by the device material over time causing parametric or functional degradation (e.g., threshold-voltage shifts, timing changes, current-leakage increase). DDD refers to non-ionizing damage that occurs when radiation particles collide with atoms of the lattice structure to form Frenkel pairs (a vacancy and interstitial atom).

SEEs are transient effects that occur due to direct-ionization or indirect-ionization processes. Direct-ionization-induced SEEs typically occur when a single heavy-ion particle traverses the device depositing enough charge to cause an effect. Indirect-ionization-induced SEEs often occur when protons, or other species of radiation, form nuclear interactions near the sensitive device causing an effect. SEEs can be destructive or nondestructive. Destructive SEEs include single-event latch-up (SEL), single-event burnout (SEB), and single-event gate rupture (SEGR), among others. Nondestructive SEEs include single-event upset (SEU), single-event transient (SET), and single-event functional interrupt (SEFI). Both cumulative and transient effects are extensively covered by the National Academies' report on the U.S. infrastructure for space radiation effects testing [64].

Several examples of spacecraft system failures and anomalies attributed to the harsh space environment, including radiation, are covered in [8, 21]. To improve mission dependability, NASA created the Radiation Hardness Assurance (RHA), a multi-step approach to address radiation concerns in spacecraft development [43, 44]. NASA further evolved RHA for SmallSat missions that require high reliability but are too cost-constrained to follow standard RHA practices [14].

2.4 Dependable Computing

Despite numerous architectural advantages provided by commercial hybrid SoCs, these devices are infrequently deployed in NASA-qualified space avionics due to their high susceptibility to radiation. The architectural response to SEEs differs between CPUs and FPGAs, and several techniques in dependable computing exist for both architectures. A comprehensive overview of the radiation effects on CPUs, including hardware and software mitigation techniques, is covered in [40, 78, 81]. Similarly, an overview of radiation effects on FPGAs, including fault-masking and fault-tolerance techniques, is provided in [110, 92] as well.

2.4.1 CPU Dependability

For CPUs, errors in the processor registers (general-purpose, special-purpose, and internal), caches, and memories (on-chip and external) can cause a variety of adverse effects, including data errors, silent data corruption (SDC), program crashes, system resets, and performance degradation [78, 81]. Various techniques exist for error mitigation in CPUs in terms of hardware, information, network, software, and time redundancy. Error detection and correction (EDAC) mechanisms, including error-correcting code (ECC), cyclic redundancy check (CRC), and parity, are often integrated into the CPU architecture to improve reliability. Time TMR (TTMR) can be used to run redundant replicas of software processes over time. Process-level redundancy (PLR) can be used to schedule replicas across available resources to reduce overhead [91]. Using algorithm-based fault tolerance (ABFT), input operands are encoded and processed to yield an encoded output that can be used to detect or correct errors [41]. One form of hardware redundancy for CPUs is lockstep operation, which involves two or more synchronized CPUs running the same software at the same instruction order, with the processor states (i.e., registers or interconnects) running through a comparator or majority voter for duplex-with-compare (DWC) or TMR operation, respectively. CPUs in lockstep operation can be combined with checkpoint and rollback recovery to minimize the latency of resynchronization [1].

2.4.1.1 Symmetric and Asymmetric Multiprocessing

Some multicore processors can operate in both symmetric multiprocessing (SMP) and asymmetric multiprocessing (AMP). In SMP, cores operate jointly under one operating system (OS). SMP provides performance benefits by facilitating multitasking (parallel execution of multiple processes) and multithreading (parallel execution of one process). However, due to the tight coupling of cores in SMP operation, the corruption or functional failure of one core due to radiation-induced errors can render the entire OS inoperable. In contrast, AMP cores are partitioned, with each partition running an independent OS (i.e., one partition runs Linux and the other runs bare-metal or FreeRTOS). AMP limits the capability for parallel processing but provides some degree of isolation to mitigate error propagation between

partitions. Thus, the corruption of one OS does not necessarily render the other inoperable. AMP provides several additional capabilities suitable for space applications, including (1) segregation of critical flight software (FSW) and science applications, (2) hybrid combination of OS paradigms (e.g., interactive and real-time OSs), (3) redundant processing for dependable computing, and (4) physical zoning for security applications. The Zynq-7000 and Zynq-MPSoC are both capable of SMP and AMP operation. Heterogeneous, multiprocessor SoCs, such as the Zynq-MPSoC, enable another dimension of feasible AMP configurations. Furthermore, the Zynq-MPSoC CPU architecture supports hardware extensions for efficient virtualization. Hypervisors, such as Xen, can physically partition cores, with each partition running its own virtual machine. Hypervisor-based fault-tolerance for space applications has been explored by Sabogal et al. [84].

2.4.2 FPGA Dependability

Many FPGAs and FPGA subsystems in hybrid SoCs are SRAM-based. SRAM-based FPGAs are high-density, high-reconfigurability architectures composed of many diverse resources, including logic blocks and hard blocks (e.g., DSPs, BRAM, and high-speed I/O), interconnected by a complex routing network. At runtime, a design bitstream is stored in CRAM to configure the resources and network routing to implement a design onto the FPGA. This paradigm provides designers with the flexibility to create customized, massively parallel datapaths to accelerate compute-intensive algorithms on FPGAs as well as the capability to reconfigure the FPGA fully or partially at runtime to multiplex applications or system configurations over time. However, despite these architectural advantages for onboard processing, SRAM-based FPGAs and SoCs are seldom deployed in NASA-qualified avionics due to their high susceptibility to radiation, which can introduce faults that manifest into a variety of error and failure modes. To address radiation concerns, many NASA missions deploy rad-hard, flash-based, or antifuse-based FPGAs, which are relatively or completely immune to CRAM faults, instead of SRAM-based FPGAs. Faults in static CRAM bits, which configure the FPGA to realize the design, can cause functional changes in the design. Examples include misconfiguration of resources (e.g., misconnected signals or functional changes) and routing

(e.g., misrouted or missing signals), as illustrated in Figure 5. Faults in dynamic CRAM bits, which are typically used for distributed RAM or shift registers, and other design-specific memories (e.g., BRAM, flip-flops, and internal hard-block registers) can also cause a wide variety of adverse effects. Typically, design-specific memories can be protected with error correction code (ECC) to improve dependability. A comprehensive overview of the radiation effects on FPGAs, including SEE mitigation techniques for fault masking, avoidance, and tolerance, is covered in the literature [110, 92, 77].

2.4.2.1 Dependability Techniques

TMR is a hardware redundancy technique that involves triplicating circuits and routing the outputs through majority voters for single-fault masking. TMR can improve the reliability of a design; however, triplication incurs a high overhead in the device resource utilization, energy consumption, and timing-critical path of a design, which can reduce performance or even detrimentally increase the critical area (critical bits) vulnerable to faults. The granularity at which triplication is applied can vary. Fine-grain TMR (FG-TMR) involves triplication of intra-modular circuits with more frequent voters to mask low-level faults (e.g., circuits within the module are triplicated with voters inserted at the inputs of each flip-flop), and coarse-grain TMR (CG-TMR) involves triplication of entire modules to mask module-level faults (e.g., voters inserted at the modular interfaces). Generally, fine-grain replication provides greater reliability due to more frequent voters that inhibit the propagation of errors, as illustrated in Figure 6, whereas coarse-grain replication provides greater area efficiency [96]. Furthermore, reliability-aware placement and routing methods have been explored to disaggregate replicas of TMR designs across separate domains to minimize common-mode failures [93, 15]. DWC is another form of hardware redundancy that involves duplicating circuits and comparing their outputs for single-fault detection.

A variety of tools have been developed for the automatic replication and insertion of majority voters or comparators of FPGA designs. Commercial tools, such as Xilinx TMRTool, Synopsis Synplify Premier, and Mentor Graphics Precision Hi-Rel can triplicate designs at the RTL level during synthesis, along with other reliability features. BL-TMR is an

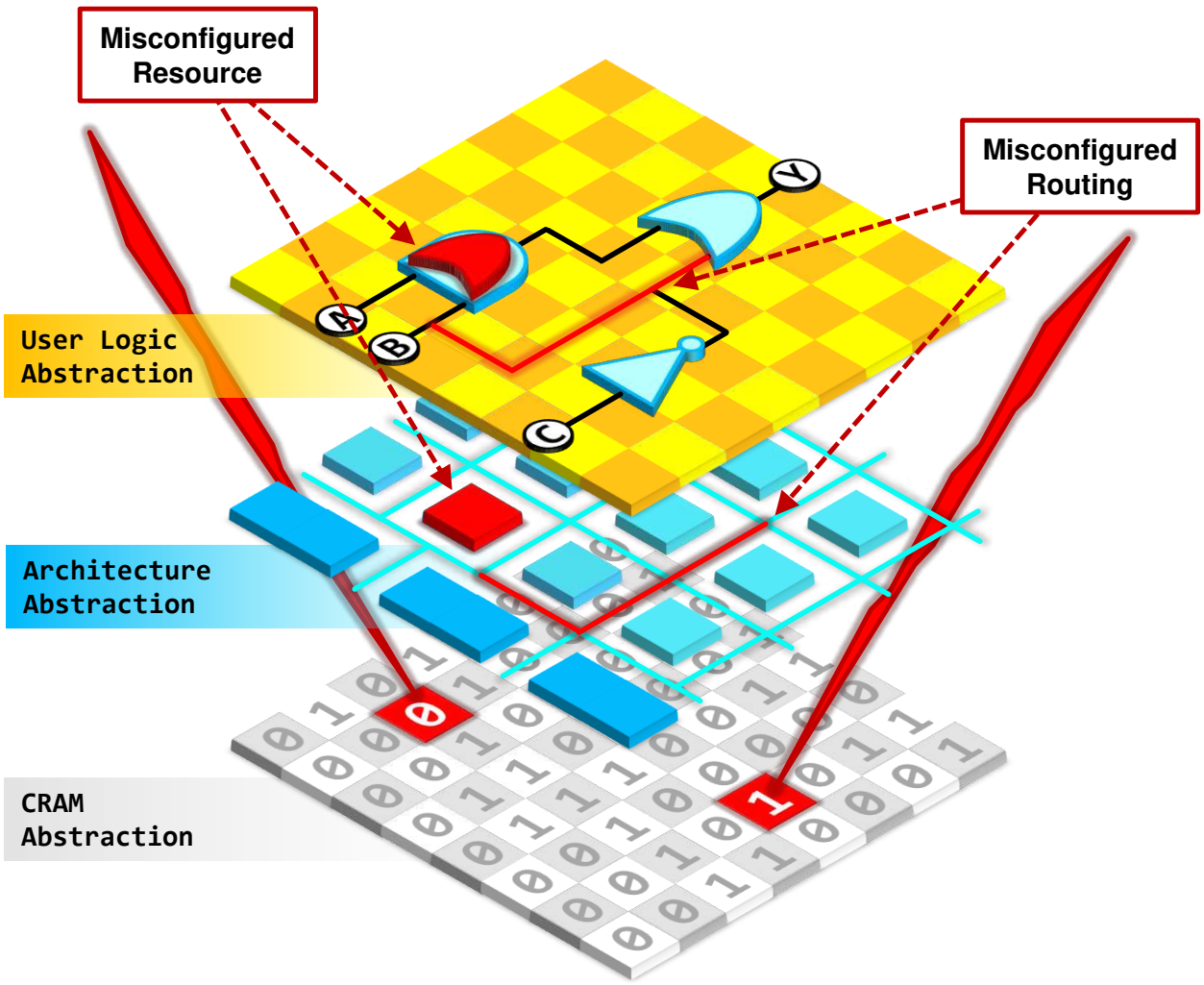
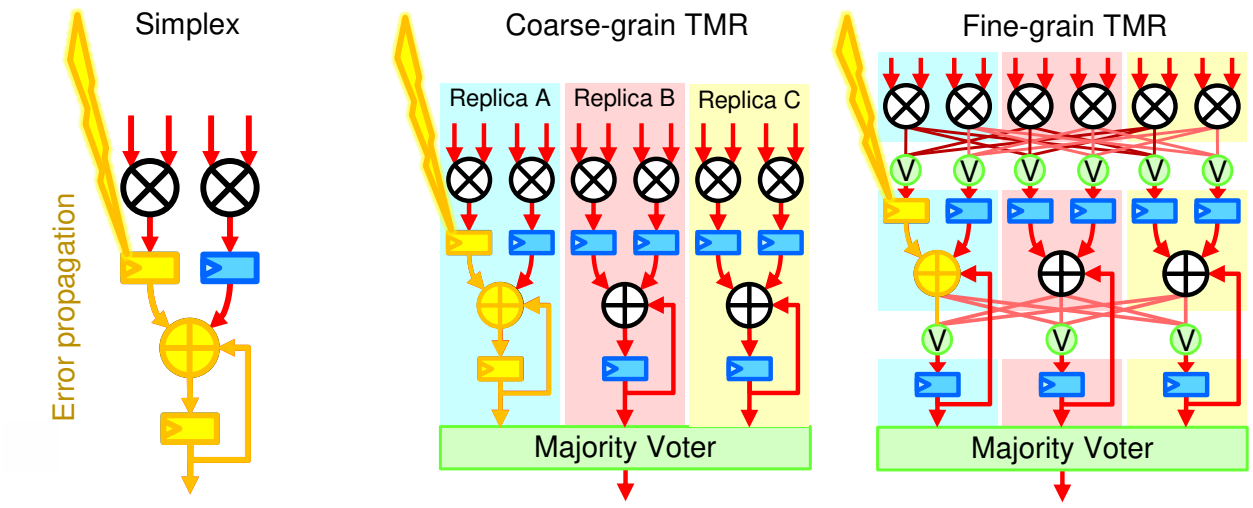


Figure 5: Functional changes of FPGA design due to SEEs in CRAM.



(a) Simplex design (b) CG-TMR design (c) FG-TMR design

Figure 6: Granularity of TMR in FPGA designs with (a) the simplex design protected by (b) CG-TMR and (c) FG-TMR.

academic tool that supports selective replication of designs in a post-synthesis netlist [37]. TLegUp is an academic extension of the LegUp tool that provides compiler directives and fault-tolerance-aware scheduling and binding to triplicate high-level synthesis designs [51].

Hardware redundancy can be combined with PR for module-based error recovery (MER) [11]. In this paradigm, the replicas of a CG-TMR design are PRMs residing in their independent PRRs with majority voters inserted in the SR near the PRR boundaries. When module-based errors are detected, the majority voters signal a reconfiguration controller residing in the SR to reconfigure faulty PRMs for recovery. Various network topologies and strategies for MER have been explored in the literature [3, 121, 122].

Hardware redundancy can also be combined with CRAM scrubbing to prevent the accumulation of faults in CRAM that can overwhelm single-fault masking techniques like TMR. CRAM scrubbing is a background process that detects and corrects faults in CRAM. On Xilinx FPGAs, scrubbing architectures can be implemented on-chip using the PCAP or ICAP or off-chip using JTAG or SelectMAP [10], and the scrubbing approach can be categorized

into blind, readback, replacement, or hybrid forms [94]. A variety of optimization techniques for CRAM scrubbing have been explored, such as prioritizing CRAM frames by number of critical bits [88] or exploiting frame-level redundancy of TMR designs to repair fault frames using the frame of the replica [98].

2.4.2.2 Dependability Evaluation of FPGA Designs

The dependability of a full or partial design of an FPGA can be measured experimentally through fault-injection or radiation-beam testing. In CRAM fault injection, a bit-flip is injected into CRAM to observe the architectural response to the fault during design operation. Two metrics of interest for quantifying the dependability of a design include the architectural vulnerability factor (AVF) and mean-work-to-failure (MWTF). In the context of this dissertation, the AVF of a design is the probability that an injected fault will manifest into an observable event [61], and MWTF describes the amount of useful work completed until an observable event is expected [79]. The classification of observable events is user-defined and can vary by design or application (e.g., SDC or hangs). AVF and MWTF are calculated using Equations (2.1) and (2.2), respectively.

$$\text{AVF} = \frac{\text{Number of Observable Events}}{\text{Number of Fault Injections}} \tag{2.1}$$

$$\text{MWTF} = \frac{\text{Amount of Useful Work Completed}}{\text{Number of Observable Events}} \tag{2.2}$$

Radiation-beam testing involves irradiating devices-under-test (DUTs) by high-energy radiation or laser beam to induce SEEs. In radiation-beam testing, the beam flux (number of particles per unit area per second) or fluence (integration of flux over time; the number of particles per unit area per second) are recorded in addition to the observed events. One metric of interest is the cross-section (σ), which is the sensitive area of the DUT where

a radiation-induced fault will manifest into an observable event [76]. The cross-section is calculated by dividing the number of observable events by the beam fluence using Equation (2.3). In practice, the AVF and cross-section results are reported with the corresponding 95% confidence interval (CI) error to provide context for uncertainty in the measurements of the experiment [76].

$$\sigma = \frac{\text{Number of Observable Events}}{\text{Total Effective Fluence}} \tag{2.3}$$

2.4.2.3 Environmentally Adaptive Resilience for Near-Earth Radiation Environments

Due to the dynamics of the near-Earth radiation environment, influenced by the geomagnetic field, solar weather, and other phenomena, spacecraft are exposed to wide variations of radiation fluxes resulting in SEE rates that can vary by multiple orders of magnitude depending upon the orbit [12, 111]. Jacobs et al. [35] and Sabogal et al. [87] proposed methodologies for modeling and evaluating adaptive and evolvable systems in near-Earth radiation environments.

First, the dynamic radiation environment is modeled using the combination of multiple well-established models to predict the time-varying SEE rates of a device. Simplified General Perturbation (SGP4) [33] is an orbital-perturbation model that can predict the geographic coordinates of the orbital position of near-Earth objects over a period. International Geomagnetic Reference Field (IGRF) [97, 103] is a geomagnetic-field model that can map the geographic coordinates to the McIlwain L-shell (L_m), which labels the drift shells that cross the geomagnetic equator in units of Earth radii ($R_\oplus$) from the geomagnetic center. Using the NASA trapped particle radiation (AP-8/AE-8) and Cosmic Ray Effects on Micro-Electronics (CRÈME96) models, the L_m can be used to estimate the fluxes of trapped particles within the geomagnetic field and GCRs attenuated after geomagnetic shielding. CRÈME96 [101],

developed by Vanderbilt University and supported by NASA, is a state-of-the-art tool that uses phenomenological models with device, mission, orbital, and environmental characteristics to predict SEE rates induced by protons and heavy ions. CRÈME96 can also predict the average SEE rates for a specific orbital segment between two drift shells bounded by lower and upper L_m. These segmented SEE rates can be assigned to the time domain by mapping SEE rates to the L_m of the spacecraft over time.

Next, for a specific observable event, the time-varying fault rate of an FPGA design ($\lambda_{\text{design}}(t)$) can be approximated using Equation (2.4). For each resource type ($r \in R$), the aggregated resource SEE rates ($\lambda_{r,\text{SEE}}(t)$) is scaled by the resource utilization (RU_r) times the resource AVF (AVF_r). In cases where determining the AVF of a specific resource type is infeasible, an estimate is made (e.g., assume worst-case or use another resource AVF). The final design fault rate is the summation of the scaled fault rates for all resource types.

$$\lambda_{\text{design}}(t) = \sum_{r \in R} \lambda_{r,\text{SEE}}(t) \cdot \text{RU}_r \cdot \text{AVF}_r \tag{2.4}$$

Finally, phased-mission system modeling is used to model adaptive and evolvable systems, where failure, recovery, and performance mechanisms change over time. At each phase of the mission, the system configuration can be modeled using continuous-time Markov chains (CTMCs), with time-varying fault rates, repair rates, and reward rates assigned to represent failure, recovery, and performance mechanisms, respectively. The instantaneous and average availability, failure rate, and performability of the system can be calculated by performing a transient analysis of the phased-mission system model. Availability describes the probability that a system is operational. The failure rate describes the rate at which a system enters a failure state. Finally, performability describes the amount of useful work completed and is dependent on system availability

2.5 Deep Learning

DL, particularly CNNs, have become increasingly popular in ML and computer-vision (CV) applications for classification, detection, localization, and segmentation tasks on image data [89] and can be applied to enhance applications in EO and remote sensing [7]. CNNs are a form of classical supervised learning algorithms with a feed-forward process for inference and a back-propagation process for training. CNNs consist of a combination of layers (or nodes) that operate on feature maps (FMs). Convolutional layers extract features of input FMs and generate new FMs that represent the locations and strengths of detected features. Each convolutional operation contains a set of learnable weights and biases that are formulated during model training. Initial convolutional layers extract low-level features (e.g., edges, corners, surfaces), and deeper layers extract more complex abstractions (e.g., structures and patterns). Activation layers (e.g., sigmoid, tanh, and rectified linear unit (ReLU)) introduce nonlinearity into the model to approximate nonlinear patterns and functions. Pooling layers (e.g., max pooling and average pooling) downsample and discretize the spatial resolution of input FMs to reduce the number of parameters and operations. Fully connected layers perform classification and map features extracted from previous layers into an output vector of classes. The arguments of the maxima (argmax) of the output vector specify the most probable classification of the input for class label assignment. CNNs can append an optional softmax layer to convert the output vector into a discrete probability-distribution vector to determine the confidence of the classification. Batch normalization (BatchNorm) is another layer that can be inserted between convolutional and activation layers to accelerate training and mitigate overfitting through the normalization of the inputs.

2.5.1 Semantic Segmentation

Semantic segmentation is a process that labels each pixel of an image, where pixels with the same label share the same semantic characteristics. The application of DL to perform semantic segmentation has been explored extensively in the literature [26, 45]. Five examples of semantic-segmentation models evaluated in this dissertation research include SegNet [6],

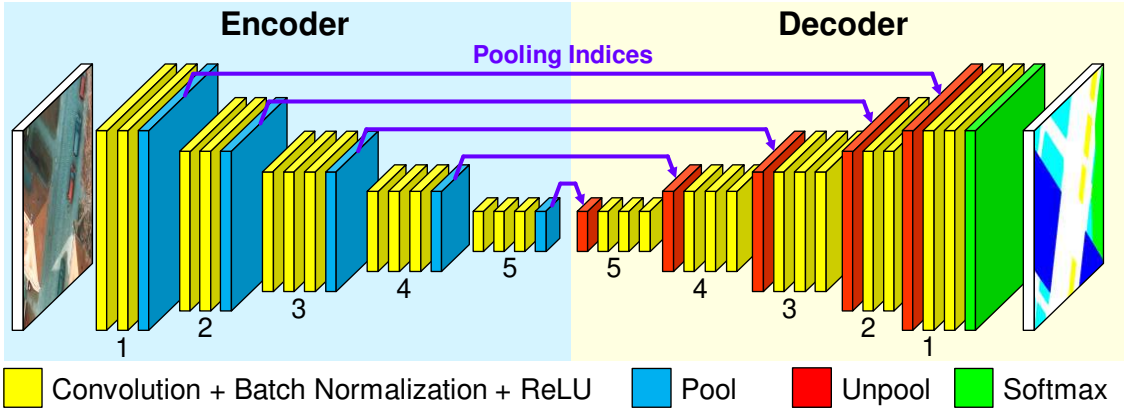


Figure 7: SegNet semantic-segmentation model.

U-Net [83], Efficient Neural Network (ENet) [70], Feature Pyramid Network (FPN) [54], and Efficient Spatial Pyramid Neural Network (ESPNet) [57], each with unique characteristics. SegNet, illustrated in Figure 7, is a symmetric autoencoder that contains five encoder blocks followed by five decoder blocks, each with two or three sets of convolutional, BatchNorm, and ReLU layers. Each encoder block is followed by a max-pooling layer which produces two outputs: downsampled FMs and pooling indices (PIs). Each decoder block begins with a max-unpooling layer which uses the PIs of the corresponding encoder block to upsample smaller FMs back to their original spatial resolution. SegNet uses PIs to perform nonlinear upsampling without the need to learn to upsample. An optional softmax layer can be appended at the end of the network to generate a discrete pixel-wise probability distribution. The argmax of the output layer can also be used to assign the most probable label for each pixel. U-Net is a symmetric autoencoder and contains contracting and expanding data pathways. The contracting pathway is a sequence of encoder blocks that perform feature extraction and pooling-based downsampling, and the expanding pathway is a sequence of decoder blocks that perform deconvolutional upsampling using feature maps (FMs) from preceding and lateral blocks. ENet is an asymmetric autoencoder that uses dilated convolution to achieve a larger receptive field of each convolutional filter, stores pooling indices from early pooling layers for unpooling-based upsampling, and factorizes filters to drastically

reduce the number of parameters. FPN is a pyramidal structure with bottom-up and top-down pathways. The bottom-up pathway generates FMs with increasing semantic value at decreasing resolution, and the top-down pathway reconstructs higher resolution layers using FMs from the preceding and lateral levels. Finally, ESPNet is an asymmetric autoencoder that uses efficient spatial pyramid convolutional modules, which perform point-wise convolution followed by a spatial pyramid of dilated convolutions that significantly decreases the number of parameters required while maintaining a large receptive field.

To evaluate the accuracy of semantic-segmentation models, two common metrics include the intersection-over-union (IoU; Jaccard index) and F1-score (F1; Dice score). The IoU is the area of intersection (overlap) divided by the area of union between the predicted output and ground-truth label mask. The F1 is defined as the harmonic mean of precision and recall. Both metrics range from 0% to 100% (higher is better) and can be calculated from a confusion matrix, which compares the predicted output and the ground-truth label in terms of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) using Equation (2.5). In multi-class segmentation, the mean IoU (mIoU) and mean F1 average the IoU and F1, respectively, across all classes.

$$\text{IoU} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}} \quad \text{and} \quad \text{F1} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} \tag{2.5}$$

2.5.2 FPGA Acceleration of DL Applications

The acceleration of DL on FPGAs has been explored extensively in the literature [30, 59]. Researchers have explored various model-compression, algorithmic, and architectural optimization techniques to efficiently map CNN algorithms to FPGAs. Model compression techniques, such as weight pruning and data quantization, can improve hardware efficiency at the cost of decreased accuracy. Weight pruning is a sparsification technique that removes weights with negligible representation in the model. Data quantization replaces high-

precision, resource-intensive floating-point (FP) data and arithmetic with low-precision integer or fixed-point to reduce the bandwidth, storage, energy, and area requirements for each operation. The 8-bit integer (INT8) quantization scheme proposed by Jacob et al. [34] constrains the continuous input set (FP) to a discrete set (INT8) using scale and zero-point parameters to map between real numbers and integers.

Additionally, algorithmic optimization techniques, such as fast convolution algorithms, BatchNorm folding, and loop optimizations, can improve the parallelism and efficiency of the accelerator architecture for FPGAs. Fast convolution algorithms, such as Winograd [46] and frequency-domain convolution, can improve the hardware efficiency of convolutional operations. Winograd convolution uses an algorithmic strength reduction technique to reduce strong operations (multiplications) at the expense of increased weak operations (additions) and is computed using Equation (2.6), where G, B, and A are Winograd transformation matrices. The weights g and FMs d are converted to Winograd space using the transformations $U = GgG^T$ and $V = B^TdB$, respectively. Next, the transformed weights and FMs undergo elementwise multiplication $W = U \odot V$ to produce the output in Winograd space. Finally, the outputs are converted back to normal space using the inverse transformation $Y = A^TWA$.

$$Y = A^T \left[\left(GgG^T \right) \odot \left(B^TdB \right) \right] A \tag{2.6}$$

The $F(2\times2, 3\times3)$ form of Winograd convolution requires a 3×3 weights matrix and 4×4 data matrix (subset of a FM) to compute a 2×2 output matrix. Compared to direct 3×3 convolution, the $F(2\times2, 3\times3)$ form has a 2.25× improvement in multiplication efficiency (number of multiplies to compute one output pixel) at a 2.625× increase in addition operations, which can minimize utilization of limited DSP resources. BatchNorm folding is a technique for embedding the parameters of a BatchNorm layer into the weights and biases of the preceding convolutional layer. BatchNorm folding is performed prior to model deployment, which can eliminate the need for processing BatchNorm layers at runtime.

Loop optimization techniques include unrolling, tiling, and interchange. Loop unrolling, combined with pipelining, exploits parallelism by executing multiple iterations of a loop using FPGA resources in parallel. Loop interchange involves reordering loop iteration variables to improve the efficiency of cache usage. Loop tiling is used to partition large FMs into tiles that can fit in on-chip memory (OCM), such as block RAM (BRAM) or Xilinx UltraRAM, for accumulation or caching to reduce the bandwidth requirement for off-chip memory access. Zhang et al. [119] proposed using design space exploration to determine the optimal parameters for loop optimization techniques given memory-bandwidth and resource constraints. Zhang et al. also proposed a roofline model to quantitatively analyze an accelerator design and determine whether the design is compute-bound or memory-bound.

Finally, architectural optimizations, such as systolic arrays, DSP time-multiplexing, and layer fusion, can further improve CNN acceleration. Wei et al. [104] proposed a 2D systolic array architecture for accelerating convolutional layers composed of processing elements (PE), each often implemented using one DSP slice. For each cycle, in a weight-stationary topology, every PE performs a multiply-accumulate (MAC) operation and shifts its input FM element and MAC output to adjacent PEs in a rippling flow. Because systolic arrays replace global multiplexers with interconnects between adjacent PEs, CNN accelerators based on this topology can achieve high-frequency operation. Furthermore, the DSP slices of Xilinx 7-Series, UltraScale, and UltraScale+ FPGAs are rated for high-frequency operation. DSP time-multiplexing involves reducing the number of DSP slices by a factor N and operating them at N times the frequency of surrounding logic to accomplish the same amount of computation [30]. This frequency technique can improve DSP efficiency but requires maintaining matched routing to ensure synchronization between two clock domains. Layer-fusion (cross-layer scheduling) techniques can improve latency and minimize off-chip memory access by fusing and executing multiple adjacent layers in a pipeline [30]. Miscellaneous, channel-wise operations can often be performed as a preprocess or postprocess of the main convolutional operation. Because multiple layers are processed in the same stream, the total number of streams and layer operations is reduced. These optimization techniques all demonstrate the viability, advantages, and limitations of FPGAs for CNN acceleration.

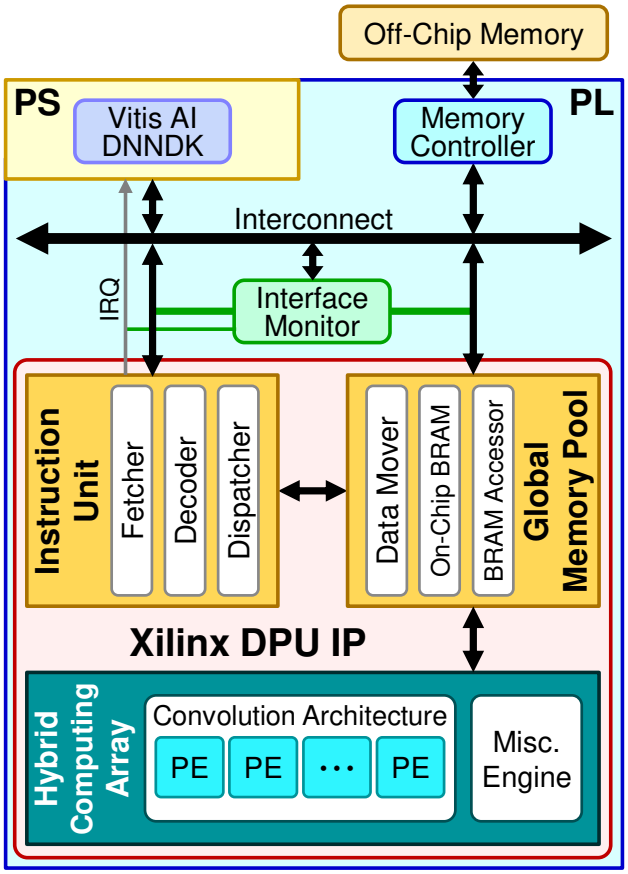


Figure 8: Xilinx Deep-Learning Processing Unit (DPU) architecture.

One example is the Xilinx DPU [118], the hardware component of the Vitis AI stack [117], illustrated in Figure 8. The DPU is a general-purpose DNN accelerator that uses a coprocessing architecture. The DPU includes (1) an instruction unit, which performs instruction fetching and scheduling of node-level operations on FM data, (2) a global memory pool, which manages on-chip and off-chip memory buffers, and (3) a hybrid computing array, which is composed of a scalable number of processing engines (PEs) that perform multiply-accumulate (MAC) and other miscellaneous operations for node processing.

The Vitis AI stack features a variety of model-compression, algorithmic, and architectural optimizations to efficiently map DL models onto the FPGA for acceleration. Model-compression optimizations include parameter pruning and data quantization that improve efficiency at the expense of minimally decreased accuracy. Pruning removes parameters with minimal representation in the model to reduce the model size, and quantization replaces resource-intensive floating-point data with low-precision integer data to reduce area, bandwidth, energy, and storage requirements. Next, algorithmic optimizations include loop optimizations (unrolling, tiling, and interchange) to maximize the data-flow efficiency and cache performance of on-chip memory. Finally, architectural optimizations include DSP time multiplexing and node fusion. The DPU operates DSP resources at twice the frequency of surrounding logic to accomplish the same amount of computations with only half of the DSP resources. Node fusion involves fusing multiple node operations into one supernode to improve latency and efficiency.

3.0 Environmentally Adaptive Resilience in Reconfigurable Space Systems

Despite the architectural advantages of commercial hybrid SoCs, these devices are highly susceptible to radiation. Hardware-redundancy techniques, such as DWC and TMR, are often employed to mitigate radiation-induced SEEs to improve system dependability; however, these techniques often incur substantial overhead that can limit system performance and energy efficiency. Furthermore, due to the dynamics of the near-Earth radiation environment, influenced by the geomagnetic field, solar weather, and other phenomena, spacecraft are exposed to wide variations of radiation fluxes resulting in SEE rates that can vary by multiple orders of magnitude depending upon the orbit [12, 111]. Traditionally, system designs are static (nonchanging) and are designed conservatively to satisfy the worst-case analyzed scenario for the orbit. However, the worst-case conditions can often be uncommon or infrequent for some orbits, which results in static designs that overcompensate for reliability with excessive redundancy and inefficient use of resources. In such dynamic environments, an adaptive design can repurpose system resources in response to the environmental condition. An adaptive system can alternate between high-performance and high-dependability modes in response to the dynamic fault rate to maximize system performance subject to availability constraints throughout the mission.

In this chapter, we propose Hybrid, Adaptive, Reconfigurable Fault Tolerance (HARFT), a reconfigurable framework for environmentally adaptive resilience in hybrid and heterogeneous SoCs and systems for space applications. The HARFT architecture combines runtime-reconfigurable fault-tolerance modes for both subsystems (CPU and FPGA) of a hybrid SoC into one integrated, synergistic framework. CPU modes include symmetric multiprocessing (SMP) and asymmetric multiprocessing (AMP), and FPGA modes include PRMs (accelerators or soft cores) arranged for simplex, DWC, or TMR operation. Each mode has unique characteristics and tradeoffs in performance and reliability. The synergy of the system refers to the recovery and failover paths within and between subsystems in HARFT to enable re-

pair and graceful-degradation mechanisms that elongate system uptime. In response to the dynamic fault rate, HARFT adapts by selecting modes from the trade space that maximize system performability while satisfying availability constraints throughout the mission.

Furthermore, to evaluate HARFT, we propose an extended methodology for evaluating environmentally adaptive and gracefully degradable systems, using phased-mission modeling, subject to dynamic near-Earth radiation environments, using a combination of orbital-perturbation, geomagnetic-field, and CRÈME96 models. Using this methodology, we can formulate a trade space in performability and availability to allow users to select the optimal strategy, static or adaptive, that achieves the highest performability subject to a user-defined availability constraint. We evaluate HARFT on the Zynq-7000 for six orbital case studies from four orbital regimes and demonstrate substantial performability gains while satisfying availability constraints.

3.1 Related Work

Environmentally adaptive and resilient computing on FPGA-based systems has been explored in the literature [35, 29, 108, 120, 60]. Jacobs et al. [35] proposed Reconfigurable Fault Tolerance (RFT), a framework that includes an environmentally adaptive resilience architecture, a time-varying fault rate model, and a performability model using phased-mission models. RFT monitors radiation stimuli to estimate the environmental condition and uses PR to reconfigure PRMs in various redundancy schemes (e.g., simplex, DWC, and TMR) at runtime to efficiently accommodate the environmental condition. Furthermore, the RFT fault rate model provides a method for predicting the SEE rate of the near-Earth radiation environment, and phased-mission Markov modeling is used to calculate the system availability and performability of RFT subject to these dynamic fault rates. Glein et al. [29] proposed Adaptive SEE Mitigation (ASEEM), an environmentally adaptive architecture for spacecraft in geostationary orbit (GEO), which monitors radiation stimuli using a BRAM-based fault detector to estimate the solar condition and uses PR to alternate between simplex and TMR designs in the FPGA. Zhang et al. [120] proposed a method for estimating the

reliability of accelerators in response to the dynamic SEE rate. This estimate is then used to determine the duration at which accelerators satisfy the mission reliability constraint once reconfigured into the FPGA. When an accelerated function is requested, a runtime reliability manager uses a heuristic algorithm to select an accelerator variant from the trade space. Möstl et al. [60] proposed a self-aware resource manager to allow a system to adapt its applications in response to changing environmental conditions. The application reliability is estimated by analyzing the fault rates at multiple decomposition levels, and the dependability technique is selected based on the application reliability in response to the dynamic particle flux. Wilson et al. [108] proposed the prototype of HARFT, which extends RFT (limited to solely FPGAs) to hybrid SoCs, where both CPU and FPGA resources can be adapted in response to the environmental condition.

We extended HARFT of Wilson et al. [108] to create a more versatile architecture and modeling approach. Compared to previous research, we eliminate the assumption of a fault-free configuration manager by decentralizing this function across multiple control agents (e.g., hard cores and soft cores). Furthermore, we introduce synergistic mechanisms for recovery and failover paths within and between subsystems (CPU and FPGA) to enable repair and graceful degradation. To evaluate HARFT as an environmentally adaptive and gracefully degradable system, we leverage the RFT fault rate model for time-varying SEE rate predictions of the near-Earth radiation environment, and we extend the phased-mission modeling approach described in [35]. Our extensions include the following: our approach (1) removes the assumption of a fault-free control agent to account for the various control agents that are available depending upon the current mode of operation and (2) models the recovery and failover paths that differ between control agents.

3.2 HARFT Architecture Overview

The HARFT architecture is a hybrid framework providing environmentally adaptive resilience in commercial hybrid SoCs. Illustrated in Figure 9, the HARFT architecture consists of three subsystems: the *Hard Processing System* (HPS) framework, the *Soft Processing Sys-*

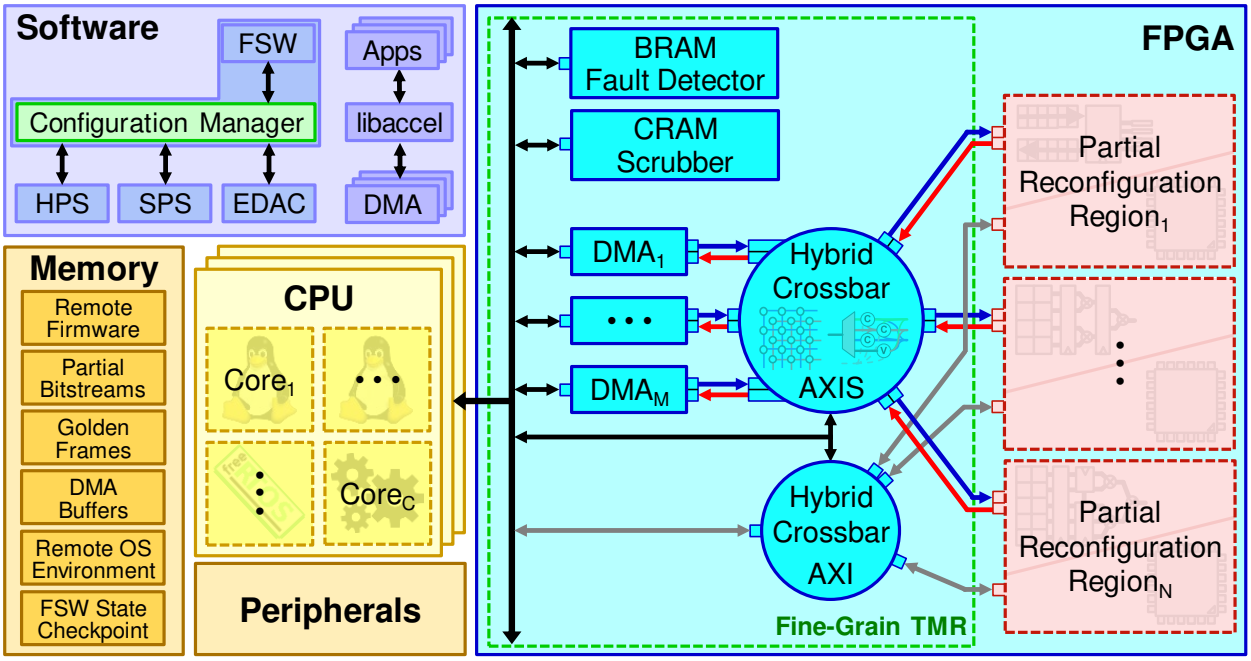


Figure 9: HARFT architecture.

tem (SPS) framework, and the *Configuration Manager* (CM). The HPS and SPS subsystems are low-level frameworks that enable versatility in the CPU and FPGA subsystems, respectively. The CM is part of the FSW and is responsible for reconfiguring the HPS and SPS in response to the environmental condition.

The HPS and SPS contain numerous control agents (e.g., hard and soft cores). These control agents can execute the FSW and CM, at varied proportions, to maintain system operation and configuration. The HARFT architecture includes failover mechanisms for graceful degradation to enable the transfer of control of the FSW across available control agents to extend system uptime as modules fail. In this section, an overview of the HPS, SPS, and CM are provided, including details on the experimental implementation for the Zynq-7000.

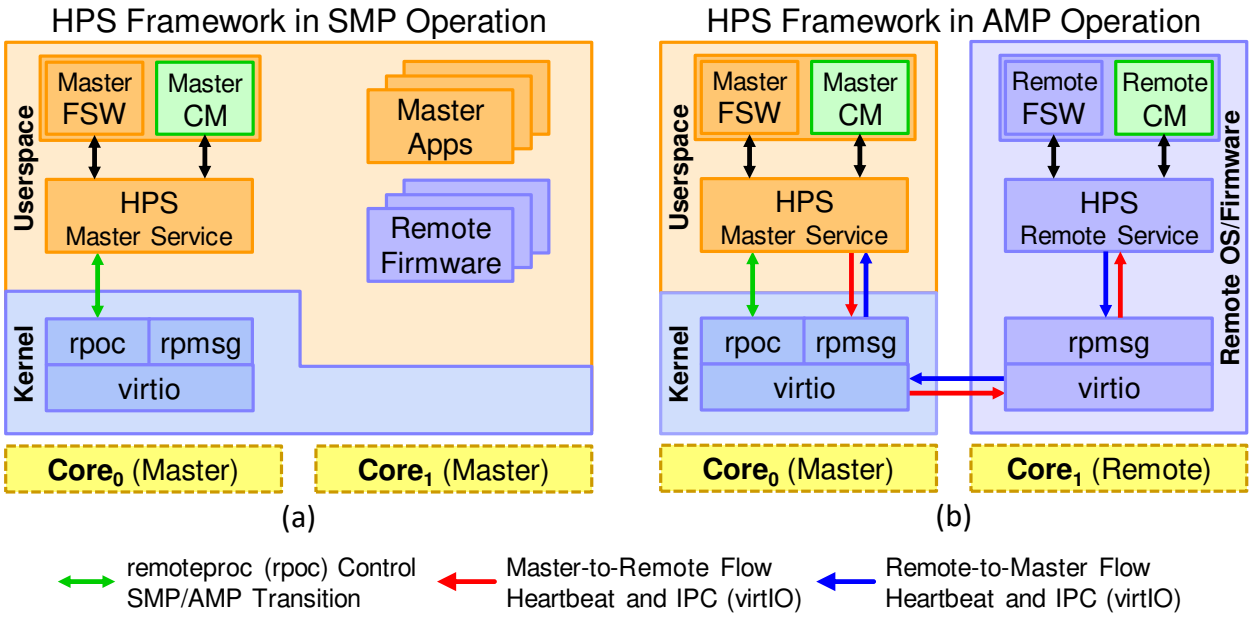


Figure 10: HARFT HPS framework for a dual-core CPU in (a) SMP and (b) AMP configurations.

3.2.1 Hard Processing System (HPS) Framework

The HPS framework, illustrated in Figure 10, resides on the CPU subsystem of the hybrid SoC. The HPS has two functions: (1) dynamically reconfiguring the CPU operation between various SMP and AMP configurations at runtime without interrupting system uptime (i.e., without a full system reset), and (2) detecting and managing failed OSs by recovery or failover. Although the HPS only addresses OS and core failures, it can be supplemented with user-defined dependability techniques (e.g., ABFT, TTMR, PLR) to protect core or specific applications. Depending upon the number of CPU resources available, the HPS partitions the CPU resources to create one *master OS* with zero or more *remote OSs*. The master OS runs on a partition of cores (*master cores*) running in SMP, and each remote OS runs on a separate partition of cores (*remote cores*). All OSs are control agents capable of executing the FSW, but the master OS assumes initial control on reset.

The HPS leverages OpenAMP and libmetal [114]. Libmetal provides an abstraction for the underlying OS (remote and master) with an API for device access, interrupt handling, and memory requests. OpenAMP is composed of three modules: virtIO (shared memory management for paravirtualization), remoteproc (life-cycle management), and RPMsg (API for inter-process communication (IPC) between isolated OSs). OpenAMP uses a master-remote paradigm for AMP, and the master OS manages all cores and launches or destroys remote OSs at runtime. Initially, the system begins with all cores running under the master OS. To launch a remote OS (SMP-to-AMP reconfiguration) at runtime, the master OS executes the following procedure. First, remoteproc decouples master cores from the master OS (i.e., the logical cores are disabled in Linux and are no longer scheduled to run processes), converting them into a partition of remote cores. Next, remoteproc creates an execution environment by loading the remote firmware to a preallocated memory space and assembles virtIO channels to enable IPC over RPMsg between the master and remote OSs. Finally, remoteproc signals the remote cores to begin executing the remote firmware, thus launching the remote OS. To destroy a remote OS (AMP-to-SMP reconfiguration) at runtime, the master OS executes the previously described procedure in reverse: remoteproc halts the remote OS and cores, disassembles the virtIO channels, destroys the execution environment of the remote cores, and consolidates the remote cores back into the master OS, converting them back into the partition of master cores.

The HPS provides recovery and failover mechanisms to detect and manage failed OSs. Self-checking duplex pairs are created between each master and remote OS by establishing dedicated virtIO channels between the isolated OSs to exchange heartbeat signals. An OS can detect the failure of another OS by lack of heartbeat signals received within an acceptable, user-defined period. Since the HPS inherits a master-remote paradigm from OpenAMP and libmetal, there are limitations to managing failed OSs depending upon the control agent. The master OS can recover a failed remote OS by executing the SMP-to-AMP reconfiguration process to recreate the execution environment, reload the remote firmware, and relaunch the remote OS. However, a remote OS is unable to recover the failed master OS. Instead, a failover occurs, and the remote OS quickly assumes control of the FSW with minimal system interruption but with degraded performance and capability. To further minimize

system interruption, the master OS periodically writes checkpoints of the FSW state to shared memory to allow other control agents to rollback and resume at a more recent state during failover. When the mission enters an operational safe state (i.e., noncritical phase), a self-reset is issued to restore the HPS and recover the master OS.

The extent of redundancy provided by the HPS is versatile and depends on the architecture and resources available in the CPU. The Zynq-7000 dual-core CPU is capable of one SMP dual-core (simplex) and two AMP single-core (duplex) structures. The Zynq-MPSoC quad-core CPU is capable of several possible operating modes, from one SMP quad-core (simplex) to four AMP single-core (quadruplex) structures. The Zynq-MPSoC is also a multiprocessor, which enables AMP operation across multiple processors. A high degree of redundancy enables multiple failover paths that can extend system uptime at degraded performance and capability.

3.2.2 Soft Processing System (SPS) Framework

The SPS framework resides on the FPGA subsystem of the hybrid SoC. The SPS includes DMAs, interconnects, and hybrid crossbars that reside in the SR and are selectively protected by FG-TMR using the BL-TMR tool. The SPS also includes the PRRs that can be configured with soft-core (e.g., MicroBlaze, RISC-V, and LEON3) or accelerator PRMs. The SPS is capable of dynamically reconfiguring the FPGA operation between various redundancy configurations at runtime without interrupting system uptime. Soft core processors can serve as control agents capable of executing the FSW, but control is initially assumed by a control agent in the HPS on reset.

The SPS leverages PR to reconfigure PRRs in the FPGA subsystem and operates the hybrid crossbars to configure the redundancy scheme. The PRRs expose three virtual interfaces: one AXI port (for soft cores to access memory) and two AXI4-Stream (AXIS) ports (for input and output data streams through stream-based accelerators). For hardware acceleration, the AXIS hybrid crossbar is used to connect DMAs to PRMs. The AXIS hybrid crossbar can connect one DMA to two or three PRRs containing the same PRM in lockstep for coarse-grain DWC or TMR operation, respectively. When configured for redundancy, the

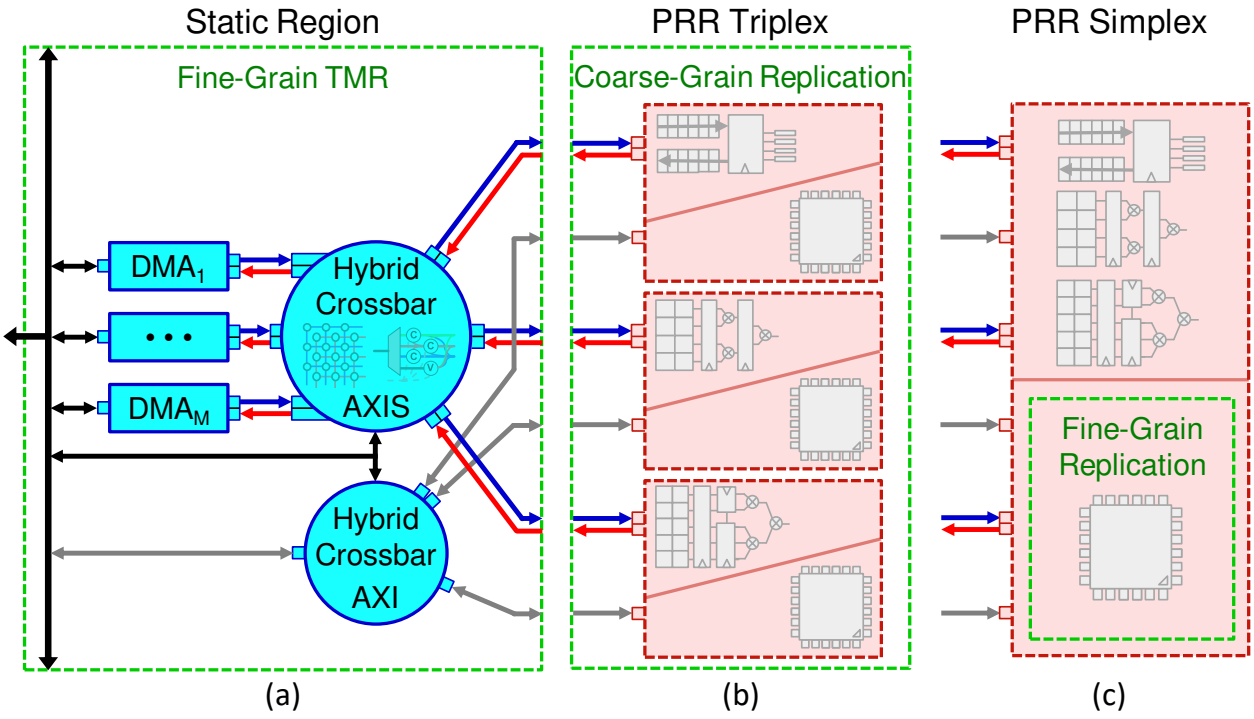


Figure 11: HARFT SPS framework with (a) static logic protected by FG-TMR, with reconfigurable regions in two configurations: (b) PRR triplex with PRMs operating in lockstep for coarse-grain redundancy and (c) PRR simplex with fine-grain replication implemented within the PRR.

input data stream is replicated to all connected PRMs, and the output data stream of each replica runs through a comparator or majority voter for DWC or TMR, respectively. Since all replicas run in lockstep, module-based errors are captured, signaled to the connected DMA, and recovery is serviced by the application using that DMA. The DMA has built-in decoupling mechanisms to safely perform PR and to inhibit the propagation of faults from faulty PRMs to the static logic. When PRMs are unable to complete the interconnect transaction (e.g., hang), the decoupling mechanism is invoked, and the transaction is terminated to protect the interconnect and static logic. Similarly, the AXI hybrid crossbar will bypass, compare, or vote on the transactions over AXI depending upon the number of redundant soft cores.

Depending upon the PRM, some form of resynchronization is required to restore lockstep operation following MER. For PRMs that do not maintain state or residue after use, PR with an optional reset can be sufficient for resynchronization. Accelerators, which do not serve as control agents, are recovered and resynchronized in between use by performing PR with an optional reset. However, soft cores, which serve as control agents, require additional steps to minimize system downtime during self-repair, especially when the soft cores are the active control agent (i.e., the HPS has failed). In TMR operation, the soft cores can identify the faulty replica via the status of the AXI hybrid crossbar. Next, the soft cores write a checkpoint of the FSW state to shared memory and issue a PR request to reconfigure the faulty replica. The soft cores signal the AXI hybrid crossbar to lock AXI (i.e., the interconnect is stalled until a transaction is received from all replicas) and issue a self-reset. All soft cores, now in a reset state, initialize and issue their first transactions to the AXI hybrid crossbar. Once the AXI hybrid crossbar receives the first transaction from all replicas, the soft cores are now resynchronized, and AXI is unlocked. The soft cores proceed to read and validate the FSW state checkpoint for rollback. In DWC operation, self-repair is also possible provided that the faulty replica is quiet (i.e., it does not issue unexpected transactions over AXI).

Due to the reconfigurability of FPGAs, the number and size of the PRRs are highly configurable and depend upon the architectural layout and resources available in the FPGA. Furthermore, there are two approaches for using the SPS, as illustrated in Figure 11. The

first approach involves instantiating PRRs in triplexes. In this approach, the PRRs can run decoupled PRMs in parallel for performance (e.g., three separate accelerators in parallel) or synchronized PRMs in lockstep for reliability (e.g., TMR soft cores) using the hybrid crossbars for coarse-grain redundancy. The second approach involves instantiating PRRs in simplexes, with mitigation techniques implemented within the PRRs. In this approach, the PRRs can run unmitigated, high-area PRMs for performance (e.g., unmitigated, large accelerators) or mitigated, low-area PRMs for reliability (e.g., mitigated, small accelerators).

The SPS also provides failover mechanisms to create redundancy structures between HPS and SPS control agents. Since the CPU-FPGA system also inherits a master-remote paradigm from the Zynq-7000 architecture, the capabilities for system recovery vary between HPS and SPS control agents. HPS control agents can recover failures in the SPS via FR or PR. However, SPS control agents are not able to recover failed HPS control agents. If the HPS fails, an SPS control agent can assume control of the FSW with degraded performance and capability. Having numerous soft cores instantiated can improve system availability; however, there is a potential performance tradeoff because FPGA resources are allocated for redundancy instead of acceleration. In the Zynq-MPSoC architecture, the control agents between the HPS and SPS can form self-checking duplex pairs because the HPS can be reset while the SPS remains active and vice versa.

3.2.3 Configuration Manager (CM)

The CM is responsible for configuring the HPS and SPS in response to the dynamic radiation environment. To maintain control of system recovery and reconfigurability, the CM is part of the FSW and is executable on all control agents (i.e., master OS, remote OSs, and soft cores). However, the extent of management and control available by the CM is proportional to the capabilities provided by the control agent.

3.2.3.1 Environmental Sensing and Prediction

To assess the environmental condition, the CM can monitor radiation stimuli (via on-board SEU-detection circuitry or external radiation-flux sensors or dosimeters) or use model-based predictions (discussed in Section 3.3). The CM monitors SEUs in various memory subsystems of the hybrid SoC. In the CPU, the processor caches, on-chip memory, and, optionally, external memory are monitored using the EDAC module. When SEUs occur, the events are captured by ECC and parity mechanisms built into the CPU architecture, which are polled and counted by the EDAC module. In the FPGA, CRAM, BRAM, and LUTRAM are monitored. Static CRAM monitoring is enabled by a hybrid CRAM scrubber. The hybrid scrubber includes a readback first-stage scrubber (FSS) and replacement second-stage scrubber (SSS). This FSS is an asynchronous controller residing in the FPGA. This controller interfaces to the ICAP, for CRAM frame readback and writeback operations, and the FRAME_ECC module, for calculating the ECC syndrome and CRC checksum of a CRAM frame during a readback operation. The FRAME_ECC can locate all single-bit faults and detect most multibit faults. The FSS periodically scans the entire device CRAM frame-by-frame. The FSS issues a frame readback command to the ICAP to retrieve and store the frame contents into a buffer, and it reads the outputs calculated by the FRAME_ECC of the corresponding frame. If the frame is faulty and correctable, the ECC syndrome is used to decode the word and bit location of the erroneous bit. A bit flip is performed in the corresponding bit location of the buffered frame, and the FSS issues a frame writeback command to the ICAP to overwrite the CRAM frame with the corrected contents. In the uncommon case where the faulty frame is uncorrectable by the FSS (e.g., a multibit upset that overcomes the FRAME_ECC EDAC mechanisms), the FSS halts and invokes the SSS via an interrupt. The SSS is software executed on the CPU that uses the frame address of the faulty frame to index golden frames stored in protected memory. The SSS issues a frame writeback command to the PCAP to replace the faulty frame. The BRAM fault detector is a counter that monitors application-specific memory structures, such as BRAM or LUTRAM equipped with ECC. The counter increments when signaled by an ECC controller on fault detection.

The CM monitors and records onboard SEUs over time. Using an aging-window approach, the CM can estimate the SEE rate for a segment of the orbit. For some missions, SEEs occur too infrequently to be reliably used as stimuli, such as orbits with low exposure to radiation or computers with a low probability of SEUs (e.g., few or small hybrid SoCs). In these cases, several orbits of SEU monitoring are necessary to estimate and predict the dynamics of the radiation environment for that orbit. Alternatively, dedicated radiation-flux sensors or model-based prediction approaches can be used.

3.2.3.2 Reconfiguration and Adaptation

The HPS and SPS are highly versatile and can be configured to one of several operating modes, each with its own set of tradeoffs. In Section 3.4, we demonstrate a model-based approach for determining the Pareto-optimal set of HARFT strategies of the availability and performability trade space. Using a threshold-based approach, the environmental conditions can be segmented into orbital segments bounded by SEE rate thresholds. For each orbital segment, the CM selects the operating mode that maximizes system performance subject to a user-defined availability constraint for that corresponding environmental condition. To adapt CPU and FPGA resources, the CM controls the HPS and SPS, respectively. To reconfigure the HPS, the CM invokes the remoteproc module to launch or destroy remote OSs. To reconfigure the SPS, the CM reconfigures the PRRs and operates the hybrid crossbars to configure the redundancy scheme.

3.2.3.3 Control-Flow Model

The HARFT architecture has numerous failover mechanisms for graceful degradation. These failover mechanisms are represented by a control-flow model that represents the transfer of control (edges) between control agents (nodes) as subsystems change or modules fail. At least one control agent must be available for the system to be operational. The control-flow model of the HARFT architecture for the Z7020 is illustrated in Figure 12. On reset, the master OS assumes initial control of the FSW. The master OS has the full capability of the CM and can reconfigure or recover modules in the HPS and SPS. If the master OS fails,

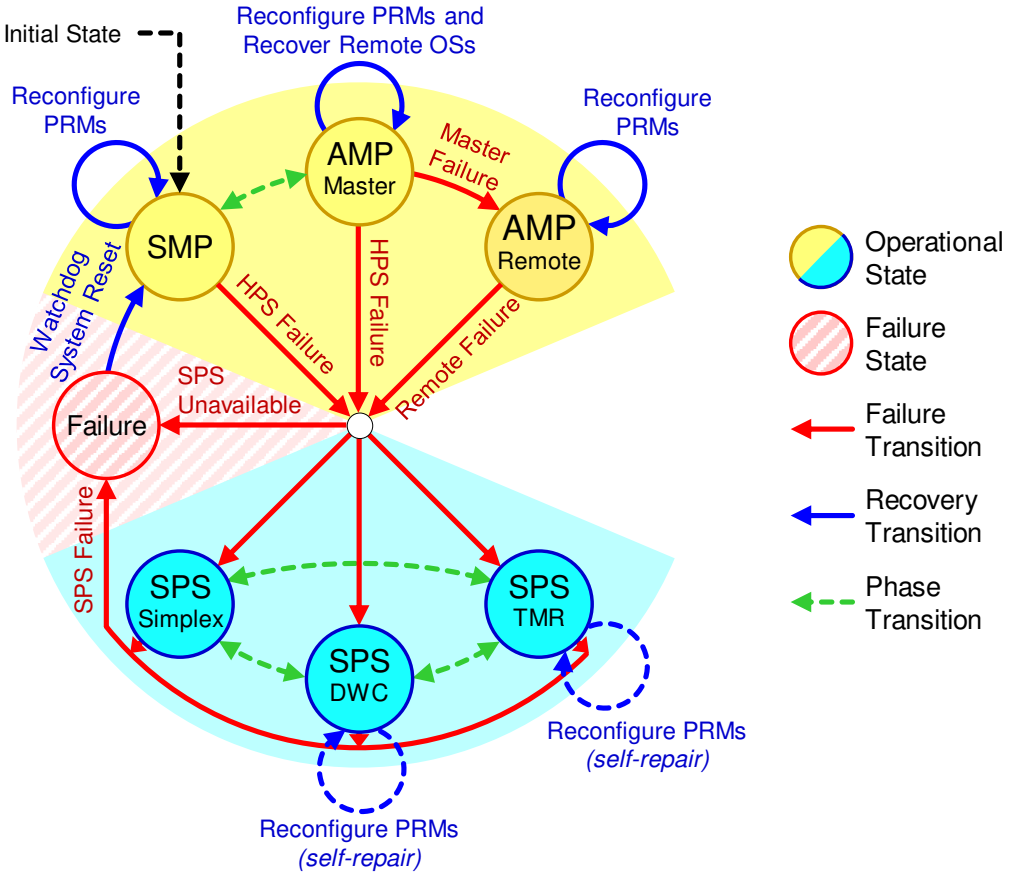


Figure 12: HARFT control-flow model on the Zynq-7000 (dual-core).

a remote OS, if available, assumes control at a degraded performance (nonfunctional cores) and capability (cannot reconfigure HPS). If the HPS (master OS and all remote OSs) fails, the soft cores, if available, can assume control at a further degraded performance (nonfunctional CPU) and capability (cannot reconfigure HPS; limited reconfiguration of SPS). The SPS control agents can perform self-repair (e.g., self-reconfiguration of one faulty PRM in the coarse-grain TMR or DWC configurations). Control agents can issue a self-reset when the mission enters an operational safe state to recover the full system.

3.3 Modeling Approach

To analyze the dependability of HARFT, a methodology is required for modeling adaptive and gracefully degradable systems subject to dynamic fault rates. To address this problem, our methodology is composed of two steps: (1) modeling the dynamic environment to determine the module fault rates over time and (2) modeling the adaptive and gracefully degradable system for reliability analysis. To account for the dynamic radiation environment, three well-established models (orbital perturbation, geomagnetic field, and CRÈME96) are combined to predict the time-varying SEE rate, which can be scaled to approximate the fault rates for modules of the HARFT architecture. To model the hybrid, adaptive system, a phased-mission system modeling approach is used to describe the reliability characteristics of the HARFT architecture. Combined, this methodology is used to analyze the availability, failure rate, and performability of various static and adaptive strategies for HARFT subject to user-defined near-Earth orbits.

3.3.1 Modeling the Dynamic Near-Earth Radiation Environment

To model the fluctuating, radiation-induced fault rate, an approach is needed for modeling the device susceptibility to the dynamic radiation environment of user-defined orbits. Jacobs et al. [35] proposed the combination of three models (orbital perturbation, geomagnetic field, and CRÈME96) to predict the time-varying SEE rate for a given system and

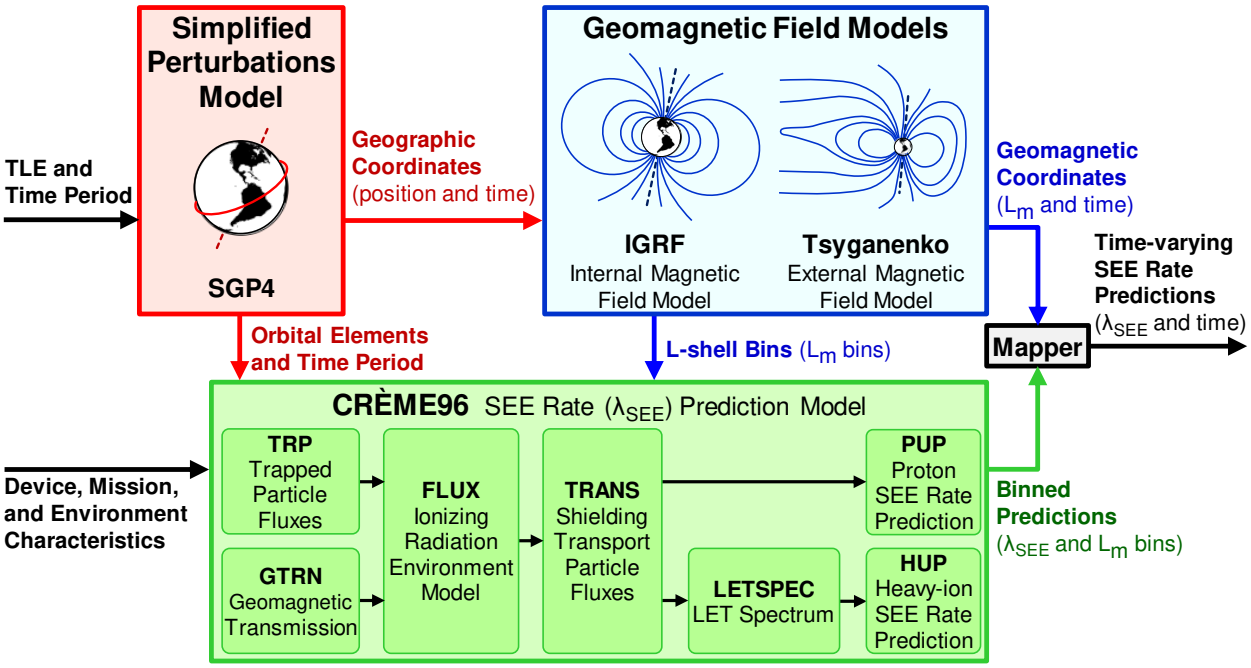


Figure 13: Methodology for time-varying SEE rate prediction.

orbit. This approach focuses on the dynamic radiation environment of near-Earth orbits, which includes the geomagnetic field with influence from the solar weather. Drift shells, labeled by L_m, are used to estimate the fluxes of trapped particles within the geomagnetic field and GCRs attenuated after geomagnetic shielding. Depending upon the orbit or trajectory, a spacecraft can traverse several drift shells, each with its own radiation characteristics (species, energies, and fluxes). The complete methodology is illustrated in Figure 13.

The Simplified General Perturbation (SGP4) model is used to predict the orbital position (in geographic coordinates) of near-Earth objects and debris for a specified period [33]. The North American Aerospace Defense Command tracks and assigns space objects with a two-line element (TLE) that provides designator information and a set of orbital elements (e.g., inclination, eccentricity, perigee) measured at some epoch time. SGP4 uses these orbital elements to predict the orbital position of the object using orbital perturbation and propagation. For specified TLE and period inputs, SGP4 predicts the orbital position over time.

The International Geomagnetic Reference Field (IGRF) and Tsyganenko geomagnetic-field models are used to convert the geographic coordinates (orbital position) into the geomagnetic-field coordinates of the McIlwain coordinate system. IGRF models Earth's main magnetic field [97, 103], and Tsyganenko models Earth's external magnetic field, which includes the geomagnetic-field influence from solar winds and interplanetary currents [100]. Both models are included in the open-source International Radiation Belt Models (IRBEM) library [68]. For specified geographic position and time, both geomagnetic-field models predict the L_m over time.

CRÈME96 is the state-of-the-art tool for SEE rate prediction based on phenomenological models, developed by Vanderbilt University and supported by NASA [101]. CRÈME96 uses orbital, device, mission, and environmental characteristics to predict the SEE rate induced by protons and heavy ions. The orbital characteristics specify the orbital elements (e.g., perigee, apogee, inclination) of the spacecraft. Device characteristics specify the SEE susceptibility of a device to protons and heavy ions. SEE susceptibility is experimentally measured by radiation-beam testing and the characterization is often specified as Weibull parameters that fit the experimental data. The SEE characterizations for numerous resource types (e.g., FPGA primitives like CRAM, BRAM, and DSPs) in Xilinx 7-Series, UltraScale, and UltraScale+ devices under heavy-ion irradiation are available [50, 48, 49]. Extrapolation methods are available to approximate the SEE susceptibility to protons using heavy-ion characterizations [72, 22, 23]. Mission characteristics include mission duration (number of orbits) and spacecraft shielding (material and thickness) parameters. Finally, environmental characteristics specify the solar condition, space-radiation models, and radiation species to consider. CRÈME96 can also predict the average SEE rates for a specific orbital segment bounded by lower and upper L_m (between two drift shells), which form an L_m bin. Multiple L_m bounds can be specified in CRÈME96 to predict the SEE rates for several L_m bins. Finally, the SEE rates are assigned to the time domain by mapping the corresponding L_m to the L_m bin. The accuracy of the SEE rate predictions depends upon the accuracy of the models used in our methodology.

To approximate the fault rate for a given design and orbit, the following procedures are performed. First, the time-varying SEE rate prediction method is used to predict the SEE rates for each resource type. For each resource type $r \in R$, this method will produce two SEE rates: one heavy-ion-induced $\lambda_{\mathrm{SEE}_{r,\mathrm{HI}}}(t)$ and the other proton-induced $\lambda_{\mathrm{SEE}_{r,\mathrm{p}^+}}(t)$. Both SEE rates are added to produce the combined SEE rate $\lambda_{r,\mathrm{SEE}}(t)$ for that resource type.

$$\lambda_{r,\mathrm{SEE}}(t) = \lambda_{r,\mathrm{SEE}_{\mathrm{HI}}}(t) + \lambda_{r,\mathrm{SEE}_{\mathrm{p}^+}}(t) \tag{3.1}$$

Next, the resource utilization of the design must be determined. For FPGA designs, the design tools can be used to obtain the FPGA resource utilization of the design, and for software/CPU designs, the embedded system configuration or software can be analyzed to estimate the memory footprint and CPU resource utilization. For each resource type, the resource utilization RU_r of the design is used to scale the corresponding SEE rates predicted for that resource. The scaled SEE rates are combined across all resource types to approximate the design fault rate.

$$\lambda_{\mathrm{design}}(t) = \sum_{r \in R} \lambda_{r,\mathrm{SEE}}(t) \cdot \mathrm{RU}_r \tag{3.2}$$

Furthermore, the AVF can be used to improve the accuracy of the fault rate approximation. AVF refers to the probability that a fault in the system will manifest into failure [61] and can be determined by fault injection or radiation-beam testing. Fault injection is the practice of injecting faults into the design to observe the architectural response and propagation of faults. With fault injection, the AVF of a design is calculated using Equation (2.1):

$$\text{AVF} = \frac{\text{Number of Observable Events}}{\text{Number of Fault Injections}} \tag{3.3}$$

In this context, the AVF describes the fraction of the resource utilization that is critical to faults. If the AVF is infeasible to measure (e.g., resources that are not accessible for fault injection), then the worst-case (AVF = 1) or an alternative resource AVF is assumed. The final approximation of the time-varying design fault rate is calculated using Equation (2.4):

$$\lambda_{\text{design}}(t) = \sum_{r \in R} \lambda_{r,\text{SEE}}(t) \cdot \text{RU}_r \cdot \text{AVF}_r \tag{3.4}$$

3.3.2 Modeling the Adaptive and Gracefully Degradable System

This section describes the modeling approach and analysis methods for evaluating adaptive and gracefully degradable systems using phased-mission system models. This approach can accurately model the dynamic recovery and failover paths to represent system adaptations in response to changes in the environmental condition and graceful degradation mechanisms as control agents fail.

3.3.2.1 Markov Modeling and Performability

Continuous-time Markov chains (CTMCs) are often used to model systems for dependability analysis [75]. A CTMC model is composed of *states* and *transition rates*. A state represents a feasible operating mode of the system. A state is an *available state* if the system is available at that state; otherwise, it is an *unavailable state*. Transition rates represent the fault and repair mechanisms of the system and are weighted by constant fault rates (denoted as λ) and repair rates (denoted as μ), respectively. The average availability of the

system is calculated as the average probability that the system is in an available state. The instantaneous availability is the probability that the system is in an available state at time t.

CTMC reward models are weighted CTMCs that include weights at each available state to model the reward rate for the system being in that state, enabling performability to be calculated. Performability is a metric that combines system availability and performance, which is useful for evaluating adaptive and gracefully degradable systems. As with availability, the average performability of a system is calculated as the average probability that the system is in an available state at time t times the reward rate of that state.

A transient analysis of a CTMC is useful for determining the transient state occupancy probabilities p_t at time t, given initial state occupancy probabilities p_0 at initial time t_0. CTMCs can be solved using symbolic or numerical methods to calculate the transient probabilities. The instantaneous availability is equal to the sum of transient probabilities of available states. Assuming CTMC $x(t)$ with state space S (available state subspace S_A and unavailable state subspace S_U), the instantaneous availability and performability at time t are defined using Equations (3.5) and (3.6):

$$\text{Availability}(t) = \sum_{a \in S_A} \text{Probability}\{x(t) = a\}, \tag{3.5}$$

$$\text{Performability}(t) = \sum_{a \in S} \text{Performance}(a) \cdot \text{Probability}\{x(t) = a\}. \tag{3.6}$$

The instantaneous failure rate, which is the rate at which the system enters an unavailable state, can also be determined using transient analysis. To calculate this quantity, the CTMC is modified to include an immediate repair transition rate ($\mu = \infty$) from any unavailable state to the initial state. When analyzed, the instantaneous failure rate of the CTMC

is the total flow rate from available states to unavailable states. This flow rate is equal to the sum of the fault transition rates from available states to unavailable states times the transient probability of the corresponding available states.

$$\text{Failure Rate}(t) = \sum_{a \in S_A} \sum_{u \in S_U} \text{Transition Rate}_{a \to u}(t) \cdot \text{Probability}\{x(t) = a\} \tag{3.7}$$

The state space of a CTMC can also be divided into subspaces, where states in one subspace share a common property. Using transient analysis, the instantaneous probability that the system is in the state subspace S_G is equal to the sum of the transient probabilities of all states in that subspace.

$$\text{Probability}_G(t) = \sum_{s \in S_G} \text{Probability}\{x(t) = s\} \tag{3.8}$$

3.3.2.2 Phased-Mission System Modeling

To model adaptive systems that change at different phases throughout the mission, a phased-mission system approach is used. A CTMC-based phased-mission system is represented as a sequence of phases, with one CTMC model for each phase and *phase transitions* modeling the transitions from states of one phase to states of the subsequent phase [4, 102]. Phase transitions can be probabilistic (weighted) or deterministic (unweighted).

For CTMC-based phased-mission systems, the transient analysis is executed sequentially in discrete timestep intervals. For each timestep, the CTMC representing the current phase of the system is determined, and the corresponding initial probabilities, fault rates, and repair rates are specified. The CTMC is then analyzed over the timestep interval, with the fault rates and repair rates assumed to be constant. Once analyzed, the transient probabilities are

calculated, which are used to determine the initial probabilities for the next timestep. Using the transient probabilities, the instantaneous availability, failure rate, and performability at time t are calculated using Equations (3.5), (3.6), and (3.7), respectively.

At each timestep, the system may or may not undergo a phase change. If a phase change occurs, the CTMC of the subsequent phase is specified, and the phase transitions between the current and subsequent phases are used to determine the initial probabilities for the next phase. The initial probability of each state in the subsequent phase is equal to the weighted (probabilistic) or unweighted (deterministic) sum of transient probabilities of states in the current phase with phase transitions to that state. Alternatively, if a phase change does not occur, then the current CTMC is reused with fault rates updated, and the initial probabilities of the next timestep are equal to the corresponding transient probabilities of the current timestep.

Figure 14 illustrates a simplified example of the phase-mission model for the HPS and SPS. The HPS model in Figure 14(a) has SMP (CTMC_{SMP}) and AMP (CTMC_{AMP}) modes as phases. CTMC_{SMP} contains two *normal states* $\{\text{MM}_{\text{SMP}}, \text{F}_{\text{SMP}}\}$, and CTMC_{AMP} contains four $\{\text{MR}_{\text{AMP}}, \text{M}_{\text{AMP}}, \text{R}_{\text{AMP}}, \text{F}_{\text{AMP}}\}$. The normal states are sufficient to model both modes independently. For example, in SMP, the HPS (master OS) fails if either core fails ($\text{MM}_{\text{SMP}} \rightarrow \text{F}_{\text{SMP}}$; a fault transition), and the HPS is recovered by a board-level watchdog reset ($\text{F}_{\text{SMP}} \rightarrow \text{MM}_{\text{SMP}}$; a periodic repair transition). In AMP, the HPS (master-remote pair) can encounter a failure of the remote core ($\text{MR}_{\text{AMP}} \rightarrow \text{M}_{\text{AMP}}$) or master core ($\text{MR}_{\text{AMP}} \rightarrow \text{R}_{\text{AMP}}$). The master OS can recover the remote core ($\text{M}_{\text{AMP}} \rightarrow \text{MR}_{\text{AMP}}$), but the HPS fails if the master core fails prior to remote recovery ($\text{M}_{\text{AMP}} \rightarrow \text{F}_{\text{AMP}}$). The remote OS cannot recover the master core, and the HPS fails if the remote core fails ($\text{R}_{\text{AMP}} \rightarrow \text{F}_{\text{AMP}}$).

Because phase transitions are instantaneous, additional states are introduced to model and account for (1) event-driven adaptation time and (2) failover. To demonstrate (1), suppose that the HPS is in state MM_{SMP} and an SMP-to-AMP reconfiguration (a phase change) occurs. Because phase changes are instantaneous, they do not model the temporary downtime due to system adaptation during a phase change. Therefore, CTMC_{AMP} introduces the state M_{AMP}, which forms a phase transition with MM_{SMP}. During the phase change, the $\text{MM}_{\text{SMP}} \rightarrow \text{M}_{\text{AMP}}$ phase transition occurs instantly followed by the HPS launching the

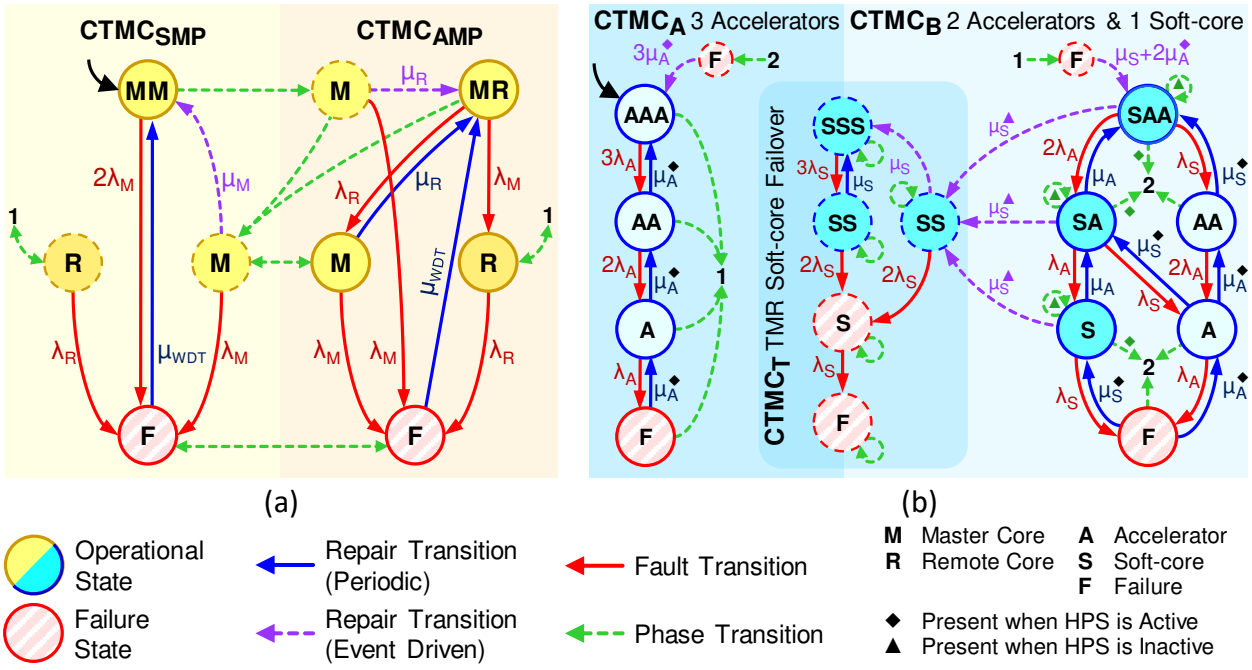


Figure 14: Example of phased-mission modeling with CTMCs for (a) HPS modes and (b) SPS modes.

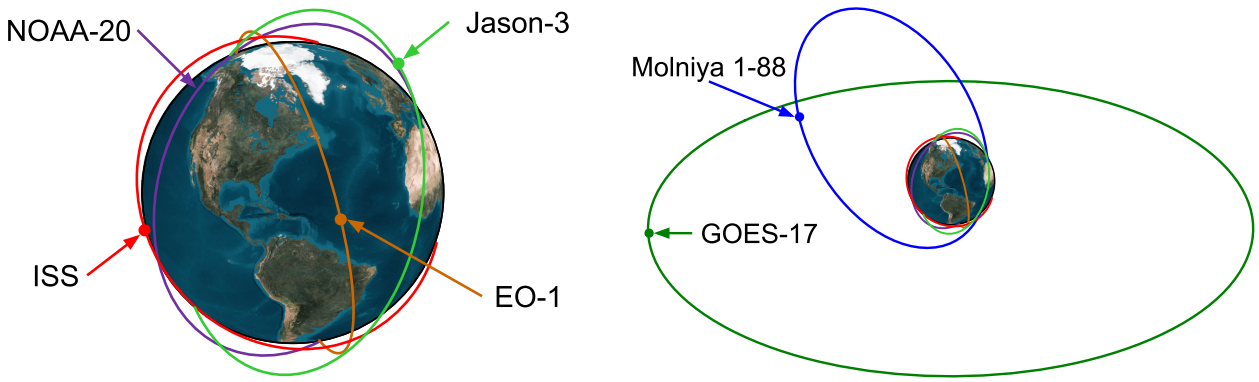
remote OS ($M_{AMP} \rightarrow MR_{AMP}$; an event-driven repair transition). Similarly, this behavior is modeled when the HPS is in state MR_{AMP} and an AMP-to-SMP reconfiguration occurs, using the additional state M_{SMP}. To demonstrate (2), suppose that the HPS is in state R_{AMP} and an AMP-to-SMP reconfiguration occurs. Because $CTMC_{SMP}$ does not have a state with the remote OS in control, there is no logical phase transition from R_{AMP} to any state in $CTMC_{SMP}$. Therefore, $CTMC_{SMP}$ introduces the state R_{SMP}, which forms a phase transition with R_{AMP} so that the remote OS remains in control after the phase change.

The SPS model in Figure 14(b) has two modes as phases: three parallel accelerators ($CTMC_A$) and two parallel accelerators and one soft core ($CTMC_B$), both with failover to TMR soft cores ($CTMC_T$) on HPS failure. $CTMC_A$ and $CTMC_B$ model the PRMs as independent, simplex modules. The SPS model is nested within each state of the HPS model, and depending upon the operational state of the HPS, there can be differences in the (1) repair and phase transitions and (2) reward rates. To demonstrate (1), consider the two following scenarios. When the HPS is operational, it uses PR to change the SPS mode. This can be modeled as a phase transition from the states of $CTMC_A$ to the failure state (no operational PRMs) in $CTMC_B$ followed by an event-driven PR to repair the PRMs of $CTMC_B$. This approach accounts for the temporary downtime due to system adaptation, but this example is simplistic because both modes have common PRMs that do not need to be reconfigured during a phase change. When the HPS fails, the soft core of $Mode_B$ can perform a failover procedure to self-reconfigure for TMR operation. This can be modeled as an event-driven repair transition from any soft-core-controlled state of $CTMC_B$ to state SS_T followed by another transition to enter TMR operation ($SS_T \rightarrow SSS_T$). To demonstrate (2), consider the scenarios when the SPS is in a state containing accelerators (e.g., AAA_A or SAA_B). When the HPS or soft core in the same state is operational, the system receives a performance reward for these states because there is an active control agent able to use the accelerators. However, if the HPS fails and there is no operational soft core, these states become failure states with no performance reward.

3.4 Evaluation and Analysis

In this section, we use the modeling methodology described in Section 3.3 to evaluate the HARFT architecture and to analyze the architectural tradeoffs between both static and adaptive strategies in terms of availability, failure rate, and performability. The HARFT architecture is implemented on the Zynq-7000 Z7020 platform and is subjected to the environmental conditions of six different near-Earth orbits from four orbital regimes: low-Earth orbit (LEO), sun-synchronous orbit (SSO), highly elliptical orbit (HEO), and geostationary orbit (GEO). The spacecraft in these orbits serve as case studies to compare the application of HARFT for various environmental conditions.

Section 3.4.1 describes the orbital case studies, evaluates the time-varying SEE rate prediction model, and analyzes the predicted SEE rates. Section 3.4.2 describes the HARFT implementation and the acquisition of experimental quantities, including performance, resource utilization, AVF, and repair rates. To quantify the performance potential of HARFT modules, the CoreMark benchmark [73], a synthetic benchmark for embedded CPUs, and two-dimensional bilateral-filtering application kernel are used. The bilateral filter is an edge-preserving kernel for reducing noise in images and is used in numerous space applications, such as image fusion for multisensor images, anomaly detection for hyperspectral images, and speckle-noise reduction for synthetic-aperture radar. To determine the fault rates of the HARFT modules, the predicted SEE rates are scaled using the resource utilization and AVF approximated by CRAM fault injection. Finally, the repair rates of several recovery mechanisms provided by HARFT are measured. The performance, fault rates, and repair rates are inputs to the phased-mission system models. Finally, Section 3.4.3 describes the static and adaptive strategies for HARFT and evaluates the transient analysis of the CTMC-based phased-mission system models. The availability, failure rate, and performability are calculated to compare between strategies and to demonstrate the advantages of adaptive resilient computing.



(a) LEO/SSO regimes (b) HEO/GEO regimes

Figure 15: Orbital case studies with spacecraft in the (a) LEO/SSO and (b) HEO/GEO regimes. Rendered using the AGI Systems Tool Kit (STK) 11 software.

Table 1: Mission parameters of orbital case studies.

Spacecraft	Identifier (SATCAT)	Orbital Regime	Perigee (km)	Apogee (km)	Inclination (°)	Period (min)
ISS	25544U	LEO	404.04	407.68	51.64	92.69
Jason-3	41240U	LEO	1331.76	1343.71	66.04	112.39
EO-1	26619U	SSO	671.04	686.47	97.81	98.27
NOAA-20	43013U	SSO	825.91	827.94	98.75	101.38
Molniya 1-88	23420U	HEO	1073.71	23942.45	63.61	430.47
GOES-17	43226U	GEO	35774.55	35800.00	0.06	1436.18

Orbital parameters calculated using SGP4 for TLEs with an epoch of January 1, 2019 UTC.

3.4.1 Orbital Case Studies

To provide a comprehensive assessment of the HARFT architecture, six different orbital case studies from four orbital regimes (LEO, SSO, HEO, and GEO) are investigated. The orbital case studies include the ISS (LEO), Jason-3 (LEO), EO-1 (SSO), NOAA-20 (SSO), Molniya 1-88 (HEO; Molniya orbit), and GOES-17 (GEO) spacecraft. For each spacecraft, the orbital parameters are shown in Table 1 and the orbital paths are illustrated in Figure 15. The ISS and Jason-3 reside in LEO and experience relatively high radiation fluxes when traversing near Earth's poles or through the South Atlantic Anomaly (SAA), where the inner Van Allen radiation belt is closest to Earth's surface, at an altitude as low as 200 km [32]. Jason-3 orbits at a higher altitude and inclination than the ISS, exposing it to higher fluxes of high-energy trapped protons in the inner Van Allen radiation belts and GCRs and SPEs due to reduced geomagnetic shielding. The EO-1 and NOAA-20 spacecraft reside in SSO, which is a nearly polar orbit with an inclination of approximately 98°. In SSO, the orbit has a precession period of one year, which allows spacecraft to maintain a synchronized orbital relationship with the Sun. As with LEO, spacecraft in SSO are exposed to relatively higher radiation fluxes near Earth's poles or at the SAA. The Molniya 1-88 spacecraft resides in HEO. The HEO regime refers to orbits with high eccentricity. A special class of HEOs, called Molniya orbits, has an inclination of approximately 63.4° to provide a wide viewing angle over high-latitude regions. Depending upon the orbital apogee and perigee, spacecraft in HEO often traverse several drift shells. During the apogee pass, which accounts for most of the orbit, spacecraft are exposed to GCRs, SPEs, and trapped particles in the outer Van Allen radiation belts, and during the perigee pass, spacecraft are exposed to trapped protons in the inner Van Allen radiation belt. The GOES-17 spacecraft resides in GEO. Spacecraft in GEO are synchronized with Earth's rotation with minimal inclination to orbit over the equator. GEO is useful for space applications that require a constant presence over a specific area on Earth. Spacecraft in GEO have low susceptibility to trapped protons, which are present at very low energy levels at this altitude. However, due to reduced geomagnetic shielding, these spacecraft are highly susceptible to GCRs and SPEs, and the radiation environment is heavily influenced by the solar weather condition. Spacecraft in these orbits can benefit

from high-performance onboard processing to compress massive volumes of raw sensor data for efficient downlink or spacecraft autonomy. However, the magnitude and variation of radiation fluxes each spacecraft is exposed to are highly characteristic of its orbit. As a result, the optimal static or adaptive strategies will depend significantly upon the orbit.

The SEE rate prediction methodology described in Section 3.3.1 is performed to determine the time-varying SEE rates for each orbit. The SEE characterizations of the Zynq-7000 are used, which approximate the device susceptibility of several resource types to SEEs induced by protons and heavy ions. For a worst-case evaluation of the orbit, solar-minimum conditions and 100 mils of aluminum shielding are assumed. Figure 16 illustrates the fluctuation of the predicted L_m and SEE rate over time for each orbital case study. The detectable SEE rate aggregates the SEE rates of all accessible, on-chip bits (L1/L2 cache, on-chip memory, BRAM, and CRAM), which can be used to monitor SEEs. Table 2 shows the average, minimum, and maximum SEE rates predicted for each orbit. The SEE rate can fluctuate by up to three orders of magnitude depending upon the orbital position of the spacecraft. Furthermore, for all orbital case studies (except GOES-17), the SEE rates are within the lower 10% of the anticipated SEE rates for most of the orbital period, and some spacecraft (e.g., Jason-3, NOAA-20, and Molniya 1-88) are often within the lower 1%. Although the worst-case SEE rates must be considered to ensure high availability, these predictions demonstrate that the worst-case conditions are infrequent, and a static system designed for the worst-case conditions can result in inefficient utilization of system resources for most of the orbit.

3.4.2 Evaluation

The HARFT architecture is implemented on the Z7020 platform, the same device featured on the CSP. The HPS on the dual-core CPU allows for SMP (one master OS with two cores) and AMP (one master OS and one remote OS with one core each) operation. Figure 17 illustrates the floorplan of the triplex configuration of the SPS implemented on the FPGA. The static logic is triplicated using the BL-TMR tool, and the PRMs include

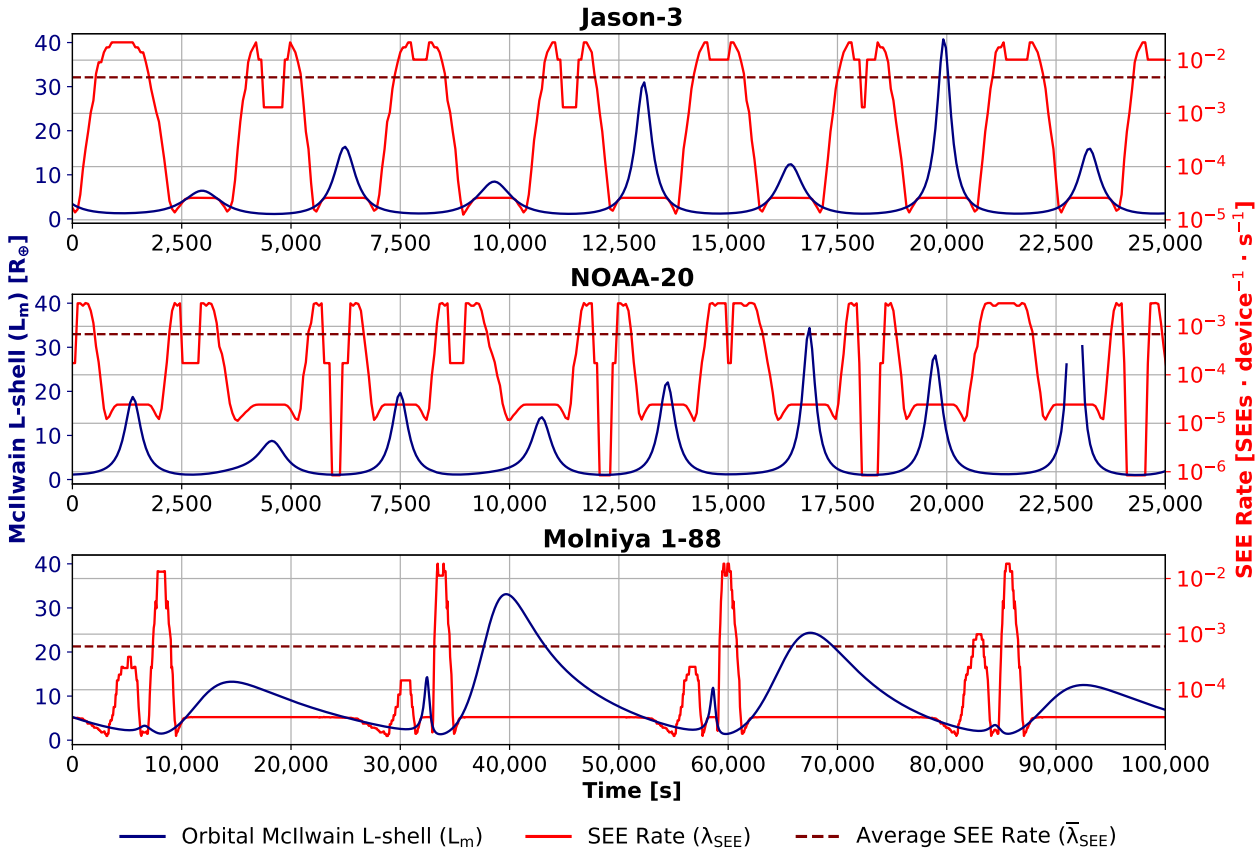
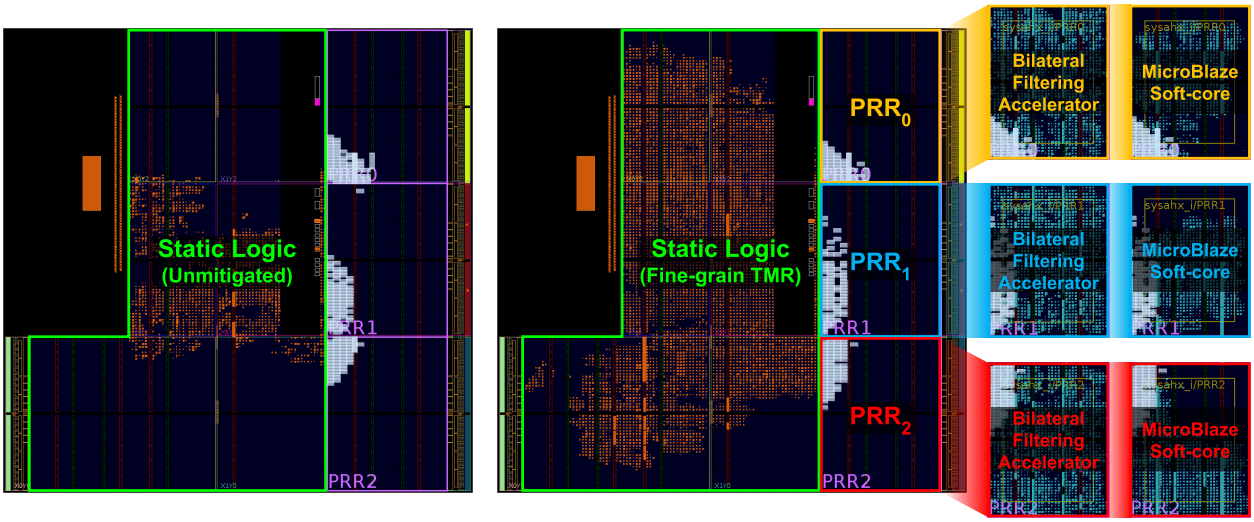


Figure 16: Predicted McIlwain L-shell (L_m) and SEE rate ($\lambda_{\text{SEE detectable}}$) for Z7020 over time for the Jason-3, NOAA-20, and Molniya 1-88 orbital case studies.

Table 2: Predicted SEE rates for Z7020 for orbital case studies.

Spacecraft	Predicted SEE Rate (λ_{SEE}) (SEEs · device^{-1} · day^{-1})			Percentage of Orbital Period (%)		
	Avg	Min	Max	$\lambda_{SEE} <$ Avg	$\lambda_{SEE} < 10\%$	$\lambda_{SEE} < 1\%$
ISS	4.48	0.05	17.80	68.45	52.81	23.59
Jason-3	410.84	1.08	1,864.50	68.01	65.37	50.32
EO-1	28.55	0.07	146.99	74.11	70.40	27.46
NOAA-20	59.55	0.07	261.95	74.19	72.18	55.07
Molniya 1-88	51.94	1.27	1,600.48	92.86	95.03	90.21
GOES-17	2.85	2.85	2.85	N/A	N/A	N/A



(a) (b) (c)

Figure 17: HARFT SPS framework implemented on the Z7020 in PRR triplex configuration with unmitigated static logic, FG-TMR static logic, and PRMs (bilateral-filtering accelerators and MicroBlaze soft-cores).

Table 3: HARFT CPU and accelerator performance and reward rates.

Module	CoreMark (iterations $\cdot$ s^{-1})	Bilateral Filtering (FPS)	Reward ($\bullet$)
ARM Cortex-A9 Master Core (667 MHz)	2,411.29	0.14	1.00
ARM Cortex-A9 Remote Core (667 MHz)	2,348.66	0.13	1.00
MicroBlaze (100 MHz)	202.77	0.01	0.10
Accelerator (100 MHz)	N/A	23.79	1.00

bilateral-filtering accelerators and MicroBlaze soft-cores. The implemented design is used to determine the performance reward rates, fault rates, and repair rates, which are specified as input parameters of the CTMC-based phased-mission system model.

3.4.2.1 Performance

The individual performance of each control agent and accelerator is measured to specify the potential performance reward rate for each state. The performance of each control agent is measured using the CoreMark benchmark and bilateral-filtering application. The performance of the accelerator is also measured using the bilateral-filtering application. Table 3 shows the measured performance of each module. The assignment of performance reward rates is user-defined and depends upon the mission requirements. For this evaluation, each state is assigned a reward rate that is the sum of (1) the reward rate of the CoreMark performance achieved by the active control agent, baselined to the highest-performing agent, and (2) the reward rate of the bilateral-filtering performance of each accelerator available for parallel operation, if a control agent is available.

Table 4: HARFT module resource utilization.

Subsystem	Configuration	LUTs	FFs	BRAM (18b×1k)	DSPs	CRAM Bits
Accelerator	Unmitigated	2,964	5,193	15	24	761,686
MicroBlaze	Unmitigated	2,720	2,073	21	6	581,801
Static Logic	Unmitigated	3,293	2,731	9	0	735,000
Static Logic	FG-TMR	14,917	7,409	27	0	2,629,400

Table 5: HARFT CRAM fault-injection test results.

Subsystem	Configuration	Injections	Errors	AVF	95% Confidence Interval
Accelerator	Unmitigated	2,020,657	608,241	30.101%	[30.026%, 30.177%]
MicroBlaze	Unmitigated	2,171,329	359,137	16.540%	[16.486%, 16.594%]
Static Logic	Unmitigated	1,910,970	119,964	6.278%	[6.242%, 6.313%]
Static Logic	FG-TMR	2,120,657	15,082	0.711%	[0.700%, 0.723%]

3.4.2.2 Resource Utilization and Architectural Vulnerability Factor

To approximate the fault rates for modules in the HARFT architecture, the module resource utilization and AVF are determined. The Xilinx design tools are used to determine the resource utilization of each module, which is shown in Table 4. CRAM fault injection is performed to approximate the AVF of these modules. In our fault injection procedure, each iteration begins with the system in a clean state. Next, the location (frame address, word, and bit) of a CRAM bit is randomly selected for fault injection. A frame-readback command is issued to the PCAP to retrieve the contents of the frame containing the selected CRAM bit into a software buffer. A single bit flip is performed on the buffered frame to inject the fault, and a frame-writeback command is issued to the PCAP to write the buffered frame back to CRAM to complete the injection. Next, the application is executed until completion (i.e., the accelerator processes one 512×512 image or the MicroBlaze runs one iteration of CoreMark). Once the execution is complete, a checksum of the output is compared against a golden checksum to determine if the outcome is correct or erroneous. An erroneous outcome indicates that the injected bit is a critical bit for that application and input, and these critical bits are counted. Finally, the system is reset into a clean state for the subsequent iteration. To minimize uncertainty in the measurements, a significant number of fault injections are performed. To accelerate this process, the Xilinx design tools are used to generate a list of essential CRAM bits (directly used by the design) to target exclusively [47]. Furthermore, eight TUL PYNQ-Z2 boards are used to parallelize the fault injection process. The AVF is calculated using Equation (2.1). Table 5 shows the resulting AVF with 95% CI of each tested module.

3.4.2.3 Time-Varying Fault Rate

Using Equation (2.4), the fault rate of each module can be determined. For each module, the resource utilization and AVF are used to scale the SEE rate of each resource type. For this evaluation, the experimentally measured AVF quantities are used for the CRAM resource

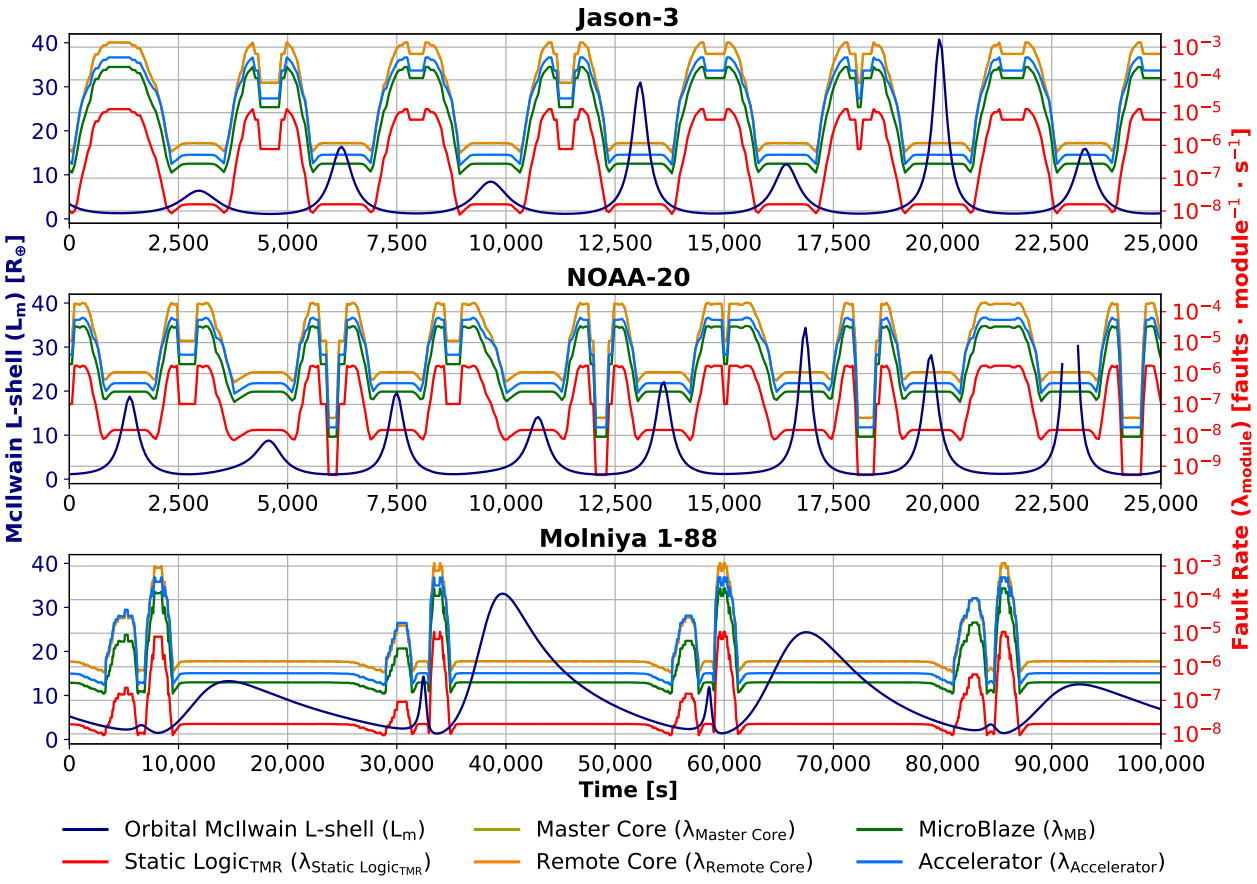


Figure 18: Predicted McIlwain L-shell (L_m) and fault rates (λ_{module}) of multiple HARFT modules for Z7020 over time for the Jason-3, NOAA-20, and Molniya 1-88 orbital case studies.

Table 6: HARFT repair rates.

Repair Mechanism	Repair Rate			
	Periodic Recovery		Event-Driven Adaptation	
	(repairs · s^{-1})	(s)	(repairs · s^{-1})	(s)
Master Core Recovery	N/A	N/A	10.0000	0.1
Remote Core Recovery	0.1000	10.0	10.0000	0.1
CRAM Scrubbing	0.0167	60.0	0.0167	60.0
Partial Reconfiguration	0.0167	60.0	10.0000	0.1
Full Reconfiguration	0.0167	60.0	10.0000	0.1
Watchdog System Reset	0.0083	120.0	0.0083	120.0

type, and the worst-case AVF is assumed for all other resource types. Finally, the scaled SEE rates are aggregated to produce the time-varying fault rate for each module. Figure 18 illustrates the approximated time-varying fault rates of several HARFT modules.

3.4.2.4 Repair Rate

Finally, the repair rates are determined to model the recovery mechanisms of HARFT. Table 6 shows the recovery mechanisms and their corresponding rates. Repair rates are measured for two scenarios. One scenario is periodic recovery, where the CM periodically checks the system to identify failed modules and perform repairs. The other scenario is an event-driven adaptation, where the CM is immediately invoked to reconfigure the system during a phase change.

3.4.3 Availability, Failure Rate, and Performability Analysis

The HARFT architecture is modeled as a CTMC-based phased-mission system using the performance reward rates, time-varying module fault rates, and module repair rates determined in Section 3.4.2. Table 7 shows the static modes and adaptive strategies used for this evaluation. CTMCs are generated to independently model each static mode. For *static*

Table 7: HARFT static modes and adaptive strategy.

Static Mode	HPS	SPS Accelerators	SPS MicroBlazes
Mode_0	SMP	3	0
Mode_1	AMP	3	0
Mode_2	SMP	2	1
Mode_3	AMP	2	1
Mode_4	AMP	0	TMR

Adaptive Strategy[a]

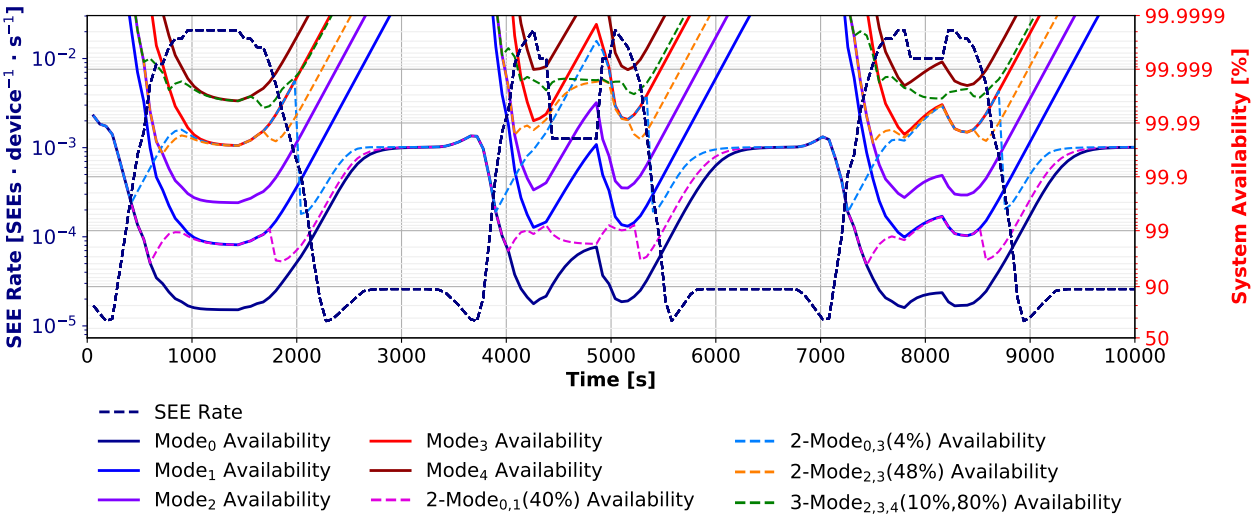
$$N\text{-Mode}_{1,2,3,\ldots,N}(\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_{N-1})(t) = \begin{cases} \text{Mode}_1, & \text{if } \lambda_{\text{SEE}}(t) < \alpha_1 \\ \text{Mode}_2, & \text{if } \lambda_{\text{SEE}}(t) \in [\alpha_1, \alpha_2) \\ \text{Mode}_3, & \text{if } \lambda_{\text{SEE}}(t) \in [\alpha_2, \alpha_3) \\ \quad\vdots & \\ \text{Mode}_N, & \text{if } \lambda_{\text{SEE}}(t) \geq \alpha_{N-1} \end{cases}$$

[a] Threshold parameters $\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_{N-1} \in [\lambda_{\min}, \lambda_{\max}]$, where $\alpha_1 \leq \alpha_2 \leq \alpha_3 \leq \ldots \leq \alpha_{N-1}$, and $[\lambda_{\min}, \lambda_{\max}]$ are the extrema of the expected range of SEE rates.

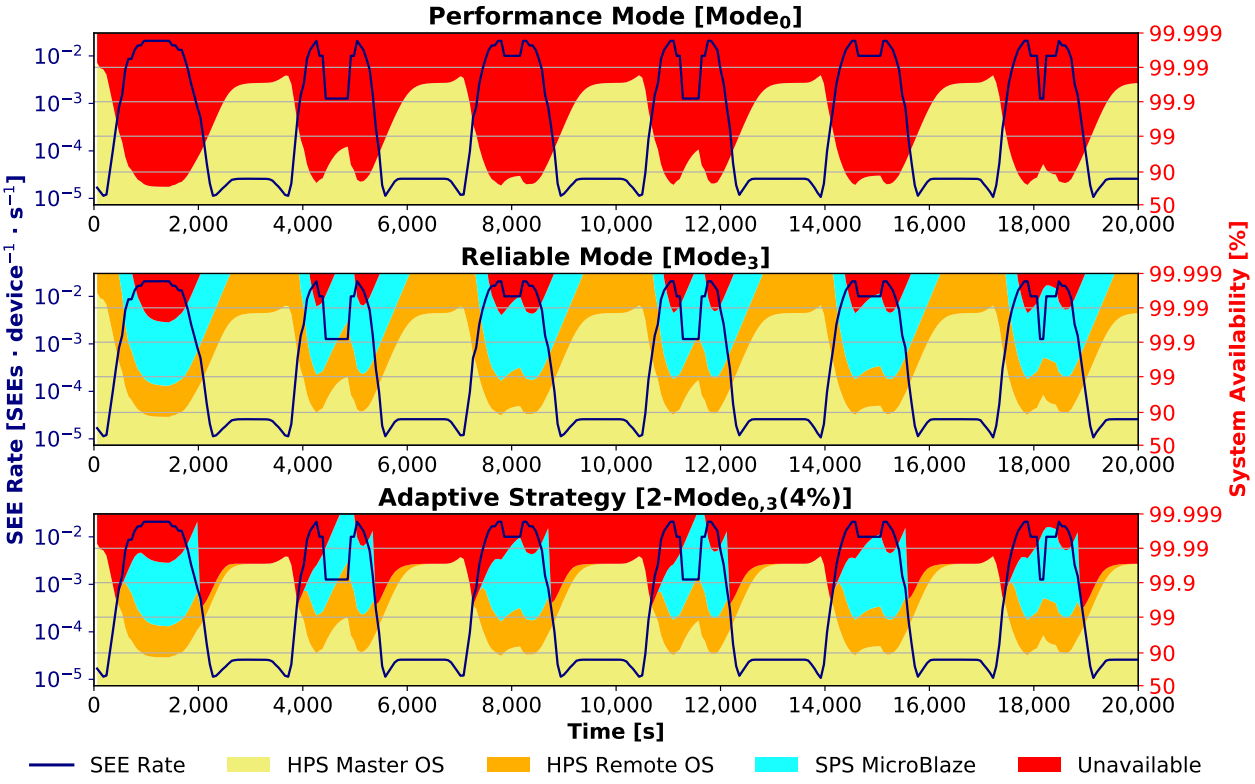
strategies, only one mode is active throughout the evaluation period. *Adaptive strategies* (denoted as N-Mode) adapt between N different modes over time using a threshold-based approach. The detectable SEE rate from Section 3.4.1 is compared to user-defined thresholds to determine the active mode over time. A transient analysis is performed for each static and adaptive strategy using 60-second timestep intervals over a one-week period. At each timestep, the module fault rates are updated, and the CTMC representing the active phase of the system is changed if the detectable SEE rate crosses any thresholds.

In our threshold-based approach, the adaptive strategies attain a combination of the properties of the modes in use. Figure 19(a) illustrates the instantaneous availability over time for numerous static and adaptive strategies in the Jason-3 orbital case study. Depending upon the thresholds and the fluctuating SEE rate, the instantaneous availability, failure rate, and performability of the adaptive strategies will alternate between those of the modes in use. Furthermore, because the HARFT architecture contains numerous failover paths to redundant control agents, the availability of control agents can also vary over time for the adaptive strategies. To determine the instantaneous probability that a control agent is active, the states of the phased-mission system model are divided into subspaces labeled by the active control agent, and a transient analysis is performed. Figure 19(b) illustrates the instantaneous probabilities of active control agents over time for three strategies in the Jason-3 orbital case study. Similarly, the adaptive strategies will attain some combination of the active control agent probabilities of each mode in use.

Analyzing the phased-mission system for varied threshold parameters produces a trade space in terms of the average system availability and performability. Figure 20 illustrates the trade space including several static and adaptive strategies for the Jason-3, NOAA-20, and Molniya 1-88 orbital case studies. The trade space is used to determine the optimal strategy subject to some constraints. For a given availability constraint, the optimal strategy is the one that satisfies that constraint and offers the most performability, and vice versa. The Pareto-optimal set is dominated by 2-Mode strategies, with some static and 3-Mode strategies but no 4-Mode or greater strategies. Because adaptive strategies attain the properties of each mode they use, adapting across too many modes can attenuate and negate the advantages of each.



(a) Instantaneous system availability versus time



(b) Instantaneous system control-state probability versus time

Figure 19: HARFT instantaneous system availability over time for various static and adaptive strategies of HARFT in the Jason-3 orbital case study, with (a) an overlay comparing strategies and (b) the instantaneous system control-state probability over time.

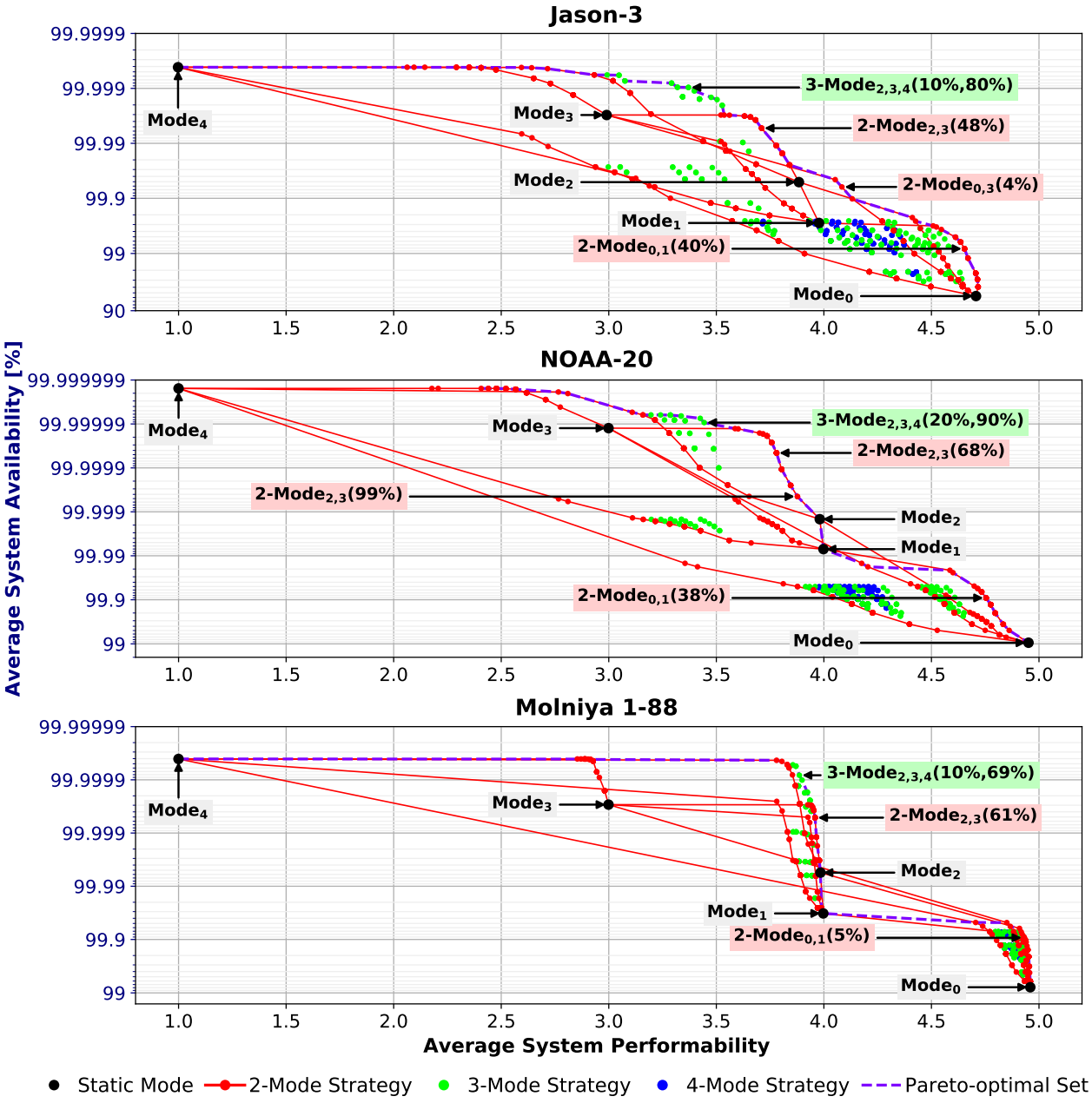


Figure 20: HARFT design tradespace in terms of performability and availability and with Pareto-optimal curves for static and adaptive strategies.

Table 8: HARFT unavailability, failure rate, and performability for orbital case studies.

Spacecraft Strategy	Availability Constraint (%)	Unavailability (days · yr^{-1}) Average	Unavailability (days · yr^{-1}) Maximum	Failure Rate (failures · day^{-1}) Average	Failure Rate (failures · day^{-1}) Maximum	Performability (•) Average	Performability (•) Improvement[a]
ISS							
Mode_0	≥99.9	2.63×10^{-1}	9.71×10^{-1}	5.19×10^{-1}	2.07×10^{0}	5.00	
2-$\text{Mode}_{0,1}$(18%)	≥99.99	3.06×10^{-2}	1.74×10^{-1}	6.05×10^{-2}	3.43×10^{-1}	4.63	1.16× (Mode_1)
2-$\text{Mode}_{0,1}$(3%)	≥99.999	2.10×10^{-3}	2.05×10^{-2}	4.16×10^{-3}	6.40×10^{-2}	4.25	1.06× (Mode_1)
Mode_1	≥99.9999	1.05×10^{-4}	6.76×10^{-4}	2.19×10^{-4}	1.50×10^{-3}	4.00	
Jason-3							
Mode_0	≥90	2.07×10^{1}	8.42×10^{1}	4.83×10^{1}	2.16×10^{2}	4.71	
2-$\text{Mode}_{0,1}$(40%)	≥99	3.00×10^{0}	1.77×10^{1}	6.13×10^{0}	5.65×10^{1}	4.65	1.17× (Mode_1)
Mode_1	≥99	1.02×10^{0}	6.68×10^{0}	2.10×10^{0}	1.35×10^{1}	3.98	
Mode_2	≥99.9	1.86×10^{-1}	1.11×10^{0}	3.75×10^{-1}	2.23×10^{0}	3.88	
2-$\text{Mode}_{2,3}$(48%)	≥99.99	1.93×10^{-2}	1.44×10^{-1}	4.25×10^{-2}	4.64×10^{-1}	3.71	1.24× (Mode_3)
Mode_3	≥99.99	1.11×10^{-2}	9.70×10^{-2}	2.30×10^{-2}	1.93×10^{-1}	2.99	
3-$\text{Mode}_{2,3,4}$(10%,80%)	≥99.999	3.51×10^{-3}	1.93×10^{-2}	7.74×10^{-3}	5.76×10^{-2}	3.37	3.37× (Mode_4)
Mode_4	≥99.999	1.51×10^{-3}	1.44×10^{-2}	3.13×10^{-3}	2.90×10^{-2}	1.00	
EO-1							
Mode_0	≥99	1.64×10^{0}	7.60×10^{0}	3.28×10^{0}	1.74×10^{1}	4.98	
2-$\text{Mode}_{0,1}$(60%)	≥99.9	2.67×10^{-1}	2.39×10^{0}	5.28×10^{-1}	8.07×10^{0}	4.78	1.20× (Mode_1)
2-$\text{Mode}_{0,1}$(1%)	≥99.99	1.78×10^{-2}	7.00×10^{-2}	3.58×10^{-2}	2.24×10^{-1}	4.28	1.07× (Mode_1)
Mode_1	≥99.99	5.68×10^{-3}	3.87×10^{-2}	1.20×10^{-2}	9.89×10^{-2}	4.00	
NOAA-20							
Mode_0	≥99	3.44×10^{0}	1.43×10^{1}	6.97×10^{0}	3.07×10^{1}	4.95	
2-$\text{Mode}_{0,1}$(38%)	≥99.9	3.32×10^{-1}	2.30×10^{0}	6.60×10^{-1}	8.13×10^{0}	4.75	1.19× (Mode_1)
Mode_1	≥99.99	2.58×10^{-2}	1.56×10^{-1}	5.40×10^{-2}	3.27×10^{-1}	4.00	
Mode_2	≥99.99	5.36×10^{-3}	3.02×10^{-2}	1.11×10^{-2}	6.11×10^{-2}	3.98	
2-$\text{Mode}_{2,3}$(99%)	≥99.999	1.64×10^{-3}	1.52×10^{-2}	4.04×10^{-3}	5.21×10^{-2}	3.88	1.29× (Mode_3)
Mode_3	≥99.9999	4.55×10^{-5}	3.34×10^{-4}	9.74×10^{-5}	6.82×10^{-4}	3.00	
Molniya 1-88							
Mode_0	≥99	2.86×10^{0}	7.44×10^{1}	6.55×10^{0}	1.86×10^{2}	4.96	
2-$\text{Mode}_{0,1}$(5%)	≥99.9	3.39×10^{-1}	4.83×10^{0}	6.77×10^{-1}	1.14×10^{1}	4.93	1.23× (Mode_1)
Mode_1	≥99.9	1.18×10^{-1}	4.83×10^{0}	2.40×10^{-1}	1.03×10^{1}	4.00	
Mode_2	≥99.99	2.03×10^{-2}	8.22×10^{-1}	4.07×10^{-2}	1.70×10^{0}	3.98	
2-$\text{Mode}_{2,3}$(61%)	≥99.999	1.88×10^{-3}	5.89×10^{-2}	4.05×10^{-3}	1.82×10^{-1}	3.96	1.32× (Mode_3)
Mode_3	≥99.999	1.07×10^{-3}	5.77×10^{-2}	2.18×10^{-3}	1.25×10^{-1}	3.00	
3-$\text{Mode}_{2,3,4}$(10%,69%)	≥99.9999	3.00×10^{-4}	1.58×10^{-2}	6.33×10^{-4}	3.29×10^{-2}	3.89	3.89× (Mode_4)
Mode_4	≥99.9999	1.48×10^{-4}	7.94×10^{-3}	3.03×10^{-4}	1.80×10^{-2}	1.00	
GOES-17							
Mode_0	≥99.9	1.31×10^{-1}	1.31×10^{-1}	2.58×10^{-1}	2.58×10^{-1}	5.00	
Mode_1	≥99.9999	1.27×10^{-5}	1.27×10^{-5}	2.51×10^{-5}	2.51×10^{-5}	4.00	

[a]The performability improvement compares the adaptive strategy to the static strategy that offers the highest performability and satisfies the same availability constraint.

Table 8 shows the average and worst-case unavailability, failure rate, and performability results for several strategies subject to select availability constraints (orders of nine). The performability improvement is measured by comparing adaptive strategies against the best performing static strategy that satisfies the same availability constraint. For example, in the ISS case study, both Mode_0 and 2-$\text{Mode}_{0,1}$(18%) satisfy the availability constraint of $\geq$ 99.9% (three nines), but 2-$\text{Mode}_{0,1}$(18%) has a 1.16× performability improvement over Mode_1. Similarly, in the Jason-3 case study, both Mode_4 and 3-$\text{Mode}_{2,3,4}$(10%,80%) satisfy the same availability constraint $\geq$ 99.999% (five nines), but 3-$\text{Mode}_{2,3,4}$(10%,80%) has a 3.37× performability improvement over Mode_4. In some cases, such as the $\geq$ 99% (two nines) availability constraint of the NOAA-20 case study, there is no adaptive strategy that outperforms Mode_1. Ultimately, the performability improvement depends upon the reliability and performance characteristics of each mode used for adaptation as well as several user-defined parameters (e.g., recovery rates, performance reward rates, and availability constraints).

The ISS, Jason-3, EO-1, NOAA-20, and Molniya 1-88 spacecraft all experience relatively low fault rates for most of the orbit, with short intervals of relatively high fault rates. As such, spacecraft in these orbits can benefit from adaptive resilience. The GOES-17 (GEO) spacecraft experiences minimal fluctuations in the SEE rate, thus making it infeasible for adaptation in response to the SEE rate based on the dynamic near-Earth radiation models described in Section 3.3.1. However, the approach in the work of Glein et al. [29] can be used to adapt in response to the SEE rate based on the dynamic solar condition. Although this evaluation focuses on the near-Earth radiation environment, which is nearly periodic for orbital missions, adaptive strategies are also applicable to missions where high reliability is required during phases of critical operations. The system can adapt in response to changes in the criticality of mission phases instead of a fluctuating SEE rate. In this approach, the failure rate is useful for determining the optimal high reliability and high-performance modes of operation for critical and noncritical phases, respectively.

3.5 Conclusion

As spacecraft designers continue to adopt SmallSat technology, there is a need for dependable, high-performance onboard processing to address the computational demands required for future missions. Commercial hybrid SoCs provide numerous architectural advantages for onboard space computing, but these devices are highly susceptible to radiation compared to traditional rad-hard alternatives. The dynamics of the near-Earth radiation environment expose spacecraft to radiation fluxes that can vary the SEE rate by multiple orders of magnitude. Due to the duality of high-performance and high-reliability system design, an adaptive approach to dependable computing can more efficiently use system resources by responding to the dynamic environment versus a static system designed for the worst-case condition.

In this chapter, we proposed HARFT, a reconfigurable framework for environmentally adaptive resilience in reconfigurable space systems. The HARFT architecture combines runtime-reconfigurable fault-tolerance modes for the CPU and FPGA subsystems of the hybrid SoC into one integrated, synergistic framework. HARFT provides numerous recovery and failover mechanisms within and between subsystems to enable repair and graceful degradation. Additionally, we extended the modeling methodology proposed by Jacobs et al. [35] to evaluate adaptive, gracefully degradable systems for the near-Earth radiation environment. Using this methodology, we evaluate the HARFT architecture for numerous static and adaptive strategies subject to the radiation environment of various orbital case studies. By responding to the fluctuating fault rate, HARFT can adapt to the dynamic environment by selecting operating modes that maximize performability while satisfying availability requirements throughout the mission. When evaluated on the Zynq-7000 Z7020, we demonstrate substantial performability improvements for given system availability constraints.

4.0 Resilient Semantic-Segmentation Acceleration for Space Apps

DL presents several opportunities for enhancing spacecraft autonomy, onboard data analysis, and intelligent applications for space missions. One example is semantic segmentation, a powerful ML/CV process that learns to classify pixels within an image. Semantic segmentation has numerous applications in onboard remote sensing for both science and defense missions, from analyzing EO for Earth science (e.g., land use, land cover, and cloud masking), to monitoring natural disasters for emergency response, and to conducting reconnaissance for national security. Despite these advantages, DL models are computationally intensive and often impractical for deployment on traditional rad-hard space processors. Commercial FPGAs and SoCs provide superior performance, energy efficiency, and affordability compared to their rad-hard alternatives but are highly susceptible to radiation-induced SEEs that can affect the dependability of the system and application [56]. To improve dependability, fault-masking techniques such as triple-modular redundancy (TMR) are frequently employed for SEE mitigation. However, TMR incurs significant overhead in area, power consumption, and timing-critical path which is impractical for resource-constrained systems and can also limit the performance and energy-efficiency potential of a system. To create a dependable and high-performance system capable of onboard DL, efficient approaches in SEE mitigation are essential.

In this chapter, we propose Reconfigurable ConvNet (RECON), a runtime-reconfigurable acceleration framework for dependable, high-performance semantic segmentation for space applications on FPGAs and SoCs. RECON uses several model-compression, algorithmic, and architectural optimization techniques to maximize the inference performance, energy efficiency, and area efficiency for onboard processing. In RECON, we propose both selective and adaptive strategies to enable efficient SEE mitigation. RECON is disaggregated into separate control-flow and data-flow subsystems. In our selective approach, the control-flow subsystem, which is vulnerable to SEE-induced hangs, is selectively protected with TMR to minimize the frequency of hangs that are disruptive and slow to repair. In our adaptive approach, the data-flow subsystem, which is more vulnerable to SEE-induced SDC but

is faster to repair, is protected using an environmentally adaptive strategy leveraging dynamic PR. Due to the dynamics of the near-Earth radiation environment, spacecraft are exposed to SEE rates that can vary by multiple orders of magnitude. Using PR, RECON can adapt its data-flow subsystem by alternating between parallel (performance) and redundant (dependable) configurations in response to the fluctuating SEE rates of the dynamic near-Earth radiation environment. RECON selects the data-flow configuration that uses only the amount of redundancy that is necessary for the immediate environmental condition and uses remaining resources for performance. Combined, both approaches enable RECON to maximize performability, in terms of performance and energy efficiency, subject to mission availability constraints. Finally, to demonstrate the efficacy of RECON for onboard semantic segmentation, we evaluate this framework accelerating the SegNet model [6], a symmetric encoder-decoder architecture for semantic segmentation, in terms of accuracy, resource utilization, performance, energy-efficiency, performability, and availability. In our dependability evaluation, we perform fault injection and neutron irradiation to analyze the SEE susceptibility of SegNet accelerated on RECON, and we use dependability modeling to evaluate RECON in various orbital case studies to demonstrate a 1.5-3.0× performability improvement in performance and energy-efficiency compared to static approaches.

4.1 Related Work

The evaluation, analysis, and mitigation of SEEs in machine-learning applications accelerated on FPGAs have been explored in the literature [99, 52, 53, 19, 9, 25, 20, 86]. An overview of concepts and taxonomy for dependability in FPGA-based NNs, including passive and active methods for fault tolerance, is provided in [99]. A variety of methods using fault injection and radiation-beam testing have been explored to evaluate the dependability of NNs. Du et al. [20] performed fault injection, targeting both static and dynamic CRAM with single-bit and multi-bit faults, to evaluate the susceptibility of a binary NN to single-bit and multi-bit upsets in various resource types. Benevenuti et al. [9] characterized the SEE susceptibility of a multi-layer perceptron for Iris flower classification accelerated on

the Zynq-7000 in terms of tolerable and critical SDC. Layers of the NN were assigned to separate FPGA partitions to analyze the design susceptibility at the model and layer levels. Dos Santos et al. [19] used fault injection and neutron irradiation to evaluate the impact of double-, single-, and half-precision floating-point data representations on the reliability of an MNIST CNN implemented on the Zynq-7000. The reduced area due to reduced precision decreased the critical area. Libano et al. [53] used fault injection to evaluate the impact of binary quantization on the reliability of an MNIST CNN implemented on the Zynq-MPSoC. The reduced area due to quantization decreased the critical area but increased the error criticality.

Methods to improve NN dependability using efficient methods for SEE mitigation have also been explored. Libano et al. [52] used fault injection to identify the most vulnerable layers of two fully unrolled models, Iris flower NN and MNIST CNN, accelerated on the Zynq-7000 and Zynq-MPSoC, respectively. Selective TMR was applied to protect the most vulnerable layers of each model to reduce redundancy overhead. Gambardella et al. [25] used fault injection to identify the most vulnerable channels of a binary NN. Selective TMR was applied to the PEs processing the most vulnerable channels to reduce redundancy overhead. For folded implementations where PEs each process multiple channels, Gambardella et al. proposed a fault-aware scheduler to schedule channels through mitigated or unmitigated PEs based on the vulnerability of the channel (e.g., the most vulnerable channels run through mitigated PEs). Sabogal et al. [86] performed fault injection and neutron irradiation to evaluate a CNN accelerator for the SegNet model on the Zynq-7000 and Zynq-MPSoC. Due to the impracticality of unrolling deep CNNs on resource-constrained FPGAs, a reusable instruction-based CNN architecture was created. TMR was selectively applied to the control-flow part of the acceleration framework to minimize the hang rate, and high-performance, unmitigated and low-performance, TMR versions of the data-flow part were evaluated to quantify the SDC rate and the tradeoffs in performance and dependability.

In this chapter, we extend upon our previous work in [86] and make the following contributions. First, we present an updated acceleration framework for RECON that is comparable to the current paradigm of state-of-the-art CNN architectures, including instruction-based processing and model-compression, algorithmic, and architectural optimizations that maxi-

mize inference performance with efficient hardware. Second, we propose the partitioning of control-flow and data-flow parts of RECON into static and reconfigurable regions, respectively, and applying selective TMR to protect control-flow parts to reduce the hang rate. Leveraging the reconfigurability of FPGAs, we also propose an environmentally adaptive approach to mitigate SDC in the data-flow part in response to the environmental condition. Combined, both approaches can maximize inference performability subject to mission availability constraints. Finally, we evaluate the susceptibility of the SegNet model accelerated on RECON for the Zynq-7000 and Zynq-MPSoC using fault injection and neutron irradiation, and we discuss our methodology and analyze the architectural response of RECON to both injected CRAM faults and neutron-induced SEEs at the model and layer levels.

4.2 Architecture Overview

This section provides an architectural overview of the RECON framework, which is illustrated in Figure 21. RECON is a runtime-reconfigurable acceleration framework for dependable, high-performance semantic segmentation for space applications. The framework is composed of three major modules including the *Configuration Manager* (CM), *RECON Scatter-Gather DMA* (RSGDMA) and *RECON Accelerator* (RACCEL).

The CM is responsible for three functions: (1) environmental monitoring, (2) system reconfiguration and adaptation, and (3) fault management. To assess the environmental condition, the CM can monitor radiation stimuli, using on-chip or onboard SEE-detection circuitry or external radiation-flux sensors or dosimeters, or use model-based predictions. In response to the severity of the environmental condition or criticality of the mission phase, the CM adapts the system by using PR to dynamically reconfigure each PRR at runtime without interrupting system operation. For fault management, the CM performs periodic CRAM scrubbing, event-driven MER using PR, and FR. The CM exists as software on a CPU or as a controller residing in the SR.

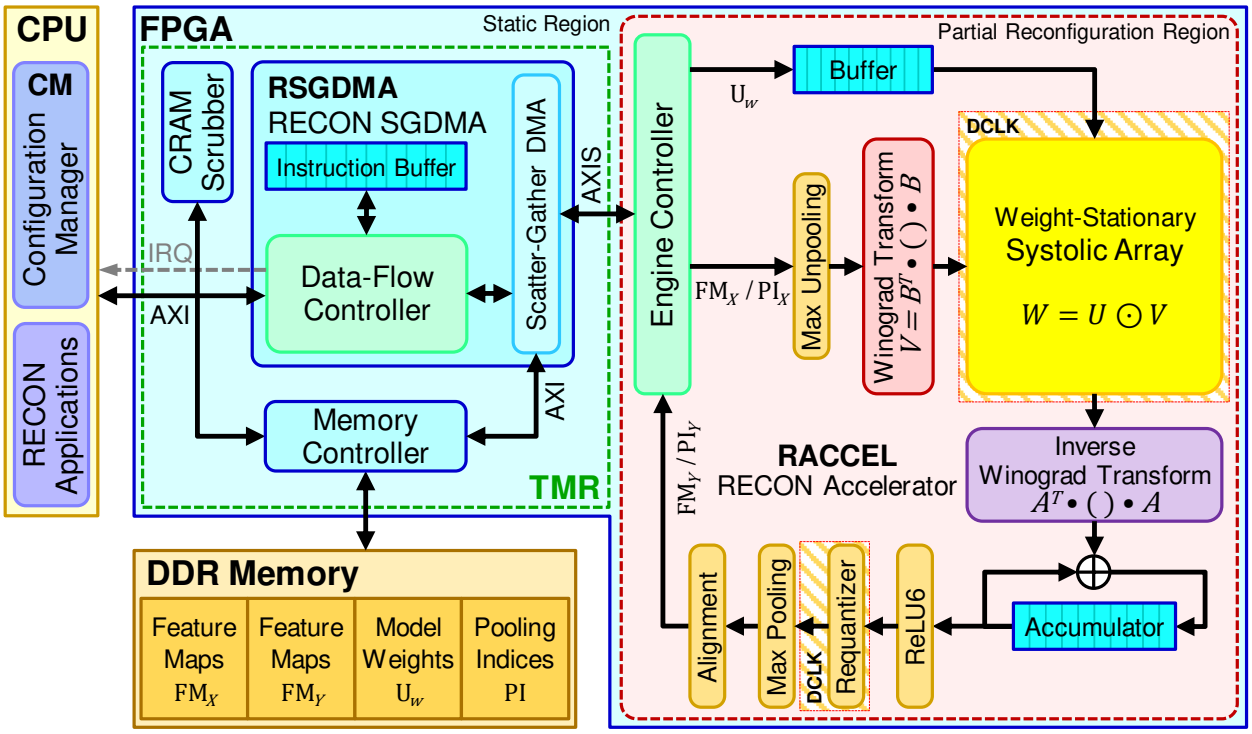


Figure 21: RECON acceleration framework.

The RSGDMA and RACCEL modules constitute the control-flow and data-flow subsystems of the RECON acceleration framework, respectively. The RSGDMA performs most of the control-flow functions of the framework, including instruction processing, accelerator configuration, and memory access, and resides in the SR. The RACCEL performs all data-flow functions of the framework to accelerate SegNet processing with optimizations and resides in a PRR as a PRM. Both the RSGDMA and RACCEL modules are scalable and runtime parameterizable to accommodate various FPGA platforms and application domains and to support runtime reconfiguration. The degree of parallelism in RECON is primarily defined by the number of input channels and output channels N processed concurrently, with four parallel pixels per channel, for a total of $16N^2$ PEs. The notations RSGDMA_M and RACCEL_N are used to denote the configuration of RSGDMA and RACCEL, respectively, where M and N are user-specified, pre-synthesis parameters. Both RSGDMA_M and RACCEL_N are compatible if $N \leq M$.

The RSGDMA and RACCEL interface via AXI4-Stream (AXIS), and AXIS packets are used to parameterize and operate the RACCEL. Input stream packets specify the datapath configuration and provide weights, biases, quantization parameters, and input data of tiled FMs and PIs. Output stream packets return output data of tiled FMs and PIs. The RSGDMA has a built-in decoupling mechanism that can sever the AXIS interface between the RSGDMA and RACCEL. This mechanism is activated during PR to protect the RSGDMA by ensuring that the AXIS interface remains inactive, and this mechanism can also be activated to inhibit the propagation of errors from faulty RACCEL PRMs to the RSGDMA and other static logic.

4.2.1 Approaches for Efficient SEE Mitigation

In RECON, we focus on mitigating two event classifications due to faults: *silent data corruption* (SDC) and *hangs*. SDC refers to an erroneous outcome of the application due to faults and is neither detectable nor correctable without dependable-computing techniques. SDC usually occurs when faults affect the data-flow parts of the design (e.g., faults corrupting the datapaths). Faults causing SDC can be repaired by CRAM scrubbing or reconfiguration.

Depending upon the application, the severity of SDC can vary broadly. Some algorithms, including NNs, have been demonstrated to have an inherent fault tolerance due to high redundancy in the weights of the model [99]. SDC events with low severity (e.g., few incorrect pixels) are classified as *tolerable SDC* (SDC_T) if the accuracy loss remains below a user-defined tolerance threshold. Otherwise, SDC events with high severity (e.g., severe distortions) are classified as *critical SDC* (SDC_C). Depending upon mission requirements, if some loss in accuracy due to SDC is acceptable, then the dependability analysis is adjusted to focus on SDC_C.

A hang refers to the nonperformance of the application that can be detected by timeout or watchdog. A hang usually occurs when faults affect the control-flow parts of the design that can corrupt finite-state machines (e.g., entry into unreachable states), disrupt flow-control processes, or activate/inhibit control signals. The severity of hangs can also vary. Some hang conditions can be repaired by a combination of CRAM scrubbing and asserting a reset signal to repair the faulty control logic and reinitialize the control state. However, some hang conditions can propagate to other subsystems and require reconfiguration or external mechanisms (e.g., software-issued reboot or watchdog timer reset) to recover.

CRAM scrubbing and PR are fast, nondisruptive recovery mechanisms to repair faults. FR and other mechanisms that reset the FPGA are slow, disruptive recovery mechanisms that must be minimized to avoid system downtime. The application of FG-TMR can substantially reduce the critical area to minimize both SDC and hangs; however, FG-TMR incurs a substantial overhead in the design area, energy consumption, and timing-critical path that can limit the performance and energy-efficiency potential of the system. To improve the dependability of RECON with minimal impact on performance, we propose selective and adaptive strategies for efficient SEE mitigation of hangs and SDC, respectively. The RECON framework is disaggregated into control-flow (RSGDMA) and data-flow (RACCEL) subsystems, and the selective and adaptive approaches are applied to the RSGDMA and RACCEL subsystems, respectively.

4.2.1.1 Selective Mitigation for RSGDMA

Since hangs result in system downtime and require slow processes to recover, the critical area vulnerable to hangs must be minimized to reduce the hang rate. FG-TMR is selectively applied to the RSGDMA and supporting logic (e.g., interconnects and memory controllers) in the SR because these subsystems perform most of the control-flow functions of the framework. Additionally, FG-TMR will also reduce SDC due to faults in the RSGDMA.

Although RACCEL is mostly data-flow-oriented, this module is not devoid of control-flow function and is also vulnerable to hangs. However, the decoupling mechanism of the RSGDMA can be activated to sever the AXIS interface between the RSGDMA and RACCEL to inhibit the propagation of both SDC and hang conditions to protect the RSGDMA and other static logic, and the CM is invoked to perform MER using fast, nondisruptive repair mechanisms. For example, if RACCEL hangs and the RSGDMA runtime exceeds a predefined timeout, the CM activates the decoupler and performs PR to recover the RACCEL with minimal system interruption.

4.2.1.2 Adaptive Mitigation for RACCEL

Since the SEE rate of a system exposed to the dynamic near-Earth radiation environment can vary by multiple orders of magnitude, the application of static (nonchanging) SEE mitigation can be excessive and inefficient, especially when the worst-case SEE rates are infrequent or brief. An environmentally adaptive approach for SEE mitigation can repurpose system resources between parallelism (performance) and redundancy (dependability) in response to the current environmental condition. Using this approach, a system can adapt its resources to maximize performance while providing SEE mitigation that is sufficient to the environmental condition to satisfy mission availability constraints.

As a PRM, the RACCEL configuration can be changed at runtime, and the degree of parallelism and redundancy of the RACCEL configuration can vary but is constrained by the amount of resources available in the PRR. Each RACCEL configuration has its own performance, energy efficiency, and dependability tradeoffs. With several configuration modes available, the CM can adapt to the environment by selecting the RACCEL configuration

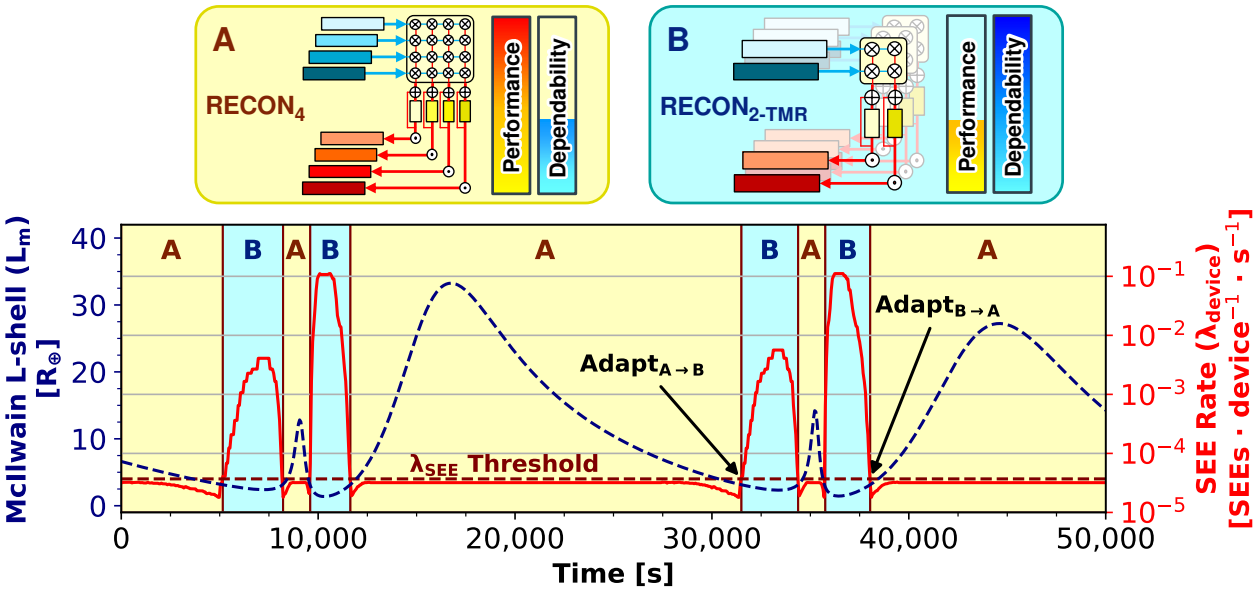


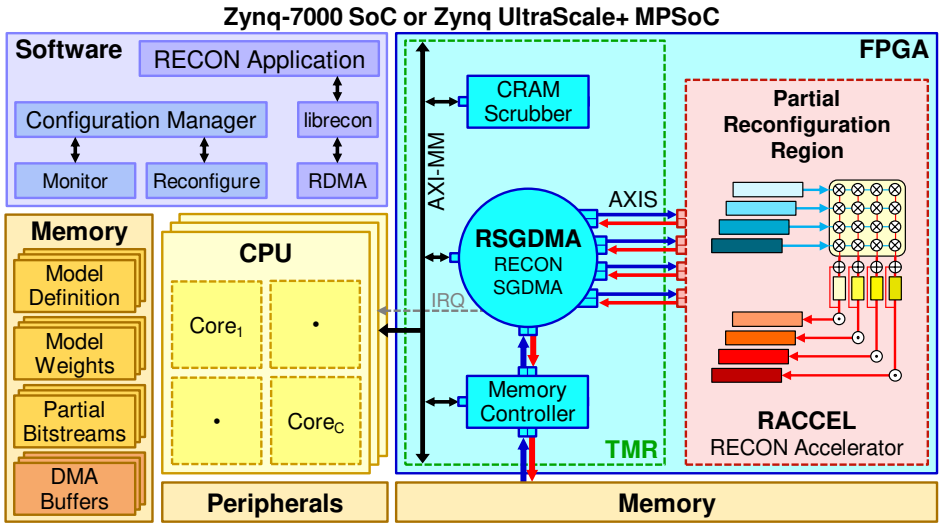
Figure 22: RECON adaptive approach for SEE mitigation selects between high-performance Mode_A (RECON_4) and high-dependability Mode_B ($\text{RECON}_{\text{2-TMR}}$) in response to the current SEE rate of the orbital environment.

with the tradeoffs best suited for the immediate environmental condition. The policy used by the CM to select a RACCEL configuration can also vary. One such policy is a threshold-based approach, where adaptation occurs when the monitored SEE rate crosses predefined thresholds.

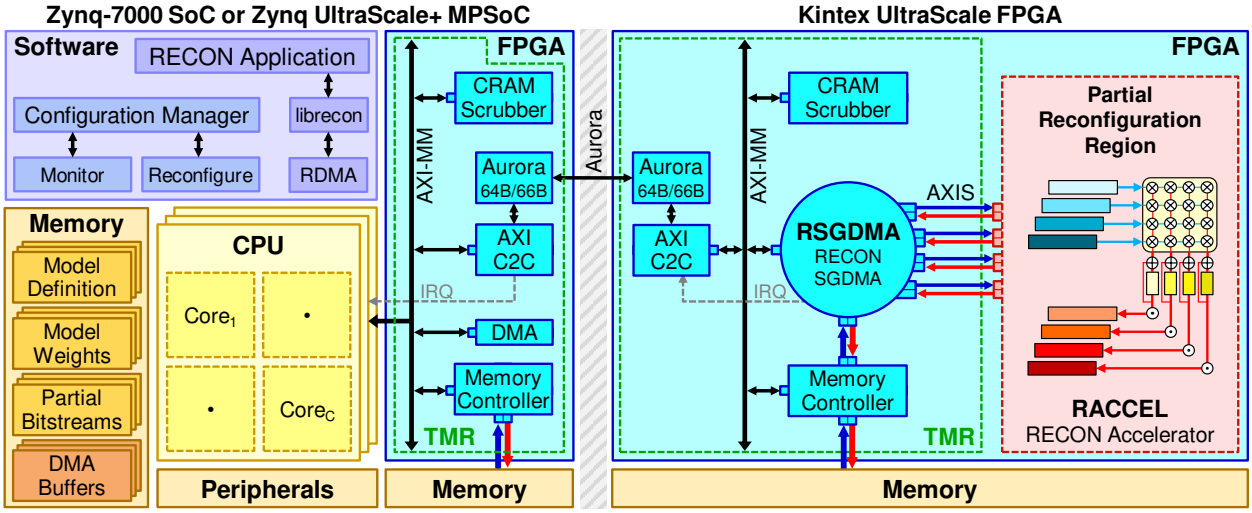
Figure 22 illustrates an example of this adaptive approach for a RECON framework with mitigated $\text{RSGDMA}_{\text{4-TMR}}$. In this example, the CM selects between high-performance Mode_{A} (RECON_4) or high-dependability Mode_{B} ($\text{RECON}_{\text{2-TMR}}$) using a threshold-based policy for mode selection. Both RACCEL configurations have similar resource utilization but different tradeoffs in performance and dependability. During periods with SEE rates below the threshold, Mode_{A} is deployed to maximize performance and energy-efficiency at the expense of dependability, and, during periods with SEE rates above the threshold, Mode_{B} is deployed to improve dependability at the expense of performance and energy-efficiency. Depending upon the orbit and mission availability constraint, this adaptive approach can achieve substantial performability gains compared to a static, high-dependability approach. In Section 4.3, we demonstrate the effectiveness of this adaptive approach and generate a design tradespace with static and adaptive strategies in terms of performability and availability. Using this tradespace, users can select the strategy that achieves the most performability subject to an availability constraint, and vice versa.

4.2.2 Architectures for Space Computers

The RECON framework can be deployed on space computers featuring a hybrid and heterogeneous SoC and system architecture. Space computers, such as the CSP (Z7020) [106], SSP (Z7030, Z7035, or Z7045) [82], and SpaceCube Mini-Z (Z7020) [13] that feature hybrid SoCs can accommodate the RSGDMA and RACCEL modules in the FPGA subsystem and run the CM software in the CPU subsystem, as illustrated in Figure 23(a). The CSP and SpaceCube Mini-Z computers, which do not contain FPGA-interfaced DDR memory, must reserve a partition of the CPU-interfaced DDR memory for use by the RSGDMA.



(a) Architecture for hybrid SoCs



(b) Architecture for heterogeneous SoC/FPGA systems

Figure 23: RECON architecture for space processors including (a) hybrid SoCs and (b) heterogeneous SoC/FPGA systems.

RECON can also be deployed on space computers featuring heterogeneous, disaggregated CPU-FPGA systems, which often combine a large FPGA coprocessor to a relatively low-profile CPU or SoC interfaced by a high-speed interconnect. One example is the space single-board computer architecture exemplified by the SCv3VPX design (Zynq-MPSoC and KU-FPGA) [27]. Another example is a space computer system with both the SoC and FPGA coprocessor as separate cards. Such a system is demonstrated by the SCv3M design (KU-FPGA) [13] connected to a SoC (e.g., SSP). In both examples, illustrated in Figure 23(b), the RSGDMA and RACCEL modules reside in the FPGA coprocessor and run the CM software in the SoC, and the model can be communicated to the RSGDMA via AXI Chip2Chip using the Aurora 64B/66B protocol with MGTs as the high-speed interconnect.

In both architectures illustrated in Figure 23, the FPGA containing RECON can serve as a coprocessor, where the adjacent CPU or SoC can offload massive workloads for acceleration with minimal communication overhead. Alternatively, the FPGA can serve as a front-end data processor for sensors interfaced directly with the FPGA. In this configuration, the FPGA can directly process raw sensor data and provide compressed data to the adjacent CPU or SoC for downlink or storage.

The RECON framework is supported by software, including Linux device driver and userspace library, that enables userspace applications to have shared access to the RSGDMA for inference acceleration. The RECON software is parameterizable to support arbitrary input image volumes (spatial resolution and dimension) and shapes or variations of a DL model to accommodate various space applications and imaging sensors (e.g., monochromatic, multispectral, or hyperspectral). When initialized, the software references two resources: the model definition, which specifies the shape of the model and the instructions to process the model, and the corresponding model parameters, which are the trained weights, biases, and quantization parameters that are loaded into memory prior to execution. Both resources are obtained after model development (training, testing, and analysis) and are uploaded to the onboard computer for deployment. For model development, a model can be constructed using a dataset generated from downlinked sensor data or approximated by using or manipulating existing datasets.

4.2.3 Accelerator Optimizations

Several optimization techniques from the literature have been incorporated into RECON to maximize the inference performance, energy efficiency, and area efficiency for onboard processing. This discussion includes model-compression, algorithmic, and architectural optimizations implemented into RECON.

4.2.3.1 Model-Compression Optimizations

RECON uses the INT8 quantization scheme of [34] for model compression. This quantization scheme is applied per-layer and per-channel for FMs and weights, respectively, with asymmetric quantization used for both FMs and weights. To improve the precision of the quantization mapping, a winsorizing approach is used to set all outliers and extreme values of the continuous input set to the edges of the user-specified percentile of the discrete set. Furthermore, ReLU layers, which have an unbounded range, $[0, \infty)$, are replaced with ReLU6 layers to constrain values to $[0, 6]$. With INT8 quantization, RECON uses resource-efficient, low-precision hardware to improve area and energy efficiency. INT8 quantization also improves the bandwidth and storage efficiency by 4× compared to single-precision FP (FP32). Combined with additional techniques in quantization mapping and model modifications, the loss in accuracy due to INT8 quantization can be a minimal and tolerable tradeoff for the substantial hardware-efficiency benefits.

4.2.3.2 Algorithmic Optimizations

Using the $F(2\times2, 3\times3)$ form of Winograd convolution, a fast algorithm for convolution, the RACCEL improves DSP efficiency by 2.25× compared to direct 3×3 convolution. Because FMs are determined at runtime, the Winograd transform for input FMs and inverse Winograd transform for output FMs are implemented into RACCEL. However, because the weights of convolutional layers are predetermined, the Winograd transform can be either

implemented using FPGA resources or be applied to the weights prior to model deployment. RECON uses the latter approach, which results in no FPGA resources being used for the Winograd transform for weights at the expense of a 1.78× larger model size.

Additionally, BatchNorm folding is used to embed the parameters of BatchNorm layers into the parameters of the preceding convolutional layer prior to model deployment. This optimization eliminates the need for RECON to process BatchNorm layers at runtime.

Finally, the RSGDMA implements the controls to perform loop tiling and access tiles of partitioned FMs stored in off-chip memory. These FM tiles are cached by the RACCEL using an OCM-based accumulator buffer of user-specified size. Because burst transactions of the RSGDMA AXI interface use an incrementing access pattern, the FMs are partitioned by rows to maximize the efficiency of DDR memory accesses and streaming bandwidth. Furthermore, because convolutional operations require complete kernel windows, tiles require additional rows from adjacent tiles to address the data dependency for the edge cases. Using this optimization, RECON substantially reduces the latency and bandwidth requirements of off-chip memory access by accumulating cacheable FM tiles in OCM.

4.2.3.3 Architectural Optimizations

The RACCEL uses a 2D weight-stationary systolic array for processing convolutional layers to achieve high-frequency operation. Each PE is implemented using one DSP slice to perform a single multiply-accumulate operation per cycle, and all PEs are interlinked using cascaded signals, which are dedicated paths between DSP slices in Xilinx FPGAs. Furthermore, since DSP slices of recent Xilinx FPGAs are rated for high-frequency operation, RACCEL uses DSP time-multiplexing with a factor of two to halve the number of DSP slices required by operating the DSPs at two times the frequency of the surrounding logic. In RACCEL, the weights are multiplexed into the inputs of the DSP, and the output of the DSP is demultiplexed into the surrounding logic.

Additionally, RECON uses layer fusion to process multiple adjacent layers in a pipeline. All instructions compute convolutional layers with optional preprocess or postprocess operations. Both the elementwise ReLU6 and compressive, channel-wise max-pooling layers are

optional postprocess operations that follow the convolutional layers. Inversely, the decompressive, channel-wise max-unpooling layers are optional preprocess operations that precede the convolutional layers. Combined with the BatchNorm folding optimization, RECON requires only 26 instructions to process all 86 layers of the SegNet model. This data-flow is illustrated in Figure 21.

Finally, the RSGDMA uses an AXI-based scatter-gather DMA (SGMDA), which contains multiple AXI descriptors to access multiple memory buffers to support scattering and gathering data-flows. In a scattering operation, the SGDMA rotates between AXI descriptors, each completing one AXI-burst transaction per rotation, to read input FMs or PIs from multiple memory buffers to generate an interleaved input stream for processing. Inversely, in a gathering operation, the SGDMA deinterleaves the output stream and writes the output FMs and PIs to multiple memory buffers. Because interleaving and deinterleaving are seamlessly performed as part of the scattering and gathering operations, the FMs and PIs remain deinterleaved in memory without the need for software interleaving or deinterleaving to reorganize accelerator inputs and outputs. Finally, pointers to memory buffers containing FMs and PIs are alternated at runtime for zero-copy to avoid inefficient memory copies. After each instruction, the buffer pointers are swapped so that the output FM buffer of the preceding instruction becomes the input FM buffer for the following instruction. During the encoder stages, PIs are generated and stored into output PI buffers, and, during the decoder stages, these PI buffers become inputs.

4.3 Evaluation

This section describes the performance and dependability evaluations for RECON. RECON is implemented for the Z7020 (PYNQ-Z2 and Zybo Z7-20) and ZU3EG (Ultra96-V2 and UltraZed-EG) that serve as facsimiles for hybrid space computers (similar to those described in Sections 2.2 and 4.2.2). The Z7020 and ZU3EG devices use the configuration illustrated in Figure 23(a) but use CPU-interfaced DDR memory for DMA buffers. In our performance evaluation, both platforms are evaluated in terms of accuracy, resource utilization, perfor-

mance, and energy efficiency. In our dependability evaluation, we use CRAM fault injection and neutron irradiation to evaluate the SEE susceptibility of the SegNet model accelerated on RECON for the Z7020. We also use dependability modeling to evaluate our adaptive strategy for various orbital case studies. For both platforms, Vivado 2020.1 is used to synthesize and implement the RECON design with default strategies, and PetaLinux 2020.1 is used to generate an embedded Linux OS. FG-TMR is applied using the BL-TMR tool [37]. The Potsdam dataset of the ISPRS commission II/4 benchmark for 2D semantic labeling [74] is used for this evaluation. This dataset uses EO images in IRRGB (infrared-red-green-blue) format with six classes for segmentation: roads, buildings, low vegetation, trees, automobiles, and clutter. Three shapes of the SegNet model are trained and evaluated: Net_A (86 layers, 7,376,806 weights), Net_B (86 layers, 1,849,814 weights), and Net_C (86 layers, 465,262 weights).

4.3.1 Performance Evaluation

This section quantifies and analyzes the RECON modules in terms of conventional metrics. Towards a dependability analysis, we measure the inference accuracy and resource utilization which affect the vulnerability of RECON to faults. Furthermore, we measure performance and energy efficiency to quantify the advantages of FPGA-accelerated DL and to define reward states to analyze the tradeoffs in performance and dependability for RECON.

4.3.1.1 Inference Accuracy

Because RECON uses INT8 quantization for efficient, low-precision hardware, the inference accuracy is measured for both FP32 and INT8 versions of the SegNet model to quantify the loss in inference accuracy. Using Equation (2.5), the interface accuracy of the segmented images is measured in terms of the mIoU and F1 metrics, and the results are shown in Table 9. For this evaluation, an mIoU difference of -0.7%--1.7% was observed with loss decreasing

Table 9: RECON inference accuracy.

Precision	Net_A (7.38M weights)		Net_B (1.85M weights)		Net_C (465k weights)	
	(mIoU)	(F1)	(mIoU)	(F1)	(mIoU)	(F1)
FP32	71.04	81.58	70.55	81.04	67.63	78.91
INT8	70.17	80.85	69.69	80.35	65.95	77.58
Difference	–0.87	–0.73	–0.86	–0.69	–1.68	–1.33

as the model size increased. Although the INT8 version deviates in accuracy compared to FP32 due to low-precision hardware, the loss in accuracy is a small and acceptable tradeoff for the hardware efficiency benefits of INT8.

4.3.1.2 Resource Utilization

The resource utilization of several implemented RECON modules (RACCEL and RS-GDMA) are shown separately in Table 10. In RACCEL_N, the number of DSP slices increase quadratically as N increases. RACCEL_N requires $16N^2$ and $4N$ DSP slices for the convolutional and requantization operations, respectively, for a total of $16N^2+4N$. Furthermore, the number of DSPs is halved when RACCEL is configured for DSP time-multiplexing, for a final total of $\frac{1}{2}(16N^2+4N)$ DSPs. Other resource types, such as LUTs, FFs, BRAM, and CRAM, increase linearly as N increases because these resources are predominately utilized for the N channel-wise datapaths. Furthermore, the application of FG-TMR in RACCEL_N incurs a 3-5× increase in resource utilization compared to $\text{RACCEL}_{N\text{-TMR}}$. For this evaluation, the amount of OCM used for tiling and accumulation is set to 8,192 pixels per channel.

4.3.1.3 Performance and Energy Efficiency

Performance and energy-efficiency, quantified in frames-per-second (FPS) and FPS-per-Watt ($^{\text{FPS}}/_{\text{W}}$), respectively, are measured for several configurations of SegNet executed as software on the SoC CPU or accelerated on RECON. In the software versions, the FPGA

Table 10: RECON module resource utilization.

Device Module	LUTs	FFs	BRAM (36b×1k)	DSPs	CRAM Bits
Z7020	53,200	106,400	140	220	25,636,224
RACCEL_1	2,547 (4.79%)	3,532 (3.32%)	7 (5.00%)	10 (4.55%)	770,172 (3.00%)
RACCEL_2	4,171 (7.84%)	7,057 (6.63%)	21 (15.00%)	36 (16.36%)	1,460,587 (5.70%)
RACCEL_4	8,102 (15.23%)	16,890 (15.87%)	41 (29.29%)	136 (61.82%)	3,294,802 (12.85%)
$\text{RACCEL}_{1\text{-TMR}}$	10,604 (19.93%)	10,939 (10.28%)	21 (15.00%)	30 (13.64%)	2,614,220 (10.20%)
$\text{RACCEL}_{2\text{-TMR}}$	16,844 (31.66%)	21,571 (20.27%)	63 (45.00%)	108 (49.09%)	4,910,929 (19.16%)
RSGDMA_4	4,577 (8.60%)	4,871 (4.58%)	21 (15.00%)	0 (0.00%)	1,166,741 (4.55%)
$\text{RSGDMA}_{4\text{-TMR}}$	22,919 (43.08%)	14,593 (13.72%)	63 (45.00%)	0 (0.00%)	4,728,197 (18.44%)
ZU3EG	70,560	141,120	216	360	30,834,336
RACCEL_1	2,808 (3.98%)	4,040 (2.86%)	7 (3.24%)	10 (2.78%)	1,256,868 (4.08%)
RACCEL_2	4,435 (6.29%)	7,814 (5.54%)	21 (9.72%)	36 (10.00%)	2,393,780 (7.76%)
RACCEL_4	8,329 (11.80%)	17,855 (16.65%)	41 (18.98%)	136 (37.78%)	5,309,913 (17.22%)
$\text{RACCEL}_{1\text{-TMR}}$	11,321 (16.04%)	11,674 (8.27%)	21 (9.72%)	30 (8.33%)	4,344,837 (14.10%)
$\text{RACCEL}_{2\text{-TMR}}$	17,562 (24.89%)	23,100 (16.37%)	63 (29.17%)	108 (30.00%)	7,302,230 (23.68%)
RSGDMA_4	6,025 (8.54%)	6,523 (4.62%)	41 (18.98%)	0 (0.00%)	2,926,567 (9.49%)
$\text{RSGDMA}_{4\text{-TMR}}$	26,379 (37.39%)	19,452 (13.78%)	123 (56.94%)	0 (0.00%)	10,630,561 (34.48%)

Table 11: RECON performance and energy-efficiency.

Platform	Configuration	Performance (FPS)			Power	Performance/Watt (FPS/W)		
Version		Net_A	Net_B	Net_C	(W)	Net_A	Net_B	Net_C
PYNQ-Z2 (Z7020)								
Software	650 MHz; 1 Thread	0.005	0.018	0.065	0.520	0.009	0.035	0.126
Software	650 MHz; 2 Threads	0.010	0.036	0.127	0.730	0.013	0.050	0.174
RECON_1	250.00/500.00 MHz	0.211	0.825	3.171	1.285	0.164	0.642	2.468
$\text{RECON}_{1\text{-TMR}}$	200.00/400.00 MHz	0.169	0.660	2.537	2.480	0.068	0.266	1.023
RECON_2	250.00/500.00 MHz	0.653	2.528	9.436	1.800	0.363	1.405	5.242
$\text{RECON}_{2\text{-TMR}}$	142.85/285.70 MHz	0.412	1.606	6.117	3.155	0.131	0.509	1.939
RECON_4	200.00/400.00 MHz	1.117	4.256	15.472	2.065	0.541	2.061	7.492
Ultra96-V2 (ZU3EG)								
Software	1.2 GHz; 1 Thread	0.011	0.041	0.147	0.310	0.016	0.059	0.211
Software	1.2 GHz; 2 Threads	0.023	0.086	0.302	0.620	0.037	0.138	0.487
Software	1.2 GHz; 4 Threads	0.042	0.155	0.546	1.060	0.040	0.146	0.515
RECON_1	375.00/750.00 MHz	0.316	1.240	4.773	1.205	0.262	1.029	3.961
$\text{RECON}_{1\text{-TMR}}$	375.00/750.00 MHz	0.316	1.240	4.773	4.100	0.077	0.302	1.164
RECON_2	375.00/750.00 MHz	1.033	4.002	15.022	1.730	0.597	2.313	8.684
$\text{RECON}_{2\text{-TMR}}$	300.00/600.00 MHz	0.938	3.655	13.877	4.935	0.190	0.741	2.812
RECON_4	375.00/750.00 MHz	2.101	7.970	28.837	2.440	0.861	3.267	11.818

is kept blank (unprogrammed) to assume a CPU-only system. The software version uses INT8 quantization, Winograd convolution, BatchNorm folding, and compilation with optimizations (`-O3`) and OpenMP for shared-memory multiprocessing. Table 11 shows the performance and energy-efficiency measurements. In all situations, RECON outperforms the software versions by up to three orders of magnitude depending upon the model shape and system configuration. In RACCEL$_N$, performance increases quadratically as N increases because the number of PEs is scaled quadratically. To maintain this quadratic relationship, the memory bandwidth must increase linearly as N (number of channels) increases; otherwise, once saturated, the performance of RACCEL$_N$ begins to increase linearly.

Using a power meter, the board power was measured when idle (i.e., CPU is not busy, and FPGA is blank) and active (i.e., continuously executing convolutional layers) to determine the dynamic power consumption. The idle power was measured at 1.97W and 5.20W for the PYNQ-Z2 and Ultra96-V2 platforms, respectively. Although RECON often has higher peak power consumption, the substantially increased performance leads to significant improvements in energy efficiency, up to two orders of magnitude compared to the software versions. To accommodate space applications with stricter power requirements, the FPGA operating frequency and RECON configuration can be reduced at the cost of decreased performance.

4.3.2 Dependability Evaluation

This section describes the dependability evaluation of RECON. Both CRAM fault injection and neutron irradiation are performed to observe the architectural response of the SegNet model accelerated on RECON to both injected and neutron-induced faults. These experiments quantify and analyze the AVF, MWTF, and neutron cross-section of multiple configurations of RECON modules to both SDC and hangs.

To evaluate our selective and adaptive approaches, we perform CRAM fault injection and use the methodology for evaluating adaptive systems in near-Earth radiation environments, described in Section 2.4.2.3, for three orbital case studies, including the Jason-3 in LEO, NOAA-20 in SSO, and Molniya 1-88 in HEO. The selected orbital case studies represent the dynamic radiation environment of three distinct orbital regimes to demonstrate the

versatility of RECON. Spacecraft in GEO experience minimal fluctuation in SEE rates due to their low susceptibility to trapped protons that are present at very low energy levels at GEO altitude. Consequently, with minimal predictable variation in the GEO radiation environment, our adaptive approach that is based on the dynamics of the near-Earth radiation environment is not applicable for GEO and is therefore not included in our analysis.

Our evaluation includes the following steps. First, we use the fault-injection results and analyze the impact of CRAM faults on the inference accuracy and adjust the AVF to account for SDC_T. Next, we use a combination of state-of-the-art models to predict the time-varying SEE rates of the Z7020 for each orbital case study. Next, using the resource utilization and AVF results, we scale the time-varying SEE rates to approximate the time-varying fault rates of multiple RECON modules on the Z7020 for each orbital case study. Finally, using the time-varying fault rates of RECON modules, repair rates for the recovery mechanisms in RECON, and reward rates (performance and energy-efficiency), we create a phased-mission system model to calculate the instantaneous and average availability, failure rate, and performability. By analyzing this phased-mission system model for several static and adaptive strategies at varied threshold parameters, a design tradespace in terms of availability and performability (FPS and $^{FPS}/_{W}$) is generated with a Pareto-optimal set for selecting the best strategy subject to some user-defined availability constraint.

4.3.2.1 CRAM Fault-Injection Experiment

CRAM fault injection was performed to observe the architectural response of the SegNet model accelerated on RECON to injected faults. In our fault-injection experiment, we evaluate several configurations of the static RSGDMA and reconfigurable RACCEL modules. The RSGDMA modules include $RSGDMA_4$ and $RSGDMA_{4\text{-}TMR}$, and the RACCEL modules include $RACCEL_1$, $RACCEL_2$, $RACCEL_4$, $RACCEL_{1\text{-}TMR}$, and $RACCEL_{2\text{-}TMR}$. FG-TMR is applied using the BL-TMR tool [37]. All RECON modules have tradeoffs in performance, energy-efficiency, and dependability.

CRAM fault injection is performed to quantify the susceptibility of each RECON module to injected CRAM faults in terms of the AVF and MWTF. Two experiments are performed to analyze RECON at the *model-level* and *layer-level*. In the model-level experiment, CRAM faults are present during the execution of the entire model, and in the layer-level experiment, CRAM faults are present only during the execution of one selected layer. In the model-level experiment, each iteration begins with the system in a clean state (i.e., FPGA is fully reprogrammed) to remove any latent faults from preceding iterations, and the input image and CRAM bit location (frame address, word, and bit) are both randomly selected using the Linux system call `getrandom()`. The input image is varied to eliminate any potential bias with the input to the model. Next, the fault is injected into the randomly selected CRAM bit, and the model is fully executed to completion. Finally, the execution event is recorded. In the layer-level experiment, the input image, CRAM bit location, and layer are all randomly selected. Next, the model is fully executed with the execution halted immediately prior to the randomly selected layer to inject the fault and after to repair the fault, thus isolated the fault to the randomly selected layer. Finally, the execution event is recorded.

The execution will either complete correctly, complete with SDC, or hang. SDC is detected if the mIoU, F1, or checksum of the output does not match that of the golden output for the randomly selected image. The mIoU and F1 are also used to analyze the impact of CRAM faults on the inference accuracy and to classify events as SDC_T and SDC_C. A hang is detected when RECON fails to fully execute the model within the expected timeout interval (2 seconds). Finally, all events (correct, SDC, and hang) are recorded and the system is reset into a clean state for the subsequent iteration.

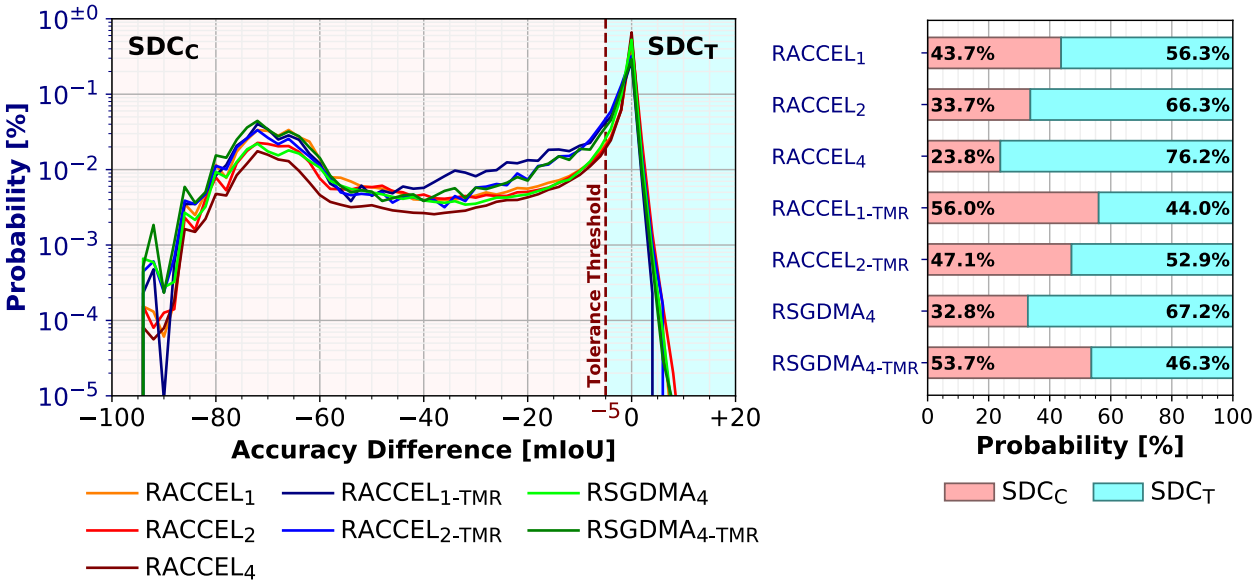
Fault injection is performed using the PCAP. A frame-readback command is issued to the PCAP to retrieve the contents of the frame containing the selected CRAM bit into a software buffer. The selected CRAM bit is inverted in the buffered frame, and a frame-writeback command is issued to the PCAP to write the faulty, buffered frame back to CRAM to complete the fault injection. To minimize uncertainty in the measurements, a significant number of fault injections, which will vary between designs, is performed to minimize the 95% CI error. To accelerate this process, the Xilinx design tools are used to generate a list of

Table 12: RECON model-level CRAM fault-injection test results on PYNQ-Z2 (Z7020).

Module	Injections	AVF (%)			Critical CRAM Bits ±95% CI Error		
		SDC_T	SDC_C	Hangs	SDC_T	SDC_C	Hangs
$RACCEL_1$	2,459,068	16.14	12.55	6.13	124,292 ± 387	96,622 ±341	47,238 ±238
$RACCEL_{1\text{-TMR}}$	6,936,806	0.09	0.11	0.06	2,336 ± 58	2,978 ± 66	1,490 ± 46
Improvement					53.2×	32.5×	31.7×
$RACCEL_2$	4,851,293	23.38	11.86	3.43	341,524 ± 628	173,268 ±448	50,096 ±241
$RACCEL_{2\text{-TMR}}$	9,876,758	0.14	0.13	0.05	6,889 ± 115	6,144 ±108	2,260 ± 66
Improvement					49.6×	28.2×	22.2×
$RACCEL_4$	8,819,211	33.02	10.33	1.85	1,087,875 ±1,250	340,388 ±699	61,107 ±296
$RSGDMA_4$	2,773,146	13.75	6.73	10.03	160,463 ± 509	78,475 ±356	117,015 ±435
$RSGDMA_{4\text{-TMR}}$	4,573,435	0.17	0.19	0.04	7,961 ± 178	9,216 ±191	2,102 ± 91
Improvement					20.2×	8.5×	55.7×

essential CRAM bits, which are CRAM bits actively used by the design, to target exclusively [47]. Furthermore, several PYNQ-Z2 boards are deployed to parallelize the fault-injection campaign.

Table 12 shows the results of the model-level fault-injection experiment including (1) the number of SDC_T, SDC_C, and hang events, (2) the measured AVF for SDC_C and hang events, and (3) the approximated number of critical CRAM bits (AVF × number of essential bits) with 95% CI error vulnerable to SDC_C and hang events of each tested module. As shown in Table 12, the static $RSGDMA_4$ module, which performs most of the control-flow operations in RECON, has the most critical bits vulnerable to hangs and is the biggest contributor to system downtime. Because a hang of the RSGDMA requires a slow, disruptive process to repair the module, SEE mitigation must be selectively applied to the RSGDMA module to minimize the critical area vulnerable to hangs and the associated downtime. The $RSGDMA_{4\text{-TMR}}$ module, which is protected by FG-TMR, substantially reduces the critical area vulnerable to hangs by 56× at the expense of a 3-5× increase in area (Table 10) and 10% decrease in energy-efficiency. Similarly, the application of FG-TMR substantially reduces the



(a) Histogram of probability distribution (b) SDC_T and SDC_C probability

Figure 24: Impact of CRAM faults on mIoU with (a) the probability distribution of mIoU difference in SDC events and (b) probability of SDC_T and SDC_C by RECON module. Multiple, overlaid histograms represented as line plots. 60 bins with widths of 2% mIoU loss per bin.

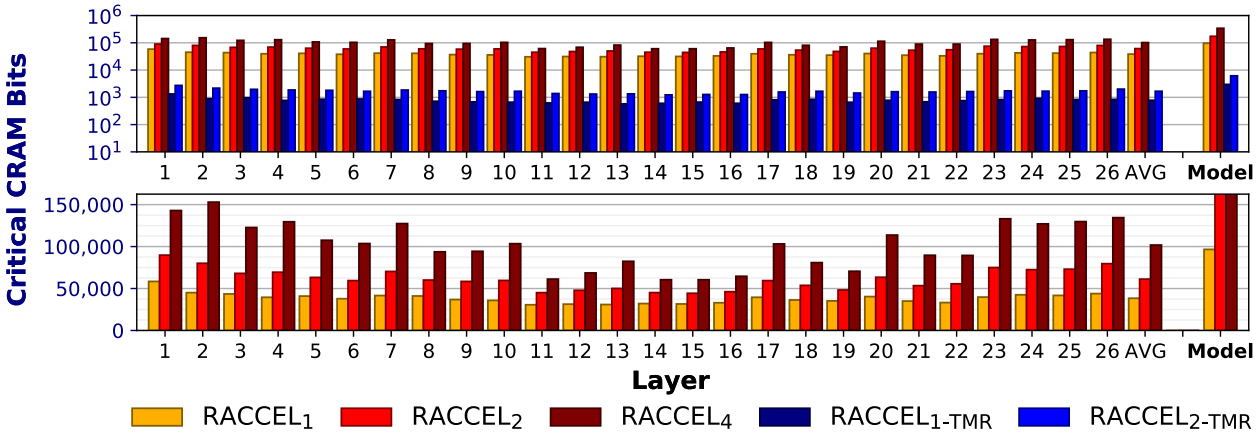


Figure 25: Critical CRAM bits vulnerable to SDC_C by layer and model for each RACCEL configuration in (top) logarithmic and (bottom) linear scale.

critical area of both $\text{RACCEL}_{\text{1-TMR}}$ and $\text{RACCEL}_{\text{2-TMR}}$ compared with their unmitigated counterparts in terms of SDC and hangs, also at the expense of increased area and energy overhead.

Figure 24(a) illustrates a histogram that shows the impact of model-level CRAM faults on the inference accuracy (mIoU) of SDC events. The impact is quantified by the mIoU difference between each SDC output and its corresponding golden output, and the histogram shows the distribution of accuracy differences across all RECON modules. SDC events vary broadly. Due to the inherent fault tolerance of CNNs, the accuracy difference of SDC events is most frequently near the golden mIoU (i.e., at the peak with an accuracy difference of 0%). In our fault-injection campaign, most input images had a golden mIoU between 60%-80%, which limits the maximum mIoU loss due to SDC to this range. The worst-case SDC events with near-zero mIoU (i.e., at the hump with an accuracy difference of -80%--60%) are relatively more frequent than intermediate SDC events between the worst-case and near-zero loss. Few SDC events resulted in a considerably improved mIoU greater than the golden mIoU (i.e., accuracy difference greater than 0%). For this evaluation, we assume a tolerance threshold of -5% mIoU, where a loss in accuracy $\geq -5\%$ mIoU is considered acceptable.

Using this tolerance threshold, Figure 24(b) illustrates the probabilities of SDC events being either SDC_{T} or SDC_{C} for each module. For both RACCEL_N and $\text{RACCEL}_{N\text{-TMR}}$ modules, as N increases, the SDC severity (i.e., probability of SDC_{C}) decreases, possibly due to the ratio of functional and faulty channels. For example, if RACCEL_1 has one faulty channel, then all input FMs flow through the faulty channel, but if RACCEL_2 has one faulty channel, then only half of all input FMs flow through the faulty channel. For all tested modules, the mitigated modules ($\text{RACCEL}_{N\text{-TMR}}$) have greater SDC severity compared to their unmitigated counterparts (RACCEL_N); however, the overall reduced critical area negates this increase. For example, $\text{RACCEL}_{\text{2-TMR}}$ has greater severity than RACCEL_2 (47% versus 34%) but a 37× reduction in critical CRAM bits vulnerable to SDC_{C}.

Figure 25 illustrates the number of critical bits vulnerable to SDC_{C} by layer and model due to layer-level and model-level CRAM faults, respectively. In both mitigated and unmitigated RACCEL_N, across all layers and the full model, the critical area vulnerable to SDC_C increases as N increases. For all RACCEL modules, outer layers had greater critical

area compared to inner layers. This architectural response is possibly due to outer layers operating on FMs with larger spatial resolution, partitioned into tiles constrained by OCM size, but less dimensionality compared to the FMs of the inner layers. The last layers of the encoder blocks (layers 2, 4, 7, 10, and 13) and the first layers of the decoder blocks (layers 14, 17, 20, 24, and 25) tend to have greater critical area compared to other layers within their respective encoder/decoder blocks. This architectural response is probably due to the adjacent max-pooling postprocess and max-unpooling preprocess, which are executed as part of the pipeline due to the layer-fusion optimization. Consequently, because additional circuits are enabled to execute these processes, CRAM faults in these circuits can also manifest into errors.

4.3.2.2 Time-Varying Fault Rate Prediction

Using the SEE rate prediction methodology of [87], the time-varying SEE rates are predicted for the Zynq-7000 for each orbital case study during the first week of 2020. The SEE characterizations of the Zynq-7000 are used, which model the SEE susceptibility of each resource type of the FPGA subsystem in the Zynq-7000 to protons and heavy ions. Solar-minimum conditions and 100 mils of spherical, aluminum shielding are assumed for a worst-case evaluation of each orbit. Next, using Equation (2.4), the time-varying fault rates due to SDC_T, SDC_C, and hangs are determined for each RECON module. For this evaluation, the resource utilization and AVF are used to scale the SEE rates of all resource types, which are then summed to produce the time-varying fault rates. Figure 26 illustrates the time-varying fault rates of multiple RECON modules to SDC_C. The average fault rates for unmitigated RECON modules are orders of magnitude greater than their TMR counterparts (100-1000× for RACCEL modules and 10-100× for RSGDMA modules). Furthermore, for all three orbital case studies, the fault rates are often within the lower 1% of the expected range of fault rates (i.e., between extrema of SEE rates during the first week of 2020) for most of the orbital period, with periodic, short-term worst-case SEE rates. Although hardware-redundancy techniques such as TMR can improve dependability substantially, especially during high SEE rates, these methods are excessive for most of the orbital period, and the

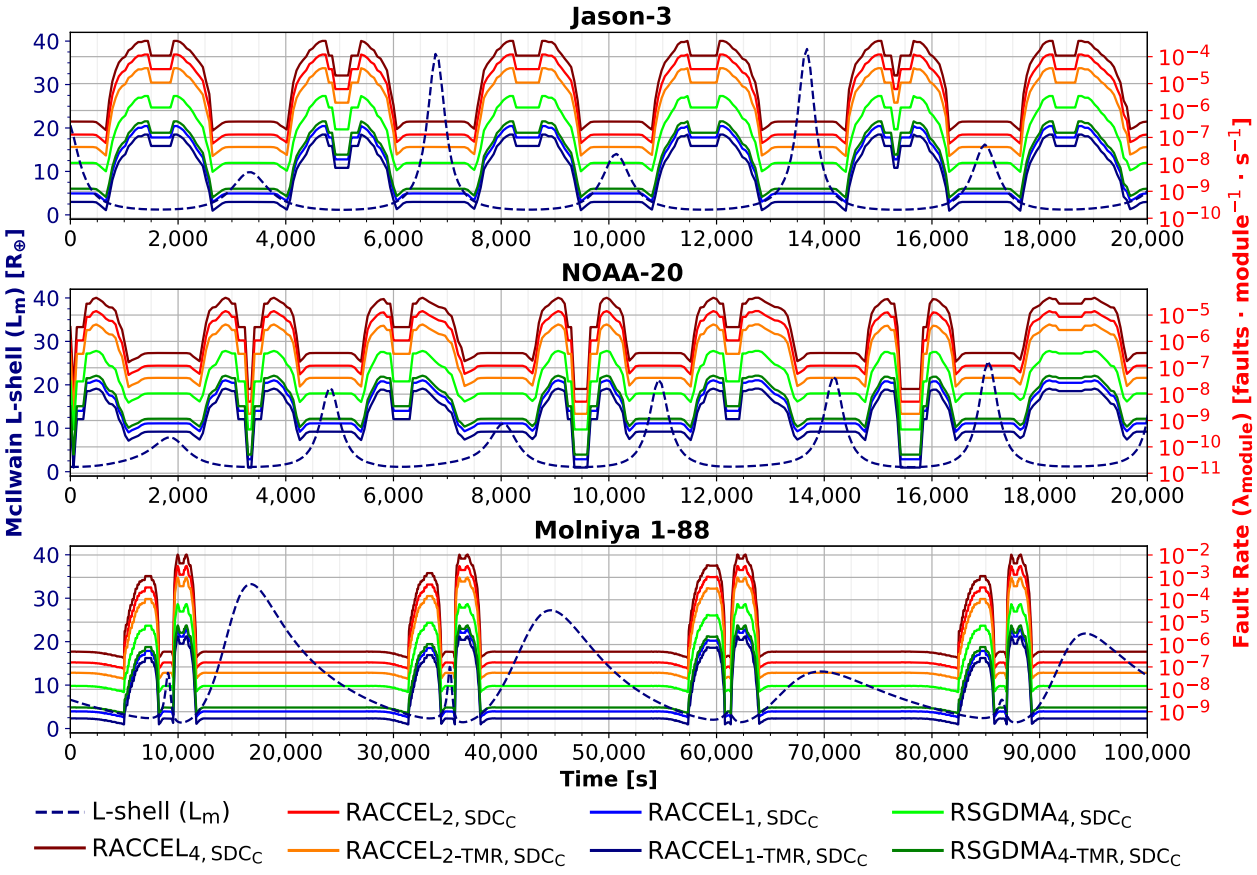


Figure 26: Predicted McIlwain L-shell (L_m) and fault rates (λ_{module}) of multiple RECON modules for Z7020 over time for the Jason-3, NOAA-20, and Molniya 1-88 orbital case studies.

resources could instead be used to improve performance and energy efficiency. By using an environmentally adaptive approach for SEE mitigation, the system can repurpose resources at runtime to improve performance while providing SEE mitigation that is sufficient to the immediate environmental condition.

4.3.2.3 Phased-Mission System Modeling and Analysis

RECON adapts to the environment by configuring the system into one of several static modes, each with its own performance and dependability characteristics, in response to the environmental condition. Table 13 lists the static modes, including the performance, energy-efficiency, and MWTF tradeoffs of each one, and shows the adaptive strategy, a threshold-based approach, used in this evaluation. *Static strategies* use only one mode during the evaluation period. *Adaptive strategies* (denoted as N-Mode) adapt between N different modes during the evaluation period, and adaptation is invoked whenever the device SEE rate crosses any user-defined thresholds. Depending upon the thresholds and fluctuating SEE rate, the adaptive strategies attain some combination of the availability, failure rate, and performability characteristics of each of the modes in use.

The RECON architecture and adaptive behavior are modeled as a CTMC-based phased-mission system model. Each mode is independently modeled as a CTMC, and all CTMCs are interconnected with phase transitions to model the transition between modes as RECON adapts. For each CTMC, all SEEs causing SDC are correctable by CRAM scrubbing (repair rate μ_{Scrub}), RACCEL SDC and hangs are recoverable by PR (repair rate μ_{PR}), and RSGDMA hangs are recoverable by an external watchdog system reset (repair rate μ_{WDT}). The time-varying module fault rates (fault-rate transitions), module repair rates (repair-rate transitions), and performance and energy-efficiency (reward rates) are assigned to the model at runtime. A transient analysis of the phased-mission system model is performed for each static and adaptive strategy using 60-second intervals over a one-week period. At each timestep, the fault-rate transitions are updated and, if the device SEE rate crosses any thresholds, the active CTMC is changed to reflect the new operating mode using the phase transitions.

Table 13: RECON static modes and adaptive strategy.

Static Mode	Configuration	Performance	Energy Efficiency	SDC_C MWTF	
		(FPS)	($^{FPS}/_{W}$)	(FPS)	($^{FPS}/_{W}$)
$Mode_0$	$RSGDMA_{4\text{-}TMR}/RACCEL_4$	15.472	5.545	82.06	14.80
$Mode_1$	$RSGDMA_{4\text{-}TMR}/RACCEL_2$	8.006	3.550	41.39	11.66
$Mode_2$	$RSGDMA_{4\text{-}TMR}/RACCEL_1$	3.171	1.639	16.48	10.05
$Mode_3$	$RSGDMA_{4\text{-}TMR}/RACCEL_{2\text{-}TMR}$	6.117	1.939	4,873.72	2,513.52
$Mode_4$	$RSGDMA_{4\text{-}TMR}/RACCEL_{1\text{-}TMR}$	2.537	1.023	2,221.65	2,171.71

Adaptive Strategy[a]

$$N\text{-Mode}_{1,2,3,\ldots,N}(\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_{N-1})(t) = \begin{cases} \text{Mode}_1, & \text{if } \lambda_{\text{SEE}}(t) < \alpha_1 \\ \text{Mode}_2, & \text{if } \lambda_{\text{SEE}}(t) \in [\alpha_1, \alpha_2) \\ \text{Mode}_3, & \text{if } \lambda_{\text{SEE}}(t) \in [\alpha_2, \alpha_3) \\ \quad \vdots & \\ \text{Mode}_N, & \text{if } \lambda_{\text{SEE}}(t) \geq \alpha_{N-1} \end{cases}$$

[a]Threshold parameters $\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_{N-1} \in [\lambda_{\min}, \lambda_{\max}]$, where $\alpha_1 \leq \alpha_2 \leq \alpha_3 \leq \ldots \leq \alpha_{N-1}$, and $[\lambda_{\min}, \lambda_{\max}]$ are the extrema of the expected range of SEE rates.

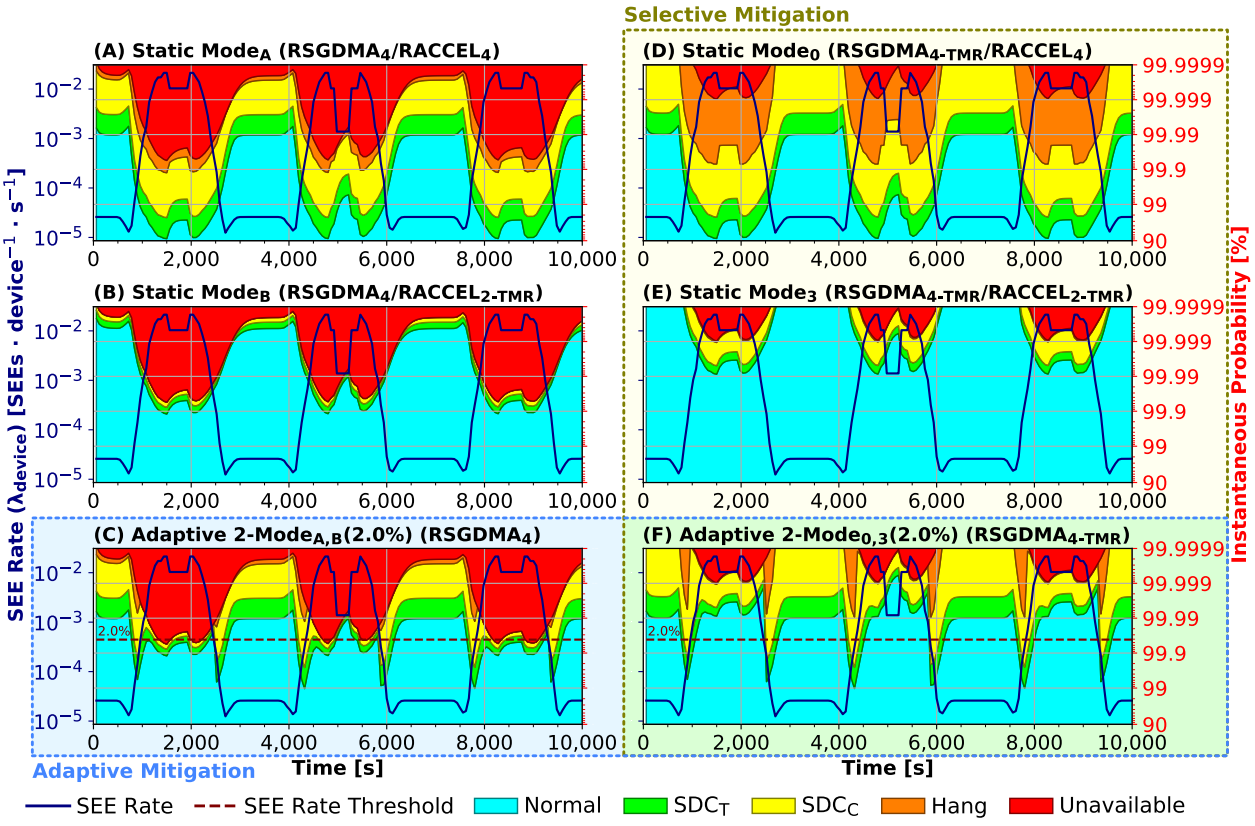


Figure 27: Instantaneous probability of system operation for six strategies in Jason-3 orbit over time. Strategies D, E, and F (right column) use selective mitigation and strategies C and F (bottom row) use adaptive mitigation.

Figure 27 illustrates several static and adaptive strategies of RECON with the instantaneous probability of the system operating in normal, SDC_T, SDC_C, hung (application is nonoperational), or unavailable (system is nonoperational) states over time. First, we examine the effect of selective mitigation in RECON. System availability is predominately affected by the vulnerability of the RSGDMA to hangs. Since selective mitigation is specific to the RSGDMA, changing RACCEL has minimal impact on system availability. Strategies D ($Mode_0$), E ($Mode_3$), and F (2-$Mode_{0,3}$(2.0%)) use selective mitigation (i.e., use $RSGDMA_{4\text{-}TMR}$), and strategies A ($Mode_A$), B ($Mode_B$), and C (2-$Mode_{A,B}$(2.0%)) mirror strategies D, E, and F, respectively, but omit selective mitigation (i.e., use $RSGDMA_4$). With selective mitigation, strategies D, E, and F have substantially lower system unavailability due to the reduced vulnerability of the RSGDMA to hangs (represented by less probability area of unavailability in Figure 27).

Next, we compare static versus adaptive strategies. Strategies D and E are static strategies tuned for performance and dependability, respectively, and strategy F is an adaptive strategy that adapts between D and E when the immediate SEE rate crosses the 2.0% threshold within the expected range of SEE rates. As a result, strategy F, which adapts between D and E, attains the static performability and availability characteristics of D and E when the SEE rate is below or above the threshold, respectively, with transients during the adaptation events. With adaptive mitigation, strategy F is beneficial when the mission availability constraint is between the availability of D and E because the threshold can be adjusted to select E to sufficiently satisfy that constraint and to select D for the remainder of the period to maximize performability. However, in strategy C, which adapts between A and B, the advantage of adaptive mitigation is negated by the high system unavailability due to the omission of selective mitigation. Therefore, the combination of both selective and adaptive approaches, as demonstrated by strategy F, is essential to minimize system unavailability due to RSGDMA hangs and to enable system adaptation to repurpose resources to maximize performability subject to availability constraints.

By analyzing this phased-mission system model for several static and adaptive strategies at varied threshold parameters, a design tradespace in terms of availability and performability is generated for each orbital case study. Figure 28 shows the design tradespace with

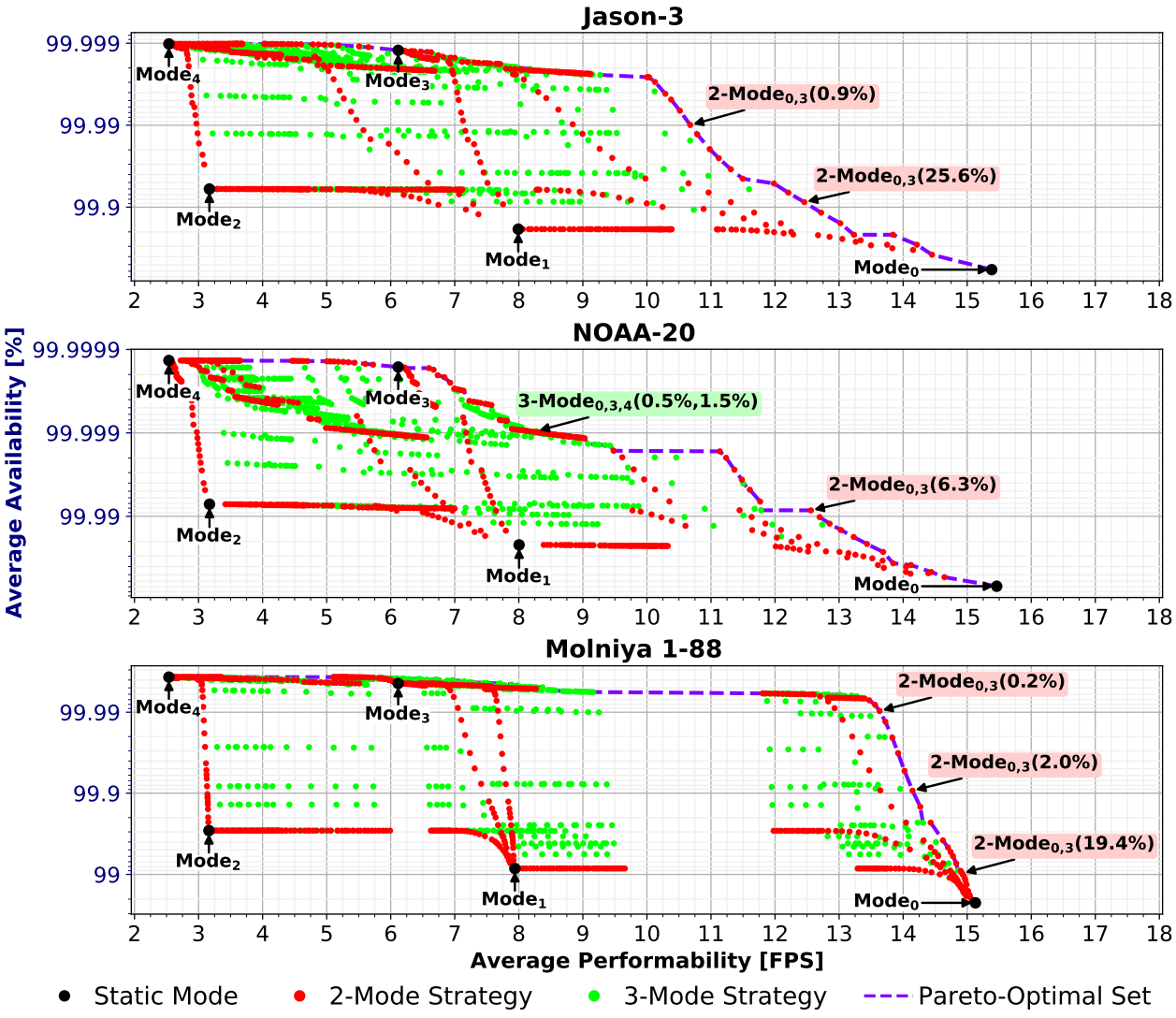


Figure 28: RECON design tradespace in terms of performability and availability and with Pareto-optimal curves for static and adaptive strategies.

Table 14: RECON unavailability, failure rate, and performability for orbital case studies.

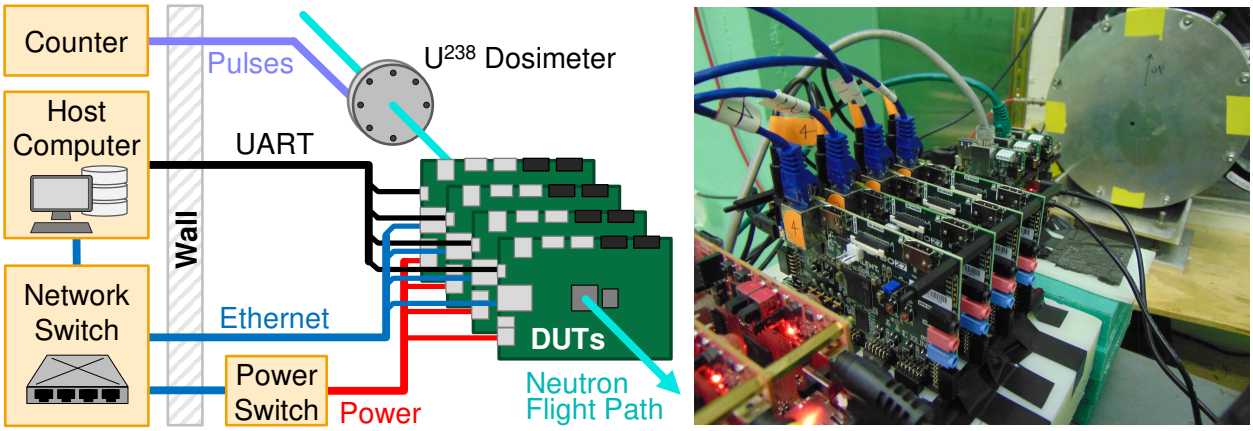
Orbit Strategy	Availability Constraint[a] (%)	Average Unavailability (days · yr^{-1})	Average Failure Rate (failures · day^{-1})	Average Performability (FPS)	Average Performability (FPS/W)
Jason-3					
Mode_0	≥99	2.10×10^{0}	9.66×10^{0}	15.38	5.51
Mode_1	≥99	6.76×10^{-1}	3.30×10^{0}	7.99	3.54
Mode_2	≥99.9	2.19×10^{-1}	1.19×10^{0}	3.17	1.64
Mode_3	≥99.99	4.45×10^{-3}	1.74×10^{-2}	6.12	1.94
Mode_4	≥99.99	3.72×10^{-3}	1.39×10^{-2}	2.54	1.02
2-$\text{Mode}_{0,3}$(25.6%)	≥99.9	3.17×10^{-1}	1.44×10^{0}	12.46 (2.04× Mode_3)	4.38 (2.26× Mode_3)
2-$\text{Mode}_{0,3}$(0.9%)	≥99.99	3.63×10^{-2}	1.61×10^{-1}	10.68 (1.75× Mode_3)	3.70 (1.91× Mode_3)
2-$\text{Mode}_{1,4}$(80.9%)	≥99.9	3.36×10^{-1}	1.63×10^{0}	7.29 (2.30× Mode_2)	3.22 (1.97× Mode_2)
2-$\text{Mode}_{1,4}$(3.1%)	≥99.99	3.14×10^{-2}	1.48×10^{-1}	5.46 (2.15× Mode_4)	2.38 (2.32× Mode_4)
NOAA-20					
Mode_0	≥99.9	2.51×10^{-1}	1.14×10^{0}	15.46	5.54
Mode_1	≥99.9	8.04×10^{-2}	3.90×10^{-1}	8.00	3.55
Mode_2	≥99.99	2.62×10^{-2}	1.41×10^{-1}	3.17	1.64
Mode_3	≥99.999	5.93×10^{-4}	2.31×10^{-3}	6.12	1.94
Mode_4	≥99.999	4.95×10^{-4}	1.85×10^{-3}	2.54	1.02
2-$\text{Mode}_{0,3}$(6.3%)	≥99.99	3.10×10^{-2}	1.40×10^{-1}	12.56 (2.05× Mode_3)	4.42 (2.28× Mode_3)
3-$\text{Mode}_{0,3,4}$(0.5%,1.5%)	≥99.999	3.48×10^{-3}	1.60×10^{-2}	8.27 (1.35× Mode_3)	3.12 (1.61× Mode_3)
2-$\text{Mode}_{1,4}$(92.6%)	≥99.99	3.13×10^{-2}	1.51×10^{-1}	6.96 (2.20× Mode_2)	3.07 (1.87× Mode_2)
2-$\text{Mode}_{1,4}$(2.0%)	≥99.999	3.55×10^{-3}	1.67×10^{-2}	5.61 (2.21× Mode_4)	2.44 (2.39× Mode_4)
Molniya 1-88					
Mode_0	≥90	8.09×10^{0}	4.98×10^{1}	15.13	5.42
Mode_1	≥99	3.08×10^{0}	1.69×10^{1}	7.94	3.52
Mode_2	≥99	1.06×10^{0}	5.98×10^{0}	3.16	1.63
Mode_3	≥99.99	1.61×10^{-2}	6.28×10^{-2}	6.12	1.94
Mode_4	≥99.99	1.34×10^{-2}	5.03×10^{-2}	2.54	1.02
2-$\text{Mode}_{0,3}$(19.4%)	≥99	3.54×10^{0}	1.81×10^{1}	14.92 (1.88× Mode_1)	5.34 (1.52× Mode_1)
2-$\text{Mode}_{0,3}$(2.0%)	≥99.9	3.40×10^{-1}	1.55×10^{0}	14.15 (2.31× Mode_3)	5.03 (2.60× Mode_3)
2-$\text{Mode}_{0,3}$(0.2%)	≥99.99	3.52×10^{-2}	1.49×10^{-1}	13.63 (2.23× Mode_3)	4.84 (2.49× Mode_3)
2-$\text{Mode}_{1,4}$(3.7%)	≥99.9	2.80×10^{-1}	1.36×10^{0}	7.41 (2.92× Mode_4)	3.28 (3.20× Mode_4)
2-$\text{Mode}_{1,4}$(0.6%)	≥99.99	3.24×10^{-2}	1.42×10^{-1}	7.05 (2.78× Mode_4)	3.11 (3.04× Mode_4)

[a]RECON availability includes normal and SDC_T operation.

the Pareto-optimal set, which can be used to identify the optimal design that achieves most of one trade subject to the constraint of another trade. In this context, for a given availability constraint, a strategy is optimal if it satisfies that constraint and achieves the most performability, and vice versa. Table 14 shows the average unavailability, failure rate, and performability of several static and adaptive strategies subject to select availability constraints (orders of nine) for each orbital case study. The performability improvement of an adaptive strategy is measured by comparing that strategy to the best performing static strategy that satisfies the same availability constraint. For example, in the Jason-3 case study, both Mode_3 and 2-$\text{Mode}_{0,3}$(0.9%) satisfy the availability constraint of ≥99.99% (four nines), but 2-$\text{Mode}_{0,3}$(0.9%) has a performability improvement in performance (1.75×) and energy-efficiency (1.91×) over Mode_3. As another example with low-area constraints for the PRR, in the Molniya 1-88 case study, both Mode_4 and 2-$\text{Mode}_{1,4}$(3.7%) satisfy the availability constraint of ≥99.9%, but 2-$\text{Mode}_{1,4}$(3.7%) has a performability improvement in performance (2.92×) and energy-efficiency (3.20×) over Mode_4. Depending upon the performance and dependability tradeoffs between the RECON modules in use, the SEE susceptibility of the FPGA device to the radiation characteristics of the orbit, and user-defined parameters (e.g., repair rates, reward rates, and availability constraints), the achievable performability gains can vary. In our evaluation, the optimal adaptive strategies of RECON for the select availability constraints achieved performability improvements of 1.5-3.0× for all orbital case studies.

4.3.2.4 Wide-Spectrum Neutron-Beam Test Experiment

RECON was irradiated under wide-spectrum neutrons at the Los Alamos Neutron Science Center (LANSCE) using the 4FP30R/ICE-II instrument [67] to characterize the susceptibility of the SegNet model accelerated on RECON to neutron-induced SEEs. In this experiment, the neutron cross-section was calculated for two design configurations of RECON: $\text{RSGDMA}_{\text{2-TMR}}/\text{RACCEL}_2$ (Mode_1) and $\text{RSGDMA}_{\text{2-TMR}}/\text{RACCEL}_{\text{1-TMR}}$ (Mode_4). The experimental setup is illustrated in Figure 29. Four Zybo Z7-20 (Z7020) and two UltraZed-EG (ZU3EG) DUTs were placed in the beam to parallelize the fluence each design was exposed



(a) Experiment block diagram

(b) Experiment photo

Figure 29: Experimental setup at LANSCE.

Table 15: RECON wide-spectrum neutron-beam test results.

Platform Design	Effective Fluence ($n \cdot cm^{-2}$)	Total Runs	Execution Events SDC_T	 SDC_C	 Hangs	Cross-Section[a] 95% Confidence Interval SDC_C (cm^2)
Zybo Z7-20 (Z7020)						
$RSGDMA_{2\text{-}TMR}/RACCEL_2$	3.80×10^{11}	158,216	491	129	18	3.69×10^{-10} $[3.06\times10^{-10}, 4.33\times10^{-10}]$
$RSGDMA_{2\text{-}TMR}/RACCEL_{1\text{-}TMR}$	4.17×10^{11}	100,702	48	29	7	7.56×10^{-11} $[4.82\times10^{-11}, 1.05\times10^{-10}]$
Improvement						4.88×
UltraZed-EG (ZU3EG)						
$RSGDMA_{2\text{-}TMR}/RACCEL_2$	1.51×10^{11}	208,128	16	2	3	1.58×10^{-11} $[7.90\times10^{-13}, 2.19\times10^{-11}]$
$RSGDMA_{2\text{-}TMR}/RACCEL_{1\text{-}TMR}$	1.50×10^{11}	135,224	0	0	1	8.01×10^{-12} $[0.00\times10^{-00}, 1.57\times10^{-11}]$
Improvement						1.97×

[a]Assuming one event when no events were detected [76].

to. DUT management software hosted on a separate monitoring laptop was used to automate the logging, monitoring, and power cycling of the DUTs. At boot, DUTs pulled their bootable image via Ethernet. At runtime, DUTs reported execution events via serial UART and transmitted SDC samples to the host via Ethernet. Using a networked power switch, a DUT was power cycled when it became unresponsive (i.e., the DUT failed to signal a heartbeat prior to timeout), reported consecutive SDC or hangs (counted as one), or detected that the CRAM scrubber had failed. For the Zybo Z7-20 DUTs, the DDR memory was configured with ECC enabled and the unified L2 caches were disabled to prevent the high neutron-flux from overwhelming the DUTs and to minimize error modes associated with the CPU and DDR memory. For the UltraZed-EG DUTs, the DDR memory was configured with ECC disabled (not supported) and the caches remained enabled because the Zynq-MPSoC APU caches have high resilience to SEUs [5]. Since the wide-spectrum neutron beam induces an uncontrolled fault rate, the CRAM scrubber was enabled to prevent the accumulation of CRAM faults. However, the neutron beam can expose the DUTs to fault modes that cannot be directly compared or reproduced with our CRAM fault-injection procedure (e.g., multi-bit upsets, CPU or memory faults, or overwhelmed scrubber).

In our radiation-beam test procedure, the DUTs continuously executed the SegNet model using RECON. Checksums were used to validate the correctness of the execution, and correct, SDC, and hang events were recorded with timestamps. The 4FP30R/ICE-II instrument contains a U^{238} dosimeter that recorded the integrated neutron flux (above 10 MeV) with timestamps. The neutron fluence (above 10 MeV) was calculated by integrating the neutron flux over the time interval that the DUTs were active. The designs were alternated between DUTs and the recorded fluence was derated to account for the distance between the DUT and beam source ($r^2/(r+d)^2$ where r is the distance between the dosimeter and beam source and d is the distance between the DUT and dosimeter).

The experimental results are shown in Table 15. For both sets of DUTs, the 5× (Zynq-7000) and 2× (Zynq-MPSoC) improvement in the neutron cross-section reaffirms the dependability advantage of $\text{RACCEL}_{\text{1-TMR}}$ over RACCEL_2 with selective mitigation applied to the RSGDMA; however, with increased overhead in area, performance, and energy-efficiency.

The dissimilarity in the cross-section magnitudes between both sets of DUTs (Zynq-7000 and Zynq-MPSoC) can be attributed to generational differences in both the device architecture and process technology.

4.4 Conclusion

Dependable, high-performance onboard processing is essential for enabling DL applications to enhance spacecraft autonomy, data analysis, and intelligent applications for both science and defense missions. Commercial hybrid and heterogeneous SoCs and systems featuring SRAM-based FPGAs provide several architectural advantages compared to rad-hard processors that can enable the deployment of DL applications for spacecraft systems. However, these commercial devices are highly susceptible to radiation-induced SEEs that can degrade the dependability of the DL application.

In this chapter, we proposed RECON, a runtime-reconfigurable framework for dependable, high-performance semantic segmentation for space applications on FPGAs and hybrid SoCs. RECON leverages several model-compression, algorithmic, and architectural optimization techniques to maximize the inference performance, energy efficiency, and area efficiency for onboard processing. To enhance the dependability of DL applications for the space environment, we proposed selective and adaptive approaches to enable efficient SEE mitigation in RECON. In our selective approach, the control-flow parts of RECON are protected by TMR to minimize SEE-induced hangs, which are slow and disruptive to repair. We demonstrated a 56× reduction in the critical area at the expense of 3-5× increase in area (for control-flow parts only) and 10% decrease in energy-efficiency. In our adaptive approach, we leverage PR to alternate between configurations of the data-flow parts of RECON to adapt the degree of SEE-induced SDC mitigation in response to the fluctuating SEE rates of the dynamic near-Earth space radiation environment. Combined, both approaches enable RECON to maximize performability subject to mission availability constraints. We performed fault injection and neutron irradiation to observe the susceptibility of the SegNet semantic-segmentation model on RECON, and we used dependability modeling to evaluate RECON

in various orbital case studies to demonstrate 1.5-3.0× performability improvements in both performance and energy-efficiency compared to static approaches. Our evaluation, which was conducted on facsimiles of flight hardware that is currently deployed in space missions, demonstrates that compute-intensive DL applications such as semantic segmentation can be dependably executed onboard for next-generation missions.

5.0 Evaluation and Analysis of FPGA-Accelerated, Deep-Learning Apps for Onboard Space Processing

Commercial FPGAs and SoCs provide several architectural advantages suitable for onboard DL acceleration. However, due to the high susceptibility of these commercial devices to radiation-induced SEEs that can degrade the dependability of the DL application, dependability must be considered for systems that use these commercial devices to deploy DL in mission-critical applications. Furthermore, researchers have created a broad variety of DL models for a wide range of applications that can also vary in dependability. Some examples include DL models that perform classification, detection, localization, and segmentation tasks on image data and can be applied to enhance applications in EO and remote sensing (e.g., semantic labeling, image compression, image super-resolution, change detection, and 3D estimation) [7]. However, due to characteristic differences between DL models (e.g., network structure, operations, and trained parameters) and accelerators (e.g., architecture, optimizations, and data-flow), DL solutions can vary broadly in terms of accuracy, resource utilization, performance, energy efficiency, and dependability. All these tradeoffs are crucial for resource-constrained and mission-critical systems. To select an optimal DL solution for a specific task that maximizes inference performance, conserves onboard resources, and satisfies dependability requirements, a methodology is required to evaluate and compare the tradeoffs between competing options.

In this chapter, we propose a methodology for evaluating FPGA-accelerated DL models and analyzing their tradeoffs. With an emphasis on the dependability evaluation, we also propose a hierarchical fault-injection approach to accelerate the characterization of fault susceptibility in DL solutions. In our hierarchical approach, fault injection is performed at multiple levels, in order of decreasing application granularity (coarse to fine), and the targeted area at each level is continually reduced by omitting inconsequential bits of the preceding level, thus substantially reducing the number of fault injections required. Furthermore, we propose analytical methods that use our hierarchical fault-injection approach to quantify and examine FPGA-accelerated DL models in terms of well-established dependability metrics

and to observe the impact of faults on inference accuracy, profile and map vulnerability to node-level operations, and predict design fault rates for near-Earth orbital environments. To demonstrate the versatility of our methodology, we evaluate four semantic-segmentation DL models accelerated on four Xilinx DPU accelerator configurations implemented on two generations of Xilinx SoCs: Zynq-7000 and Zynq-MPSoC.

5.1 Related Work

The dependability of FPGA-accelerated ML models, including methods for evaluation and mitigation, has been explored in the literature [99]. A variety of approaches for evaluating ML dependability using fault injection and radiation-beam testing have also been proposed. In [20], single-bit and multi-bit fault injection were performed in both static and dynamic CRAM to observe the architectural response of a binary NN to both single and multi-bit upsets. In [19, 53], the dependability tradeoffs between mixed-precision float-point and binary quantization data types of a CNN were analyzed. In [9], fault injection and neutron irradiation were performed on multi-layer perceptron with layers assigned to separate FPGA partitions to analyze the ML model at the model and layer levels.

The dependability analysis of FPGA-accelerated ML models has also enabled efficient methods for SEE mitigation. In [52, 25], fault injection was performed to identify the most vulnerable layers and channels, respectively, for selective replication to improve overall dependability with minimal overhead due to replication. In [86], a CNN was disaggregated into a static, replicated control-flow subset and a runtime-reconfigurable data-flow subset that can be exchanged with unmitigated, high-performance and mitigated, low-performance versions of the accelerator. In this chapter, we propose an efficient fault-injection approach to accelerate the evaluation of DL solutions for FPGAs to enable a rapid tradespace analysis between DL models and accelerators for optimal selection, and to quickly identify vulnerable parts of the DL solution for selective or adaptive SEE mitigation.

5.2 Approach

Due to the depth of DL models (up to hundreds of nodes) and area of FPGA accelerators (up to tens of millions of CRAM bits), which can amount to billions of possible fault injections per DL solution, a time-efficient approach is required to accurately and comprehensively quantify the dependability of FPGA-accelerated DL models. In this section, we present an overview of our hierarchical fault-injection approach and describe the low-level details about our CRAM fault-injection process used in our evaluation.

5.2.1 Hierarchical Fault-Injection Approach

Our hierarchical approach involves fault injection at multiple levels, in order of decreasing granularity (e.g., application → phases → subphases), and focuses on continually narrowing the size of targeted CRAM between levels by omitting bits with noncritical representation in the preceding level. At each level transition, the subset of tested CRAM that manifests into observable events becomes the target CRAM for the subsequent level, and the remaining bits (noncritical and untested) are omitted. The continual omission of inconsequential bits can substantially reduce the number of fault injections required to analyze an FPGA-accelerated application at low levels of granularity. For our evaluation of FPGA-accelerated DL models, two levels are sufficient to evaluate a DL model at the model and node levels. The approach is illustrated in Figure 30.

Initially, the set of targeted CRAM must be determined prior to fault injection. The Xilinx design tools can be used to generate the set of essential CRAM bits (essential area), which refers to the subset of CRAM that is actively used by the design. Since nonessential bits do not affect the design, these bits are omitted. Furthermore, to evaluate the DL accelerator exclusively, only the subset of the essential area associated with the partial design is to be targeted. To generate the essential area of a partial design, the Xilinx design tools are used to generate the essential areas from two designs. The first design is the complete design that generates the full essential area. The second design is a post-implementation modification the first design, with all cells and nets associated with the DL accelerator

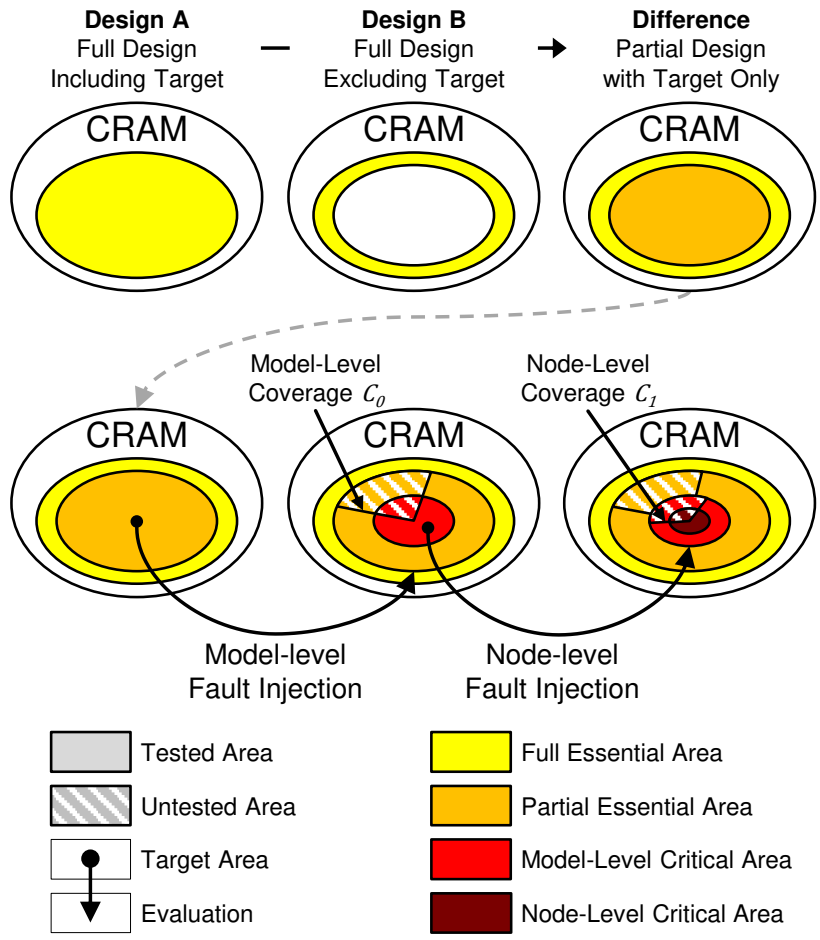


Figure 30: Hierarchical fault-injection approach.

removed, that generates an essential area that excludes the partial design. The difference between both essential areas is the partial essential area (Area_E) that is exclusive to the DL accelerator.

In our hierarchical approach, fault injection is first performed at the model level (highest granularity), where injected faults are present during the execution of the entire model (all nodes). Area_E is the target area, and fault injection is performed to generate the model-level critical area ($\text{Area}_\text{M,C}$), which is the subset of Area_E that is vulnerable to model-level faults that will manifest into observable events. The coverage factor C_0 refers to the fraction of the target area that has been tested. At this level, the model-level dependability metrics AVF_M, MWTF_M, and $\text{Area}_\text{M,C}$ are measured, and $\text{Area}_\text{M,C}$ can be approximated using Equation (5.1).

$$\text{Area}_\text{M,C} = \text{AVF}_\text{M} \times \text{Area}_\text{E} \tag{5.1}$$

Next, fault injection is performed at the node level, where injected faults are present only during the execution of a randomly selected node. $\text{Area}_\text{M,C}$ is the targeted area, and fault injection is performed to generate the node-level critical area ($\text{Area}_\text{N,C}$), which is the subset of $\text{Area}_\text{M,C}$ that is vulnerable for node-level faults that cause observable events. At this level, node-level dependability metrics are measured, and $\text{Area}_\text{N,C}$ can be approximated using Equation (5.2). The coverage factor C_0 inversely scales the $\text{Area}_\text{M,C}$ to account for any untested, critical bits potentially missed in the previous level of fault injection. The accuracy of $\text{Area}_\text{N,C}$ increases when the coverage factor of the model-level process increases ($C_0 \to 1$).

$$\text{Area}_\text{N,C} = \frac{1}{C_0} \times \text{AVF}_\text{N} \times \text{Area}_\text{M,C} \tag{5.2}$$

5.2.2 Fault-Injection Procedure

Initially, a target CRAM bit, input image, and node (node-level process only) are randomly selected. In the model-level process, the fault is injected, then the model is executed completely, and finally, the fault is repaired. In the node-level process, the DL model executed is halted immediately before and after the execution of the randomly selected node to inject and repair a fault, respectively. Fault injection is performed using a CAP device (PCAP or ICAP). First, a frame readback command is issued via the CAP to read a CRAM frame into a software buffer. Next, a bit-flip is performed on the randomly selected bit in the buffered CRAM frame. Finally, a frame writeback command is issued via the CAP to write the fault-injected, buffered CRAM frame back to CRAM. To repair the fault, the same process is repeated on the same CRAM bit.

In our evaluation of the DPU, a custom interface monitor (IM), illustrated in Figure 8, was created to monitor the interrupt and all AXI interfaces of the DPU (one instruction and two data interfaces). Since the DPU interrupt fires at the end of each node operation, the IM can calculate checksums of each interface at the node level. Consequently, the IM can identify errors at the node level, including harmless faults that may corrupt intermediate data but may be masked by a later node (e.g., faulty FMs generated from convolutional nodes masked by later pooling or activation nodes).

At the end of each fault-injection iteration, the execution output and intermediate data are analyzed to classify the outcome. We focus upon two classifications of observable events, including SDC and hangs, and the criticality of the faults. SDC events are erroneous and normally undetectable outcomes of an application execution due to faults. SDC events usually occur when faults affect the data-flow subset of a design (e.g., corrupting datapaths). In our fault-injection procedure, SDC events are observed if the checksum of the predicted output does not equal the checksum of the golden (fault-free) output. SDC events can also be observed if any intermediate checksums generated by the IM are not equal to the checksums of a golden execution. Hang events refer to the nonperformance of the application execution due to faults. Hang events usually occur when faults affect the control-flow subset of a design (e.g., corrupting finite-state machines). In our fault-injection procedure, hang events

are observed if a model execution is preempted to abort by timeout prior to completion. The DPU runtime software was configured to timeout after 3 seconds if the execution did not finish.

Criticality is associated with SDC events and refers to the negative impact a fault has on inference accuracy. The criticality of SDC events can vary broadly, from low criticality (e.g., few bad pixels) to high criticality (e.g., severe distortions). Depending upon mission requirements, SDC with low criticality may be acceptable. SDC events with criticality below a user-specified tolerance threshold can be classified as tolerable SDC (SDC_T), and SDC events above this threshold can be classified as critical SDC (SDC_C). SDC_C events are the primary classification used to analyze dependability. Due to high redundancy in the parameters of a model, DL algorithms have been demonstrated to have an inherent tolerance to faults [99]. Therefore, the SDC criticality is essential for an accurate dependability assessment. When an SDC event is observed, the criticality can be measured as the difference in accuracy between the predicted output and golden output using a standard metric depending upon the DL task (e.g., mIoU or F1 metrics for semantic segmentation).

5.3 Evaluation

In our evaluation, we quantify and analyze the accuracy, area, performance, energy-efficiency, and dependability characteristics of four semantic-segmentation models (ENet, ESPNet, FPN, and U-Net, shown in Table 16) on four configurations of the DPU (B512, B800, B1024, and B1152, shown in Table 17) on two generations of Xilinx SoC platforms: the TUL PYNQ-Z2 (PYNQ-Z2) and UltraZed-EG (UZED-EG), which feature a Z7020 and ZU3EG, respectively. The DL models are based on the Caffe models in [116] and were modified to use the Potsdam dataset [74] in 512×512 RGB image patches. The pixel parallelism (PP), input channel parallelism (ICP), and output channel parallelism (OCP) parameters correspond to the convolution architecture of the DPU, and the peak number of operations per cycle is equal to 2×PP×ICP×OCP. The DPU exposes 64-bit AXI3 and 128-bit AXI4 interfaces for the Zynq-7000 and Zynq-MPSoC, respectively. The DPU v3.1 IP is configured

with one DPU core, low RAM and DSP usage, and extras enabled (channel augmentation, depth-wise convolution, average pool, ReLU, Leaky ReLU, and ReLU6). The trained DL models are quantized and compiled for the DPU using the DNN Development Kit (DNNDK; Vitis AI predecessor) v3.1 flow, with PetaLinux v2019.2 and Vivado v2020.1.

Table 16: DPU DL models.

Model	Number of Nodes	Parameter Size (MB)	Workload MACs (GOps)	I/O Memory Space (MB)
ENet	98	0.36	4.06	3.53
ESPNet	190	0.33	3.71	6.96
FPN	76	5.84	17.30	8.63
U-Net	33	7.40	96.68	40.78

Table 17: DPU convolution architectures.

DPU	PP	ICP	OCP	Peak Ops
B512	4	8	8	512
B800	4	10	10	800
B1024	8	8	8	1024
B1152	4	12	12	1152

5.3.1 Accuracy

The Vitis AI (and DNNDK) compiler quantizes the DL models to use the INT8 data type as a model-compression technique to reduce area, bandwidth, energy, and storage requirements with low-precision integer arithmetic at the cost of reduced accuracy. Table 18 shows the inference accuracy (mIoU and F1) of each evaluated DL model in floating-point FP32 and quantized INT8 forms and the accuracy loss due to quantization. As demonstrated, these efficiency benefits of quantization can be attained with a slight tradeoff in accuracy. The accuracy loss varies by DL model but was less than 2% across all evaluated models.

Table 18: DPU model accuracy.

Model	FP32		INT8		*Difference*	
	(mIoU)	(F1)	(mIoU)	(F1)	(mIoU)	(F1)
ENet	64.9	76.1	63.3	74.8	–1.7	–1.3
ESPNet	56.7	69.4	55.5	68.4	–1.2	–1.0
FPN	65.7	77.1	65.3	76.9	–0.4	–0.3
U-Net	62.1	74.3	61.4	73.8	–0.7	–0.5

5.3.2 Resource Utilization

The FPGA design resource utilization for each DPU is shown in Table 19. The DSP resources, which implement the PEs that form the convolution architecture of the DPU, scale linearly to PP×ICP×OCP and quadratically to the channel parallelism (ICP/OCP parameter). Since the DPU uses the DSP time-multiplexing optimization, the DSPs operate at twice the frequency to halve the number of DSPs required. Observing configurations B512, B800, and B1152, which have fixed PP and varied ICP/OCP, the resource utilization of other resource types (LUTs, FFs, BRAM, and essential CRAM) is approximately linear to the channel parallelism (ICP/OCP parameter).

Table 19: DPU resource utilization.

DPU	LUTs	FFs	BRAM (36b×1k)	DSPs	CRAM Bits
Z7020	53,200	106,400	140	220	25,636,224
B512	48.47%	39.60%	52.14%	35.45%	28.61%
B800	59.57%	50.03%	57.86%	53.18%	36.73%
B1024	64.87%	62.33%	75.00%	70.00%	45.49%
B1152	67.18%	61.39%	80.00%	74.55%	46.93%
ZU3EG	70,560	141,120	216	360	30,834,336
B512	38.24%	24.85%	33.80%	21.67%	35.28%
B800	42.28%	29.89%	37.50%	32.50%	40.94%
B1024	47.81%	35.77%	48.61%	42.78%	47.61%
B1152	45.85%	34.55%	51.85%	45.56%	46.68%

Table 20: Evaluation results for performance, energy-efficiency, and model-level dependability for the Z7020 (PYNQ-Z2).

DPU	Model	Performance			MWTF		AVF (%)				SDC-Critical Area ±95% CI Error	SDC Rates (SDC · dev^{-1} · day^{-1})	
		FPS	FPS/W	Util (%)	FPS	FPS/W	SDC	SDC_T	SDC_C	Hangs		LEO	GEO
B512 250/500MHz	ENet	13.2	4.9	41.9	63.4	23.6	31.0	17.2	13.8	2.8	2,270,773 ± 3,954	0.19	0.11
	ESPNet	5.5	2.3	15.9	18.6	7.8	31.4	12.0	19.3	3.2	2,299,408 ± 4,180	0.19	0.11
	FPN	6.0	2.1	80.5	33.4	12.0	27.9	15.5	12.4	2.7	2,045,829 ± 4,227	0.17	0.10
	U-Net	1.2	0.5	88.3	5.9	2.6	28.8	15.4	13.5	2.7	2,114,796 ± 4,466	0.18	0.10
	Average	6.5	2.5	56.7	29.4	11.6	29.8	15.0	14.8	2.8	2,182,701 ± 2,100	0.18	0.10
B800 250/500MHz	ENet	13.3	4.1	26.9	58.8	18.0	36.2	22.3	13.9	2.3	3,406,655 ±12,562	0.33	0.18
	ESPNet	6.2	2.2	11.4	18.0	6.4	36.6	15.8	20.8	2.6	3,445,554 ±12,450	0.33	0.18
	FPN	7.8	2.0	67.3	39.2	10.2	33.6	20.9	12.7	2.3	3,165,791 ±13,581	0.30	0.16
	U-Net	1.5	0.5	73.6	6.9	2.2	34.5	20.6	13.9	2.3	3,247,737 ±11,928	0.31	0.17
	Average	7.2	2.2	44.8	29.2	9.0	35.2	19.9	15.3	2.4	3,316,434 ± 6,298	0.32	0.17
B1024 200/400MHz	ENet	14.8	5.0	29.3	53.5	18.2	42.5	27.3	15.2	2.6	4,958,499 ±15,683	0.50	0.27
	ESPNet	6.5	2.6	11.8	14.8	5.9	42.2	18.0	24.1	2.9	4,917,392 ±16,489	0.50	0.27
	FPN	8.9	2.3	75.3	36.1	9.5	39.3	25.0	14.3	2.5	4,587,410 ±16,751	0.46	0.25
	U-Net	1.7	0.6	81.2	6.5	2.2	39.7	24.4	15.3	2.5	4,627,702 ±14,580	0.47	0.25
	Average	8.0	2.6	49.4	26.2	8.5	40.9	23.7	17.2	2.6	4,772,751 ± 7,907	0.48	0.26
B1152 167/333MHz	ENet	12.7	4.2	26.9	44.7	14.7	43.9	28.6	15.3	2.3	5,276,269 ±18,517	0.55	0.29
	ESPNet	5.8	2.3	11.3	12.4	4.9	44.4	19.5	24.9	2.7	5,344,894 ±18,015	0.56	0.29
	FPN	8.6	2.3	77.8	32.8	8.7	41.3	26.4	14.9	2.3	4,964,799 ±19,871	0.52	0.27
	U-Net	1.5	0.5	75.1	5.3	1.8	41.8	25.9	15.9	2.3	5,026,491 ±17,671	0.52	0.28
	Average	7.2	2.3	47.8	22.2	7.2	42.8	25.1	17.7	2.4	5,153,113 ± 9,236	0.54	0.28

Table 21: Evaluation results for performance, energy-efficiency, and model-level dependability for the ZU3EG (UZED-EG).

DPU	Model	Performance			MWTF		AVF (%)				SDC-Critical Area ±95% CI Error	SDC Rates (SDC · dev^{-1} · day^{-1})	
		FPS	FPS/W	Util (%)	FPS	FPS/W	SDC	SDC_T	SDC_C	Hangs		LEO	GEO
B512 375/750MHz	ENet	23.3	7.4	49.3	158.5	50.2	20.6	9.3	11.3	2.2	2,240,818 ± 5,569	0.03	0.08
	ESPNet	9.7	3.4	18.8	51.7	17.8	21.7	7.4	14.3	2.5	2,357,773 ± 9,253	0.03	0.09
	FPN	9.2	3.1	82.5	80.9	27.4	18.1	9.0	9.0	2.2	1,966,019 ± 7,946	0.03	0.07
	U-Net	1.8	0.7	91.4	12.8	5.1	19.3	8.2	11.2	2.2	2,100,344 ± 8,769	0.03	0.08
	Average	11.0	3.8	60.5	74.5	25.9	19.9	8.5	11.5	2.3	2,166,238 ± 3,714	0.03	0.08
B800 375/750MHz	ENet	24.8	6.6	33.5	165.8	43.9	21.3	9.9	11.5	2.0	2,692,359 ±22,366	0.05	0.13
	ESPNet	11.5	3.5	14.2	60.4	18.3	22.2	7.8	14.4	2.4	2,803,479 ±22,630	0.05	0.13
	FPN	12.2	3.1	70.6	102.7	26.3	20.0	10.7	9.3	2.0	2,526,930 ±21,299	0.04	0.12
	U-Net	2.4	0.7	77.0	16.0	4.9	20.8	9.4	11.5	2.0	2,631,772 ±22,479	0.05	0.13
	Average	12.7	3.6	48.9	84.3	23.7	21.1	9.4	11.7	2.1	2,663,635 ±11,095	0.05	0.13
B1024 300/600MHz	ENet	25.2	7.5	33.2	168.7	50.1	22.5	11.2	11.3	1.8	3,301,512 ±26,157	0.07	0.18
	ESPNet	11.7	3.8	14.2	60.9	19.5	22.9	8.4	14.5	2.0	3,363,317 ±26,821	0.07	0.18
	FPN	14.1	3.5	79.6	119.5	29.5	20.2	11.0	9.2	1.7	2,968,996 ±24,144	0.06	0.16
	U-Net	2.7	0.8	85.6	18.5	5.6	20.9	9.5	11.4	1.7	3,071,384 ±27,529	0.06	0.17
	Average	13.4	3.9	53.1	89.1	25.7	21.6	10.0	11.6	1.8	3,176,302 ±13,051	0.06	0.17
B1152 300/600MHz	ENet	25.0	6.6	29.4	170.3	45.1	21.9	10.7	11.2	1.7	3,150,305 ±25,030	0.07	0.18
	ESPNet	11.4	3.5	12.2	59.6	18.5	22.5	8.1	14.4	2.0	3,240,436 ±26,008	0.07	0.19
	FPN	16.3	3.7	81.5	133.4	29.9	20.9	11.5	9.4	1.7	3,011,382 ±26,530	0.06	0.18
	U-Net	2.8	0.8	77.2	18.6	5.3	21.2	9.8	11.4	1.7	3,052,462 ±28,955	0.07	0.18
	Average	13.9	3.7	50.1	91.3	24.4	21.6	10.0	11.6	1.8	3,113,646 ±13,264	0.07	0.18

5.3.3 Performance and Energy-Efficiency

The inference performance (FPS) and energy-efficiency ($^{FPS}/_{W}$), for each DL solution are shown in Tables 20 and 21. Both performance and energy efficiency are dependent upon both the DL model (e.g., network, operations, size, and data-flow) and accelerator (e.g., number of PEs and other computational units, bandwidth and caching, and operating frequency). Vitis AI (and DNNDK) provide runtime tools and libraries that can profile the inference performance and DPU utilization of a DL model. Furthermore, a power meter was used to measure the board power when the DL solution was (1) active and (2) unloaded (i.e., the FPGA is programmed with the design with all cells and nets associated with the DPU removed). The difference is approximately the power consumption exclusive to the DPU. The power of the unloaded DPU was measured at approximately 2.1W and 6.5W for the PYNQ-Z2 and UZED-EG, respectively.

The DPU primarily accelerates convolutional operations that often dominate the execution time of DL model inference. Despite this capability, DL models may contain nodes with little to no convolutional operations that underutilize the convolution architecture of the DPU. When profiled, all four evaluated DL models demonstrate varied DPU utilizations, resulting in varied performance scalability. For example, for the PYNQ-Z2 and B512 configuration operating at 100MHz/200MHz operation, both FPN and U-Net have high average utilizations (83% and 94%, respectively) resulting in an inference performance that scales approximately linearly to the DPU peak performance (2×PP×ICP×OCP), and both ENet and ESPNet have low average utilizations (51% and 22%, respectively) resulting in an inference performance that scales sublinearly. For the evaluated models, the DPU utilization decreases when the DPU scales and operating frequency increases, possibly due to insufficient scaling of memory bandwidth to maintain the DPU utilization. Furthermore, as the DPU scales, the area and power increase but the maximum frequency decreases due to place-and-route difficulties in designs with high resource utilization.

Next, we compare DL models in order of performance. ENet achieves the best performance due to low parameter count and workload MACs despite a medium DPU utilization (51%). Next, FPN is second due to a high DPU utilization (83%) despite high parameter

count and workload MACs. Next, ESPNet is third due to having the lowest DPU utilization (22%) despite also having the lowest parameter count, workload MACs. Next, U-Net is fourth due to having the highest parameter count, workload MACs despite also having the highest DPU utilization (94%).

When compared cycle-per-cycle, the Z7020 and ZU3EG both have similar inference performance. However, the maximum frequency is substantially higher for the ZU3EG than the Z7020, possibly due to a combination of (1) generational differences between both FPGA architectures (i.e., combined configurable logic blocks, added control sets, distribution RAM control, and flip-flop I/O, upgraded tile-based columnar architecture to improve flexibility and logic and routing efficiency in UltraScale and UltraScale+ FPGAs compared to 7-Series FPGAs [112]) and (2) higher bandwidth in the UZED-EG compared to the PYNQ-Z2 (128-bit AXI4 interface and DDR4 memory versus 64-bit AXI3 and DDR3 memory).

To summarize, DL models with higher DPU utilization have greater scalability, and performance and energy efficiency will increase with larger DPUs; however, this will decrease the maximum frequency and increase resource utilization and critical area vulnerable to SDC and hangs. DL models with lower DPU utilization have lesser scalability and can suffice with smaller DPUs with improved maximum frequency, resource utilization, and critical area. Additionally, lightweight properties (e.g., low parameter size and workload MACs) in DL models can also be favorable to improve performance and energy-efficiency.

5.3.4 Dependability

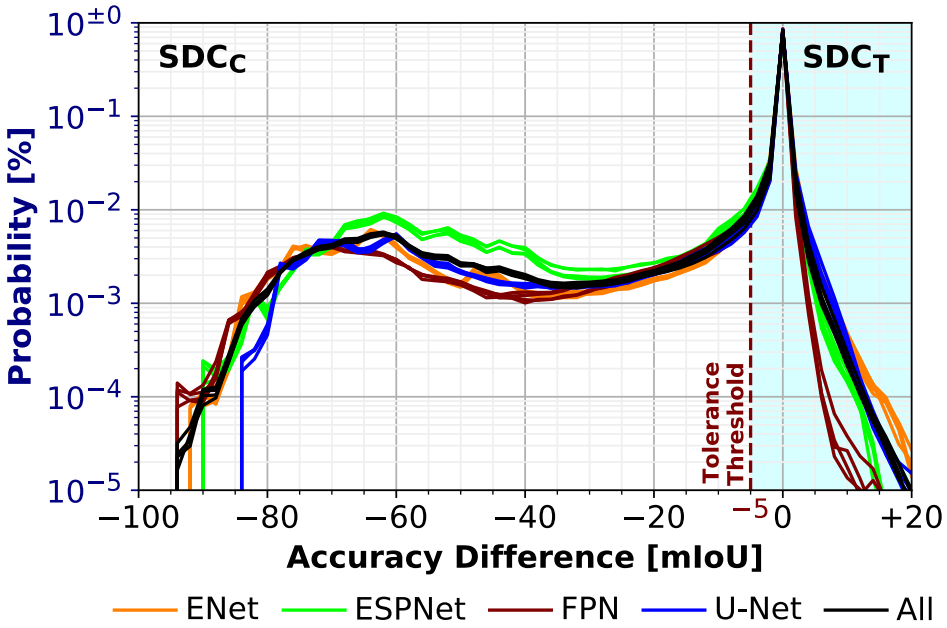
In this section, we apply our hierarchical CRAM fault-injection approach to evaluate and analyze the susceptibility of each DL solution (model, configuration, and SoC) to CRAM faults. Dependability metrics are quantified at both the model and node levels and are used to compare tradeoffs between options and to identify trends. Finally, we compare our hierarchical fault-injection approach to a traditional, direct approach to demonstrate substantial efficiency improvements. To parallelize the process, fault injection was performed on a cluster of 20 PYNQ-Z2 and 2 UZED-EG boards.

5.3.4.1 Model-Level Analysis

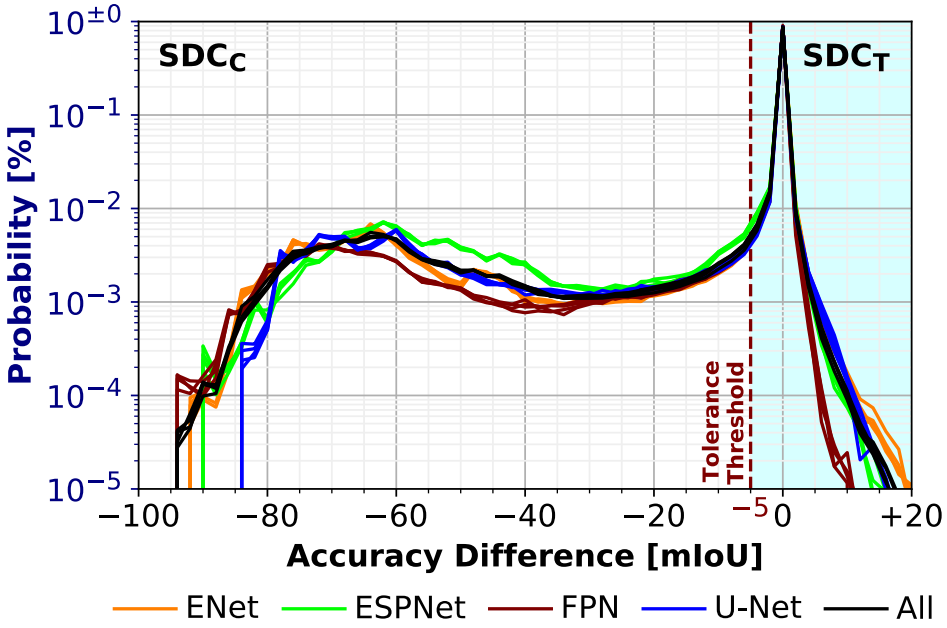
At the model level, fault injection is performed on the full DL solution, and the model-level AVF, MWTF, and critical area are calculated using equations Equations (2.1), (2.2), and (5.1), respectively, and are shown in Tables 20 and 21. In Figure 31, SDC criticality is represented as a histogram to illustrate the probabilities of losses in inference accuracy due to CRAM faults. Figure 32 illustrates experimental samples including the input image, ground-truth label mask, golden output, and SDC outputs with varied criticality. To demonstrate the significance of SDC criticality in our dependability analysis, we assume a tolerance threshold of -5%, where an mIoU difference greater than or equal to -5% is tolerable.

Most fault injections resulted as correct outputs or SDC_T with minor defects (mIoU difference near 0%). This observation is supported by related works that have demonstrated an inherent fault tolerance in DL algorithms [99]. A few fault injections resulted in SDC_T that improved the inference accuracy by a considerable amount (mIoU difference greater than 0%). The remaining fault injections resulted in SDC_C (mIoU difference less than the -5% tolerance threshold). Most notably, for each model, the shape of the distribution is similar across all tested DPUs for both SoCs. This observation indicates that SDC criticality is possibly more dependent on the characteristics of the DL model than the characteristics of the DPU.

Figure 33 shows the SDC-critical area averaged across all DL solutions in terms of SDC_T and SDC_C. For both SoCs, the SDC_T and SDC_C-critical areas both increase as the DPU scales. However, both the AVF and critical areas increase more rapidly in the Z7020 compared to the ZU3EG, possibly due to (1) generational differences between both FPGA architectures or (2) substantially fewer resources in the Z7020 resulting in higher FPGA design resource utilization. This trend is not observed for the average hang-critical area, which increases slightly as the DPU scales. For both SoCs, the ratio of SDC criticality (SDC_T to SDC_C) increases as the DPU scales, possibly due to the increasing ratio of functional to faulty PEs. For example, each PE in B512 will process more FM data and parameters than

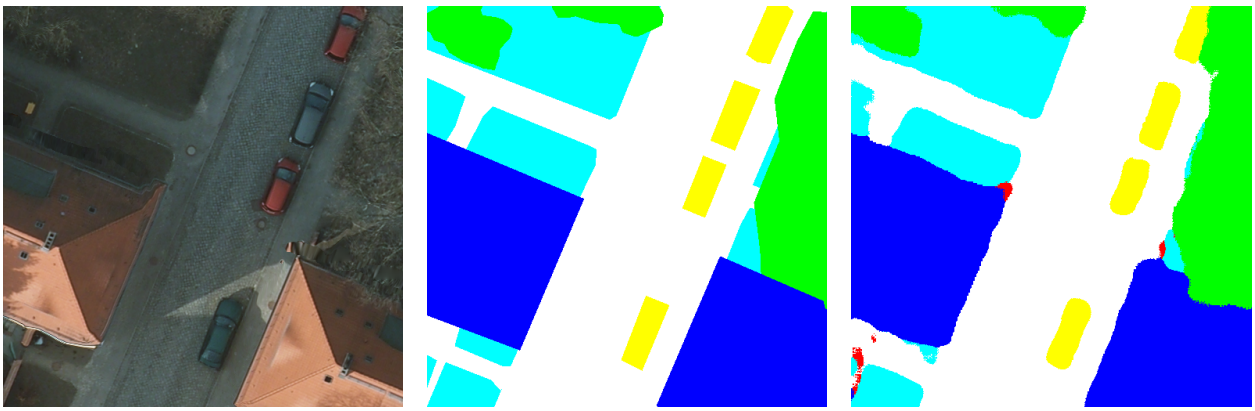


(a) Histogram of probability distribution for PYNQ-Z2 (Z7020)

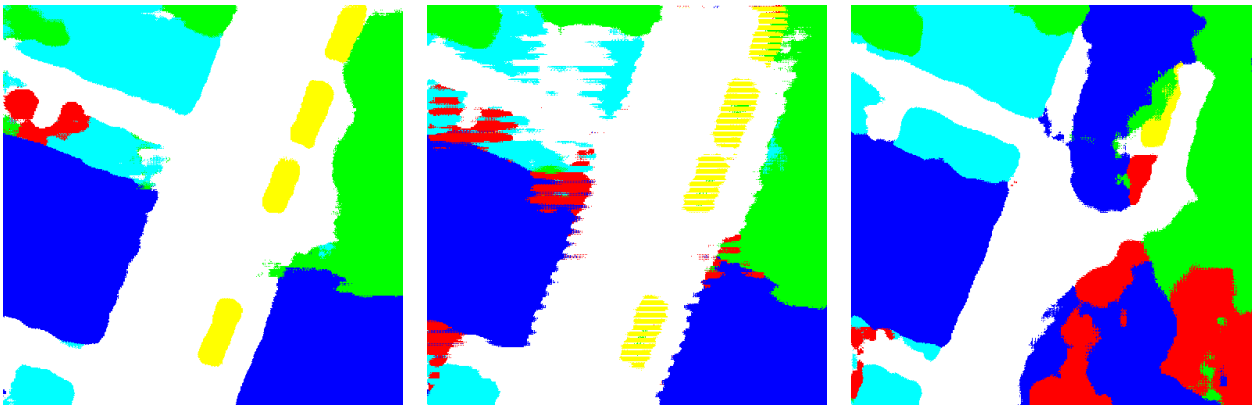


(b) Histogram of probability distribution for UZED-EG (ZU3EG)

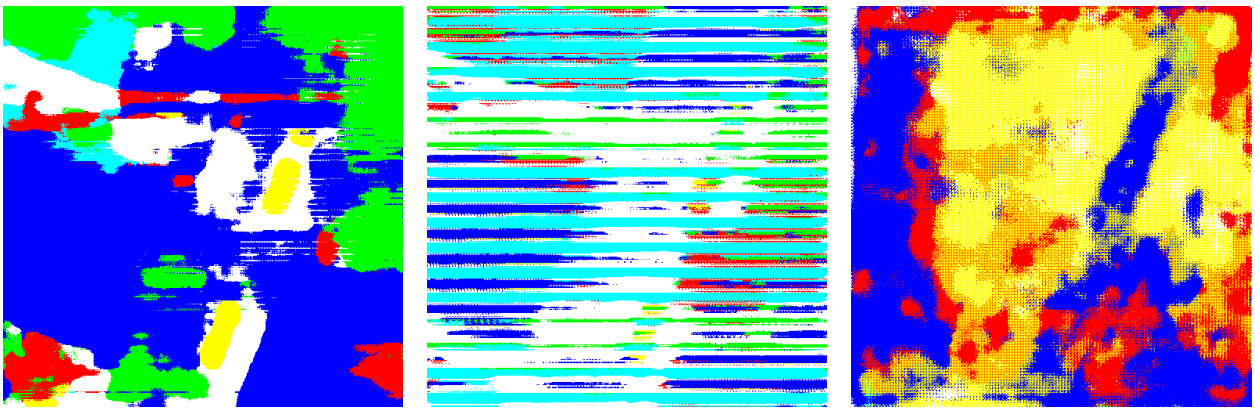
Figure 31: Impact of CRAM faults on mIoU with the probability distribution of mIoU difference in SDC events by DL solution for the (a) PYNQ-Z2 (Z7020) and (b) UZED-EG (ZU3EG). Multiple histograms represented as line plots are overlaid. 60 bins with widths of 2% mIoU loss per bin.



(a) Input image (b) Ground-truth mask (c) Golden output



(d) SDC_T (–4.86%) (e) SDC_C (–16.93%) (f) SDC_C (–33.81%)



(g) SDC_C (–50.07%) (h) SDC_C (–64.74%) (i) SDC_C (–76.76%)

Figure 32: DPU fault-injection experiment samples including input image, ground-truth label mask, golden output (86.33% mIoU), and variety of SDC outputs (with mIoU difference).

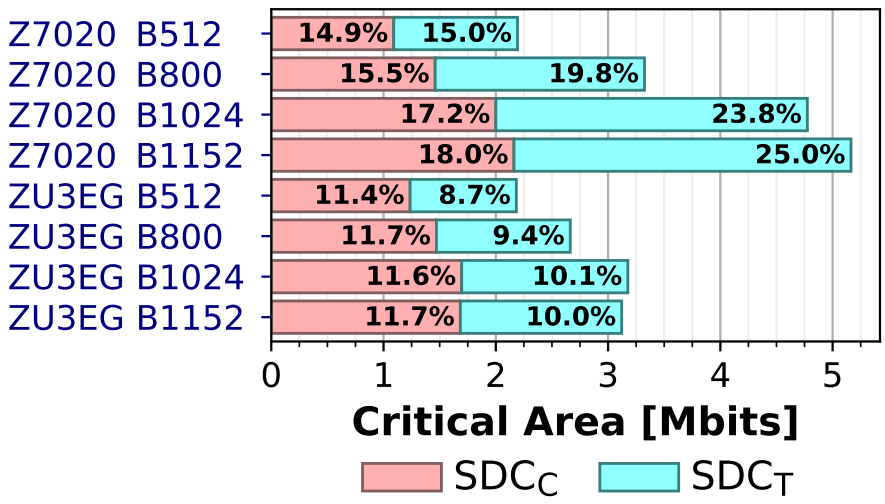


Figure 33: Average SDC-critical area of DPUs in terms of SDC_T and SDC_C. Percentage with respect to total area of DPU.

each PE in B1152 due to reduced parallelism in B512. As a result, one faulty PE in B512 can corrupt more data than one PE in B1152. However, the ratio of SDC criticality is greater and increases more rapidly in the Z7020 compared to the ZU3EG.

In the context of DL dependability, MWTF is a useful metric that combines the amount of useful work completed (performance and energy-efficiency) between SDC_C events. The MWTF for each DL solution is shown in Tables 20 and 21. MWTF is highly dependent upon the performance and SDC AVF of the DL solution. For the Z7020, MWTF decreases as the DPU scales due to rapid increases in SDC-critical area despite varied increases in performance. For the ZU3EG, MWTF increases slightly as the DPU scales due to moderate increases in SDC-critical area surpassed by increases in performance. ENet offers the highest performance and has a relatively low SDC AVF, and thus, achieves the highest MWTF. FPN offers medium performance but has the lowest SDC AVF, resulting in the second-highest MWTF. ESPNet also offers medium performance but has the highest SDC AVF, resulting in a low MWTF. Finally, U-Net offers the lowest performance despite a low SDC AVF, resulting in the lowest MWTF.

Using CRÈME96, the SEE fault rates for the LEO, particularly ISS orbit, and GEO environments were predicted for various resource types of the Z7020, assuming solar-minimum conditions and 100 mils of spherical, aluminum shielding. Using Equation (2.4), the predicted SEE rates are scaled by the DPU resource utilization (Table 19) and AVF of SDC (Tables 20 and 21) to approximate the SDC rates of each DPU for both orbital environments, as shown in Tables 20 and 21. The SDC rates are approximately proportional to the SDC-critical area of the DL solution. The substantially lower SDC rates in ZU3EG compared to the Z7020 designs are due to the reduced SEE susceptibility in UltraScale+ FPGAs compared to 7-Series FPGAs [49].

5.3.4.2 Node-Level Analysis

At the node level, fault injection is performed on randomly selected nodes of each DL model for each DL solution, and fault injection is performed exclusively on the model-level critical area vulnerable to both SDC and hangs. The node-level critical areas are calculated using Equation (5.2). Figures 34 and 35 illustrate the sequence of nodes executed by the DPU for each DL model, including the SDC_C-critical area (tolerance threshold of -5%) and set of operations performed by each node for the Z7020 and ZU3EG, respectively. Nodes with more than one operation are referred to as supernodes which have fused multiple node operations into one to improve efficiency. Convolution-based operations include convolution (ConvNd) and deconvolution (DeConvNd), and miscellaneous operations include concatenation (Concat), pooling (Pooling), elementwise (Eltwise), scale (Scale), and rectified linear unit (ReLU).

When all SDC (SDC_C and SDC_T) is considered, the SDC-critical area is roughly consistent between nodes and supernodes with the same set of operations. Furthermore, the SDC-critical area is generally much greater in nodes containing convolution-based operations than nodes containing solely miscellaneous operations, possibly due to the much greater DPU utilization in convolution-based operations. However, the hang-critical area is roughly consistent across all nodes and supernodes regardless of the operations.

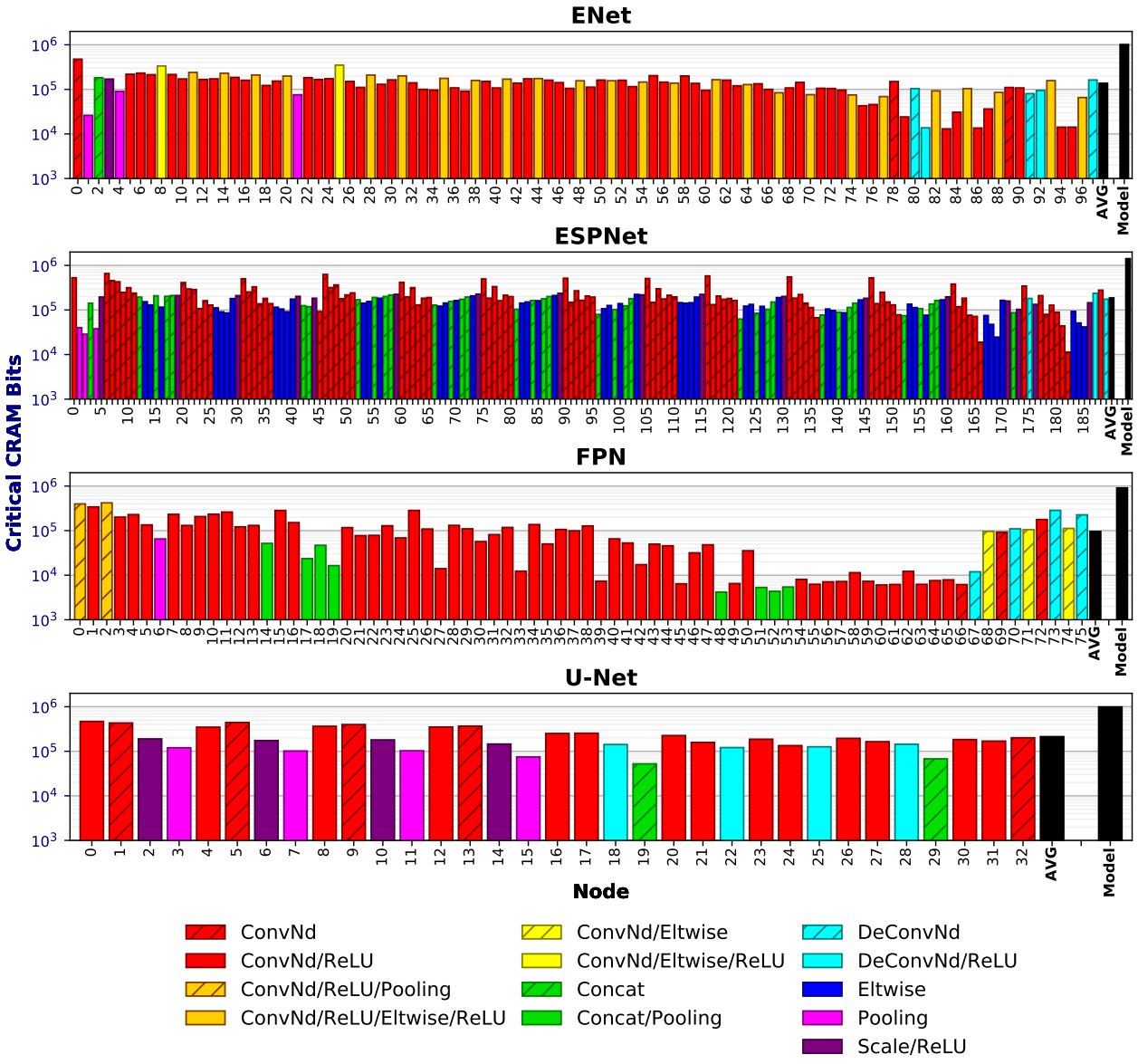


Figure 34: SDC_C-critical area and operations by node for each DL model for the B512 configuration of the DPU on the Z7020.

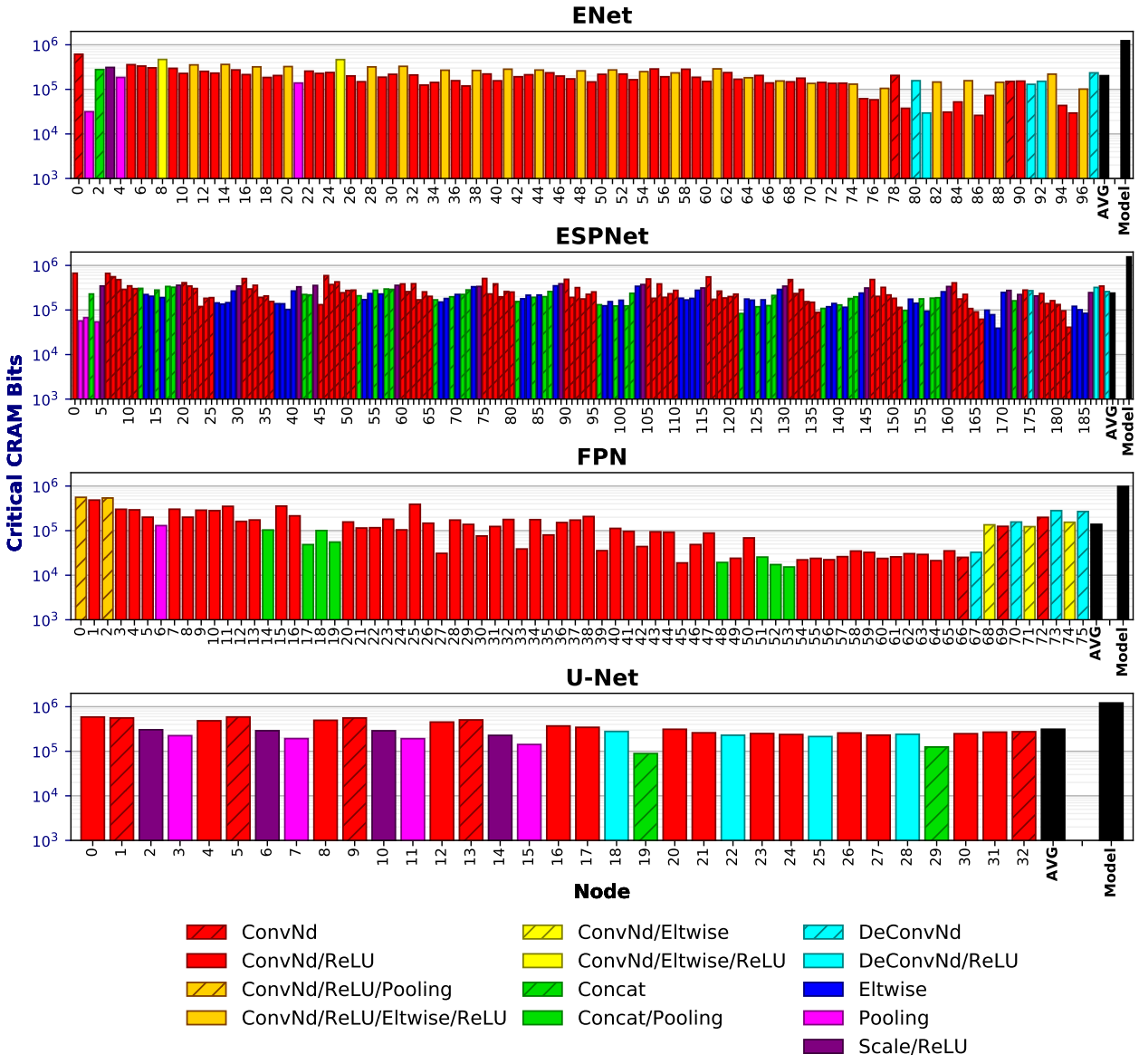


Figure 35: SDC_C-critical area and operations by node for each DL model for the B512 configuration of the DPU on the ZU3EG.

However, when only SDC_C is considered, the SDC_C-critical area has a much greater variation between nodes and supernodes with the same set of operations. Generally, the SDC_C-critical area in convolution-based nodes is greater than nodes without it, but some nodes are substantially less vulnerable than others. For example, in FPN, convolution-based nodes 54-66 are an order of magnitude less vulnerable than other convolution-based nodes of the same model. By evaluating and analyzing DL models at the node level, one can identify the most vulnerable nodes that can be prioritized for efficient SEE mitigation (e.g., selective replication).

5.3.4.3 Fault-Injection Evaluation

To demonstrate the efficiency and accuracy of our hierarchical fault-injection approach, we compare with a direct fault-injection approach targeting solely the essential area of the DPU for both model and node levels. First, we compare the fault-injection efficiency and speed-up. The total number of possible fault injections for both direct and hierarchical approaches is represented by Equation (5.3) and Equation (5.4), respectively, where N_m is the number of nodes for each model $m \in M$.

$$\text{Injections}_{\text{Direct}} = \sum_{m \in M} (N_m + 1) \times \text{Area}_{\text{E}} \tag{5.3}$$

$$\text{Injections}_{\text{Hierarchical}} = \sum_{m \in M} (\text{Area}_{\text{E}} + N_m \times \text{Area}_{\text{M,C}}) \tag{5.4}$$

For example, in the B512 configuration for the Z7020, the partial essential area (Area_{E}) is 7.3 Mbits, and the aggregated critical area ($\text{Area}_{\text{M,C}}$), which is the union of critical areas across all evaluated DL models, is 2.6 Mbits ($\text{AVF}_{\text{M}} = 35\%$). To evaluate all four models (397 nodes) at the node level, a direct approach has 2.91 billion possible injections, whereas our hierarchical approach has 1.04 billion possible injections, thus a maximum efficiency

improvement of 2.7×. Similarly, the B512 configuration for the ZU3EG, with Area_{E} and aggregated $\text{Area}_{\text{M,C}}$ equal to 10.9 Mbits and 2.5 Mbits, respectively, there is a maximum efficiency improvement of 4.2×. Efficiency is dependent upon the ratio of $\text{Area}_{\text{M,C}}$ to Area_{E}. As shown in Figure 33, this ratio increases rapidly as the DPU scales for the Z7020, resulting in efficiency decreasing from 2.7× (B512) to 2.1× (B1152). However, this ratio increases relatively slightly for the ZU3EG, so efficiency decreases slightly from 4.2× (B512) to 4.1× (B1152).

Table 22: Fault-injection accuracy for DPU B512 configuration on Z7020.

Approach	Level	Target Area (bits)		AVF SDC	Critical Area (bits)
Both	Model	Area_{E}	(7,333,820)	29.76%	2,182,702
Direct	Node	Area_{E}	(7,333,820)	16.39%	*1,201,803*
Hierarchical	Node	$\text{Area}_{\text{M,C}}$	(2,459,512)	46.07%	1,133,154
Inverse scaling by model-level coverage C_0 (94.72%)					*1,196,380*
Error compared to direct node-level approach					*0.45%*

Next, we compare the fault-injection accuracy. Both direct and hierarchical approaches result in equivalent approximations for $\text{Area}_{\text{M,C}}$ and similar estimations for $\text{Area}_{\text{N,C}}$. An example for the B512 configuration for the Z7020 averaging all four DL models is shown in Table 22. The direct approach, which targets Area_{E}, has a SDC AVF of 16.39% resulting in $\text{Area}_{\text{N,C}}$ with 1,201,803 bits. The hierarchical approach, which targets $\text{Area}_{\text{M,C}}$, has a SDC AVF of 46.07% resulting in $\text{Area}_{\text{N,C}}$ with 1,133,154 bits. However, our model-level procedure had a coverage C_0 of 94.72%, so 1–C_0 of Area_{E}, which may contain critical bits, was not tested. To adjust for untested, critical bits, $\text{Area}_{\text{N,C}}$ is inversely scaled by C_0, resulting in a final approximation for $\text{Area}_{\text{N,C}}$ with 1,196,380 bits, and a 0.45% error compared to the direct approach. Notably, compared to the direct approach, the SDC AVF of the hierarchical approach increased by a factor of 2.7× (equal to the maximum efficiency improvement), thus reaffirming the efficiency benefits of omitting inconsequential bits and exclusively targeting vulnerable bits.

5.4 Conclusion

Modern spacecraft increasingly require more computational capabilities to enable compute-intensive, DL methods that can enhance onboard analysis, spacecraft autonomy, and intelligent applications. Commercial-off-the-shelf FPGAs and hybrid SoCs, which provide the architectural capabilities for the onboard acceleration of DL algorithms, can address these requirements. However, commercial FPGAs and SoCs are highly susceptible to radiation-induced SEEs that can affect dependability. Furthermore, with a broad variety of DL tasks and a diverse collection of DL models, each with its own accuracy, resource utilization, performance, energy efficiency, and dependability characteristics, a methodology is required to evaluate and compare the tradeoffs between DL models and accelerators to select the optimal design.

In this chapter, we proposed a comprehensive methodology to evaluate the tradeoffs between DL models and accelerators. With an emphasis on dependability, we proposed a hierarchical fault-injection approach that continually narrows the set of targeted bits by removing bits with noncritical representation and thereby accelerating the fault-injection process. Compared to direct fault injection, our approach achieves an efficiency improvement of 2.1-2.7× and 4.1-4.2× for the Zynq-7000 and Zynq-MPSoC, respectively, to evaluate all DL solutions in this study. Furthermore, we describe methods to analyze the dependability of DL models and accelerators at both the model and node levels and in terms of the AVF, MWTF, and critical area. Finally, using this methodology, we evaluated, analyzed, and compared the tradeoffs and trends of four semantic-segmentation models across four configurations of the Xilinx DPU for two generations of Xilinx SoCs (Zynq-7000 and Zynq-MPSoC). Our evaluation, which was conducted on facsimiles of flight hardware that is currently deployed in space missions, demonstrates that compute-intensive DL applications can be dependably executed onboard for next-generation missions.

6.0 Conclusions

Due to continued innovations in onboard data analysis and spacecraft autonomy, enabled by deep learning (DL), modern spacecraft increasingly require dependable, high-performance computers to process onboard an immense volume of raw sensor data into actionable information to formulate critical decisions autonomously. To enable compute-intensive DL algorithms, commercial-off-the-shelf (COTS) processors, including FPGAs and system-on-chips (SoCs), are often employed for their superior performance, energy-efficiency, affordability, and capability compared to traditional radiation-hardened (rad-hard) alternatives; however, these commercial processors are highly susceptible to radiation-induced single-event effects (SEEs) that can degrade the dependability of the system and application. To enable dependable, high-performance systems capable of onboard DL using commercial FPGAs and SoCs, efficient methods for SEE mitigation are essential for maximizing system performability while satisfying mission dependability requirements.

In this dissertation, we proposed three key contributions to address these space-computing challenges. First, we proposed Hybrid, Adaptive, Reconfigurable Fault Tolerance (HARFT), a reconfigurable framework for environmentally adaptive resilience in hybrid and heterogeneous SoCs and systems for space applications. HARFT includes a runtime-reconfigurable system architecture that combines fault-tolerance frameworks in both CPU and FPGA subsystems of a hybrid SoC into one integrated, synergistic framework to enable unique repair and graceful-degradation mechanisms that elongate system uptime. By selecting between performance and dependability modes for both subsystems in response to the environmental condition, HARFT can maximize system performability subject to mission availability constraints. To evaluate this capability, a methodology was enhanced for evaluating environmentally adaptive and gracefully degradable systems, using phased-mission modeling, subject to dynamic near-Earth radiation environments, using a combination of orbital-perturbation, geomagnetic-field, and CRÈME96 models. When evaluated on the Zynq-7000, HARFT demonstrated substantial system performability gains compared to static approaches.

Next, we proposed Reconfigurable ConvNet (RECON), a reconfigurable framework for dependable, high-performance semantic segmentation for space applications. RECON consists of a runtime-reconfigurable semantic-segmentation accelerator architecture and includes selective and adaptive approaches for efficient SEE mitigation. In our selective approach, control-flow parts are selectively protected by TMR to minimize SEE-induced hangs, and in our adaptive approach, partial reconfiguration is leveraged to adapt the mitigation of silent data corruption of data-flow parts in response to the dynamic radiation environment. Combined, both approaches enable RECON to maximize performability subject to mission availability constraints. When evaluated on the Zynq-7000, RECON demonstrated substantial inference performability gains compared to static approaches.

Finally, we proposed a methodology for evaluating FPGA-accelerated DL models and analyzing their performance and dependability tradeoffs, including a hierarchical fault-injection approach to accelerate the characterization of fault susceptibility in DL solutions. In our hierarchical approach, fault injection is performed at multiple levels with varied application granularities (coarse to fine) while continually omitting inconsequential bits between levels to significantly reduce the number of fault injections required. Additionally, we proposed analytical methods that use our hierarchical fault-injection approach to quantify and examine FPGA-accelerated DL models in terms of well-established dependability metrics. We demonstrated the versatility of this methodology by evaluating, analyzing, and comparing multiple DL models accelerated on multiple configurations of the Xilinx DPU for the Zynq-7000 and Zynq-MPSoC.

The entirety of this dissertation research was evaluated on facsimiles of flight hardware that is currently deployed in space missions to demonstrate that compute-intensive DL algorithms, which are useful for spacecraft autonomy, onboard data analysis, and intelligent applications, can be dependably executed onboard for next-generation missions. This research can directly benefit missions that employ reconfigurable space computers, require high-performance onboard processing capabilities for compute-intensive applications, and are subject to mission dependability constraints. The application of these efficient resilience strategies can enable spacecraft designers to select from a design tradespace the optimal solution that maximizes system or application performability while satisfying dependability con-

straints. Opportunities for future work include (1) exploring novel, efficient SEE-mitigation strategies for other applications (e.g., other ML/CV, navigation, communication, etc.) and architectures (e.g., GPUs, TPUs, neuromorphic, etc.) and (2) enhancing the evaluation of these methods by radiation-beam testing or on-orbit demonstration.

Bibliography

[1] Francesco Abate, Luca Sterpone, Carlos A. Lisboa, Luigi Carro, and Massimo Violante. New techniques for improving the performance of the lockstep architecture for SEEs mitigation in FPGA embedded processors. *IEEE Transactions on Nuclear Science*, 56(4):1992–2000, Aug 2009.

[2] National Aeronautics and Space Administration. The year in review 2018, 2019 the year ahead. *Kennedy Space Center's Spaceport Magazine*, 6(1):1–22, February 2019.

[3] Dimitris Agiakatsikas, Nguyen T. H. Nguyen, Zhuoran Zhao, Tong Wu, Ediz Cetin, Oliver Diessel, and Lingkan Gong. Reconfiguration control networks for TMR systems with module-based recovery. In *2016 IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 88–91, May 2016.

[4] Mansoor Alam and Ubaid M. Al-Saggaf. Quantitative reliability evaluation of repairable phased-mission systems using Markov approach. *IEEE Transactions on Reliability*, 35(5):498–503, Dec 1986.

[5] Jordan D. Anderson, Jennings C. Leavitt, and Michael J. Wirthlin. Neutron radiation beam results for the Xilinx UltraScale+ MPSoC. In *2018 IEEE Nuclear Space Radiation Effects Conference (NSREC 2018)*, pages 1–7, July 2018.

[6] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. SegNet: a deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, 2017.

[7] John E. Ball, Derek T. Anderson, and Chee Seng Chan Sr. Comprehensive survey of deep learning in remote sensing: theories, tools, and challenges for the community. *Journal of Applied Remote Sensing*, 11(4):1 – 54, 2017.

[8] Keith L. Bedingfield, Richard D. Leach, and Margaret B. Alexander. Spacecraft system failures and anomalies attributed to the natural space environment. *NASA Technical Report NASA-RP-1390*, 1390:56, August 1996.

[9] Fabio Benevenuti, Fabiano Libano, Vincent Pouget, Fernanda Lima Kastensmidt, and Paolo Rech. Comparative analysis of inference errors in a neural network implemented in SRAM-based FPGA induced by neutron irradiation and fault injection methods. In *2018 31st Symposium on Integrated Circuits and Systems Design (SBCCI)*, pages 1–6, Aug 2018.

[10] Melanie Berg, Christian Poivey, David Petrick, Daniel Espinosa, Austin Lesea, Kenneth A. LaBel, Mark Friendlich, Hak Kim, and Anthony Phan. Effectiveness of internal versus external SEU scrubbing mitigation strategies in a Xilinx FPGA: Design, test, and analysis. *IEEE Transactions on Nuclear Science*, 55(4):2259–2266, Aug 2008.

[11] Cristiana Bolchini, Antonio Miele, and Marco D. Santambrogio. TMR and partial dynamic reconfiguration to mitigate SEU faults in FPGAs. In *22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT 2007)*, pages 87–95, Sep. 2007.

[12] Sébastien Bourdarie and Michael Xapsos. The near-Earth space radiation environment. *IEEE Transactions on Nuclear Science*, 55(4):1810–1832, Aug 2008.

[13] Cody Brewer, Nicholas Franconi, Robin Ripley, Alessandro Geist, Travis Wise, Sebastian Sabogal, Gary Crum, Sabrena Heyward, and Christopher Wilson. NASA SpaceCube intelligent multi-purpose system for enabling remote sensing, communication, and navigation in mission architectures. In *Proceedings of the 34th Annual AIAA/USU Conference on Small Satellites*, pages 1–6, Logan, UT, 2020. AIAA.

[14] Michael J. Campola and Jonathan A. Pellish. Radiation Hardness Assurance: Evolving for NewSpace. 2019.

[15] Matthew J. Cannon, Andrew M. Keller, Hayden C. Rowberry, Corbin A. Thurlow, Andés Pérez-Celis, and Michael J. Wirthlin. Strategies for removing common mode failures from TMR designs deployed on SRAM FPGAs. *IEEE Transactions on Nuclear Science*, 66(1):207–215, Jan 2019.

[16] Robert Cardillo. 2017 Small Satellite Conference keynote address. In *Proceedings of the 31st Annual AIAA/USU Conference on Small Satellites*, Logan, UT, aug 2017. AIAA.

[17] BAA DARPA. Blackjack (BAA HR001118S0032), May 2018.

[18] BAA DARPA. Blackjack Pit Boss (BAA HR001119S0012), April 2019.

[19] Fernando Fernandes dos Santos, Caio Lunardi, Daniel Oliveira, Fabiano Libano, and Paolo Rech. Reliability evaluation of mixed-precision architectures. In *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 238–249, Feb 2019.

[20] Boyang Du, Sarah Azimi, Corrado de Sio, Ludovica Bozzoli, and Luca Sterpone. On the reliability of convolutional neural network implementation on SRAM-based FPGA. In *2019 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pages 1–6, Oct 2019.

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

[22] Larry D. Edmonds. Proton SEU cross sections derived from heavy-ion test data. *IEEE Transactions on Nuclear Science*, 47(5):1713–1728, Oct 2000.

[23] Larry D. Edmonds and Farokh Irom. Extension of a proton SEU cross section model to include 14 MeV neutrons. *IEEE Transactions on Nuclear Science*, 55(1):649–655, Feb 2008.

[24] Marco Esposito, Simon S. Conticello, Massimiliano Pastena, and Bernardo Carnicero Domínguez. In-orbit demonstration of artificial intelligence applied to hyperspectral and thermal sensing from space. In *CubeSats and SmallSats for Remote Sensing III*, volume 11131, pages 88 – 96. International Society for Optics and Photonics, SPIE, 2019.

[25] Giulio Gambardella, Johannes Kappauf, Michaela Blott, Christoph Doehring, Martin Kumm, Peter Zipf, and Kees Vissers. Efficient error-tolerant quantized neural network accelerators. In *2019 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pages 1–6, Oct 2019.

[26] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, Pablo Martinez-Gonzalez, and Jose Garcia-Rodriguez. A survey on deep learning techniques for image and video semantic segmentation. *Applied Soft Computing*, 70:41–65, 2018.

[27] Alessandro Geist, Cody Brewer, Milton Davis, Nicholas Franconi, Sabrena Heyward, Travis Wise, Gary Crum, David Petrick, Robin Ripley, Christopher Wilson, and Thomas Flatley. SpaceCube v3.0 NASA next-generation high-performance processor for science applications. In *Proceedings of the 33rd Annual AIAA/USU Conference on Small Satellites*, pages 1–9, Logan, UT, 2019. AIAA.

[28] Alan D. George and Christopher M. Wilson. Onboard processing with hybrid and reconfigurable computing on small satellites. *Proceedings of the IEEE*, 106(3):458–470, March 2018.

[29] Robért Glein, Florian Rittner, and Albert Heuberger. Adaptive single-event effect mitigation for dependable processing systems based on FPGAs. *Microprocessors and Microsystems*, 59:46–56, 2018.

[30] Kaiyuan Guo, Shulin Zeng, Jincheng Yu, Yu Wang, and Huazhong Yang. [DL] a survey of FPGA-based neural network inference accelerators. *ACM Trans. Reconfigurable Technol. Syst.*, 12(1), March 2019.

[31] Tanya Harrison. Science enabled by high cadence and high resolution imagery from the Planet constellation of satellites. In *AGU Fall Meeting Abstracts*, volume 2019, pages PA54B–09, December 2019.

[32] James R. Heirtzler. The future of the South Atlantic anomaly and implications for radiation damage in space. *Journal of Atmospheric and Solar-Terrestrial Physics*, 64(16):1701–1708, 2002. Space Weather Effects on Technological Systems.

[33] Felix R. Hoots and Ronald L. Roehrich. Models for propagation of NORAD element sets. Technical report, AEROSPACE DEFENSE COMMAND PETERSON AFB CO OFFICE OF ASTRODYNAMICS, 1980.

[34] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2704–2713, 2018.

[35] Adam Jacobs, Grzegorz Cieslewski, Alan D. George, Ann Gordon-Ross, and Herman Lam. Reconfigurable fault tolerance: A comprehensive framework for reliable and adaptive FPGA-based space computing. *ACM Trans. Reconfigurable Technol. Syst.*, 5(4):21:1–21:30, December 2012.

[36] Andrew E. Johnson, Yang Cheng, James F. Montgomery, Nikolas Trawny, Brent Tweddle, and Jason X. Zheng. *Real-Time Terrain Relative Navigation Test Results from a Relevant Environment for Mars Landing*.

[37] Jonathan M. Johnson and Michael J. Wirthlin. Voter insertion algorithms for FPGA designs using triple modular redundancy. In *Proceedings of the 18th Annual ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, FPGA '10, pages 249–258, New York, NY, USA, 2010. ACM.

[38] Shrikanth Kanekal, Lauren Blum, Eric Christian, Gary Crum, Jeff Dumonthier, Allison Evans, Thomas Flatley, Ashley Greeley, Sergio Guerro, Agbontaen Imasuen, John Lucas, James Mackinnon, Nikolaos Paschalidis, Deepak Patel, Khary Parker, Quintin Schiller, Errol Summerlin, and George Suarez. CeREs: The Compact Radiation Belt Explorer. In *Proceedings of the 29th Annual AIAA/USU Conference on Small Satellites*, pages 1–11, Logan, UT, 2018. AIAA.

[39] Josef Koller, Geoffrey D. Reeves, and Reiner H. W. Friedel. LANL* V1.0: a radiation belt drift shell model suitable for real-time and reanalysis applications. *Geoscientific Model Development*, 2(2):113–122, 2009.

[40] Israel Koren and C. Mani Krishna. *Fault-Tolerant Systems*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2007.

[41] Kuang-Hua Huang and Jacob A. Abraham. Algorithm-based fault tolerance for matrix operations. *IEEE Transactions on Computers*, C-33(6):518–528, June 1984.

[42] Kenneth A. LaBel. Radiation effects on electronics 101. *NASA Electronic Parts and Packaging Program (NEPP)*, Apr 2004.

[43] Kenneth A. LaBel, Allan H. Johnston, Janet L. Barth, Robert A. Reed, and Charles E. Barnes. Emerging Radiation Hardness Assurance (RHA) issues: a NASA approach for space flight programs. *IEEE Transactions on Nuclear Science*, 45(6):2727–2736, Dec 1998.

[44] Kenneth A. LaBel and Jonathan A. Pellish. National Radiation Hardness Assurance (RHA) planning for NASA missions: Updated guidance. *NASA Electronic Parts and Packaging Program (NEPP)*, Mar 2014.

[45] Fahad Lateef and Yassine Ruichek. Survey on semantic segmentation using deep learning techniques. *Neurocomputing*, 338:321–348, 2019.

[46] Andrew Lavin and Scott Gray. Fast algorithms for convolutional neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4013–4021, 2016.

[47] Robert Le. Soft error mitigation using prioritized essential bits. *Xilinx XAPP538 (v1. 0)*, 2012.

[48] David S. Lee, Gregory R. Allen, Gary Swift, Matthew Cannon, Michael Wirthlin, Jeffrey S. George, Rokutaro Koga, and Kangsen Huey. Single-event characterization of the 20 nm Xilinx Kintex UltraScale field-programmable gate array under heavy ion irradiation. In *2015 IEEE Radiation Effects Data Workshop (REDW)*, pages 1–6, July 2015.

[49] David S. Lee, Michael King, William Evans, Matthew Cannon, Andrés Pérez-Celis, Jordan Anderson, Michael Wirthlin, and William Rice. Single-event characterization of 16 nm FinFET Xilinx UltraScale+ devices with heavy ion and neutron irradiation. In *2018 IEEE Nuclear Space Radiation Effects Conference (NSREC 2018)*, pages 1–8, July 2018.

[50] David S. Lee, Michael Wirthlin, Gary Swift, and Anthony C. Le. Single-event characterization of the 28 nm Xilinx Kintex-7 field-programmable gate array under heavy ion irradiation. In *2014 IEEE Radiation Effects Data Workshop (REDW)*, pages 1–5, July 2014.

[51] Ganghee Lee, Dimitris Agiakatsikas, Tong Wu, Ediz Cetin, and Oliver Diessel. TLegUp: A TMR code generation tool for SRAM-based FPGA applications using HLS. In *2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 129–132, April 2017.

[52] Fabiano Libano, Brittany Wilson, Jon-Paul Anderson, Michael J. Wirthlin, Carlo Cazzaniga, Christopher Frost, and Paolo Rech. Selective hardening for neural networks in FPGAs. *IEEE Transactions on Nuclear Science*, 66(1):216–222, Jan 2019.

[53] Fabiano Libano, Brittany Wilson, Michael Wirthlin, Paolo Rech, and John Brunhaver. Understanding the impact of quantization, accuracy, and radiation on the reliability of convolutional neural networks on FPGAs. *IEEE Transactions on Nuclear Science*, 67(7):1478–1484, July 2020.

[54] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, 2017.

[55] Tyler M. Lovelly, Donavon Bryan, Kevin Cheng, Rachel Kreynin, Alan D. George, Ann Gordon-Ross, and Gabriel Mounce. A framework to analyze processor architectures for next-generation on-board space computing. In *2014 IEEE Aerospace Conference*, pages 1–10, March 2014.

[56] Tyler M. Lovelly and Alan D. George. Comparative analysis of present and future space-grade processors with device metrics. *Journal of Aerospace Information Systems*, 14(3):184–197, Mar 2017.

[57] Sachin Mehta, Mohammad Rastegari, Anat Caspi, Linda Shapiro, and Hannaneh Hajishirzi. ESPNet: Efficient spatial pyramid of dilated convolutions for semantic segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[58] David J. Miranda. 2020 NASA technology taxonomy: 2015 technology areas to 2020 taxonomy areas crosswalk. 2020.

[59] Sparsh Mittal. A survey of FPGA-based accelerators for convolutional neural networks. *Neural Computing and Applications*, 32(4):1109–1139, Feb 2020.

[60] Mischa Möstl, Alexander Dörflinger, Mark Albers, Harald Michalik, and Rolf Ernst. Self-adaptation for availability in CPU-FPGA systems under soft errors. In *2019 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, pages 9–16, July 2019.

[61] Shubhendu S. Mukherjee, Christopher Weaver, Joel Emer, Steven K. Reinhardt, and Todd Austin. A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor. In *Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36.*, pages 29–40, Dec 2003.

[62] National Academies of Sciences, Engineering, and Medicine. *Vision and Voyages for Planetary Science in the Decade 2013-2022*. The National Academies Press, Washington, DC, 2011.

[63] National Academies of Sciences, Engineering, and Medicine. *Achieving Science with CubeSats: Thinking Inside the Box*. The National Academies Press, Washington, DC, 2016.

[64] National Academies of Sciences, Engineering, and Medicine. *Testing at the Speed of Light: The State of U.S. Electronic Parts Space Radiation Testing Infrastructure*. The National Academies Press, Washington, DC, 2018.

[65] National Academies of Sciences, Engineering, and Medicine. *Thriving on Our Changing Planet: A Decadal Strategy for Earth Observation from Space*. The National Academies Press, Washington, DC, 2018.

[66] National Academies of Sciences, Engineering, and Medicine. *Report Series: Committee on Astrobiology and Planetary Science: Review of the Planetary Science Aspects of NASA SMD's Lunar Science and Exploration Initiative*. The National Academies Press, Washington, DC, 2019.

[67] Suzanne F. Nowicki, Stephen A. Wender, and Michael Mocko. The Los Alamos Neutron Science Center spallation neutron sources. *Physics Procedia*, 90:374–380, 2017.

[68] Paul P. O'Brien and Sébastien Bourdarie. The IRBEM library – open source tools for radiation belt modeling. *AGU Fall Meeting Abstracts*, pages IN53C–1760, December 2012.

[69] Björn Osterloh, Harald Michalik, Sandi A. Habinc, and Björ Fiethe. Dynamic partial reconfiguration in space applications. In *2009 NASA/ESA Conference on Adaptive Hardware and Systems*, pages 336–343, July 2009.

[70] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. ENet: A deep neural network architecture for real-time semantic segmentation. *CoRR*, abs/1606.02147, 2016.

[71] Sam Pedrotty, Jacob Sullivan, Elisabeth Gambone, and Thomas Kirven. Seeker free-flying inspector GNC flight performance. 02 2020.

[72] Edward L. Petersen. The SEU figure of merit and proton upset rate calculations. *IEEE Transactions on Nuclear Science*, 45(6):2550–2562, Dec 1998.

[73] Jason A. Poovey, Thomas M. Conte, Markus Levy, and Shay Gal-On. A benchmark characterization of the EEMBC benchmark suite. *IEEE Micro*, 29(5):18–29, Sep. 2009.

[74] ISPRS Potsdam. 2D semantic labeling dataset, 2018.

[75] Paul Pukite and Jan Pukite. *Markov Modeling for Reliability Analysis*. Wiley-IEEE Press, 1st edition, 1998.

[76] Heather Quinn. Challenges in testing complex systems. *IEEE Transactions on Nuclear Science*, 61(2):766–786, April 2014.

[77] Heather Quinn. Radiation effects in reconfigurable FPGAs. *Semiconductor Science and Technology*, 32(4):044001, mar 2017.

[78] Heather Quinn, Tom Fairbanks, Justin L. Tripp, George Duran, and Beatrice Lopez. Single-event effects in low-cost, low-power microprocessors. In *2014 IEEE Radiation Effects Data Workshop (REDW)*, pages 1–9, July 2014.

[79] George A. Reis, Jonathan Chang, Neil Vachharajani, Shubhendu S. Mukherjee, Ram Rangan, and David I. August. Design and evaluation of hybrid fault-detection systems. In *32nd International Symposium on Computer Architecture (ISCA'05)*, pages 148–159, June 2005.

[80] Kimberly Robinson, Andrew Schorr, and David Smith. NASA's Space Launch System: Opportunities for small satellites to deep space destinations. In *Proceedings of the 32nd Annual AIAA/USU Conference on Small Satellites*, pages 1–9, Logan, UT, 2018. AIAA.

[81] William H. Robinson, Michael L. Alles, Theodore A. Bapty, Bharat L. Bhuva, Jeffrey D. Black, Alfred B. Bonds, Lloyd W. Massengill, Sandeep K. Neema, Ronald D. Schrimpf, and Jason M. Scott. Soft error considerations for multicore microprocessor design. In *2007 IEEE International Conference on Integrated Circuit Design and Technology*, pages 1–4, May 2007.

[82] Seth Roffe, Theodore Schwarz, Thomas Cook, Noah Perryman, Justin Goodwill, Evan Gretok, Aidan Phillips, Mitchell Moran, Tyler Garrett, and Alan George. CASPR: autonomous sensor processing experiment for STP-H7. In *Proceedings of the 34th Annual AIAA/USU Conference on Small Satellites*, pages 1–11, Logan, UT, 2020. AIAA.

[83] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[84] Daniel Sabogal and Alan D. George. Towards resilient spaceflight systems with virtualization. In *2018 IEEE Aerospace Conference*, pages 1–8, March 2018.

[85] Sebastian Sabogal, Patrick Gauvin, Brad Shea, Daniel Sabogal, Antony Gillette, Christopher Wilson, Ansel Barchowsky, Alan D. George, Gary Crum, and Thomas Flatley. SSIVP: Spacecraft supercomputing experiment for STP-H6. In *Proceedings of the 31st Annual AIAA/USU Conference on Small Satellites*, pages 1–12, Logan, UT, 2017. AIAA.

[86] Sebastian Sabogal, Alan George, and Gary Crum. ReCoN: A reconfigurable CNN acceleration framework for hybrid semantic segmentation on hybrid SoCs for space applications. In *2019 IEEE Space Computing Conference (SCC)*, pages 41–52, July 2019.

[87] Sebastian Sabogal, Alan George, and Christopher Wilson. Reconfigurable framework for environmentally adaptive resilience in hybrid space systems. *ACM Trans. Reconfigurable Technol. Syst.*, 13(3), July 2020.

[88] Aitzan Sari and Mihalis Psarakis. Scrubbing-based SEU mitigation approach for systems-on-programmable-chips. In *2011 International Conference on Field-Programmable Technology*, pages 1–8, Dec 2011.

[89] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[90] Christopher Scolese. 2020 Small Satellite Conference keynote address. In *Proceedings of the 34th Annual AIAA/USU Conference on Small Satellites*, Logan, UT, aug 2020. AIAA.

[91] Alex Shye, Joseph Blomstedt, Tipp Moseley, Vijay J. Reddi, and Daniel A. Connors. PLR: A software approach to transient fault tolerance for multicore architectures. *IEEE Transactions on Dependable and Secure Computing*, 6(2):135–148, April 2009.

[92] Felix Siegle, Tanya Vladimirova, Jørgen Ilstad, and Omar Emam. Mitigation of radiation effects in SRAM-based FPGAs for space applications. *ACM Comput. Surv.*, 47(2):37:1–37:34, January 2015.

[93] Luca Sterpone and Massimo Violante. A new reliability-oriented place and route algorithm for SRAM-based FPGAs. *IEEE Transactions on Computers*, 55(6):732–744, June 2006.

[94] Aaron Stoddard, Ammon Gruwell, Peter Zabriskie, and Michael J. Wirthlin. A hybrid approach to FPGA configuration scrubbing. *IEEE Transactions on Nuclear Science*, 64(1):497–503, Jan 2017.

[95] Michael A. Swartout. CubeSats mission assurance trends. In *Proceedings of the NASA Electronic Parts and Packaging (NEPP) Electronics Technology Workshop (ETW)*, Greenbelt, MD, June 2020. NASA GSFC.

[96] Lucas A. Tambara, Felipe Almeida, Paolo Rech, Fernanda L. Kastensmidt, Giovanni Bruni, and Christopher Frost. Measuring failure probability of coarse and fine grain TMR schemes in SRAM-based FPGAs under neutron-induced effects. In *Applied Reconfigurable Computing*, pages 331–338, Cham, 2015. Springer International Publishing.

[97] Erwan Thébault, Christopher C. Finlay, Ciarán D. Beggan, Patrick Alken, Julien Aubert, Olivier Barrois, Francois Bertrand, Tatiana Bondar, Axel Boness, Laura Brocco, et al. International Geomagnetic Reference Field: the 12th generation. *Earth, Planets and Space*, 67(1):79, 2015.

[98] Jorge Tonfat, Fernanda Lima Kastensmidt, Paolo Rech, Ricardo Reis, and Heather M. Quinn. Analyzing the effectiveness of a frame-level redundancy scrubbing technique for SRAM-based FPGAs. *IEEE Transactions on Nuclear Science*, 62(6):3080–3087, Dec 2015.

[99] César Torres-Huitzil and Bernard Girau. Fault tolerance in neural networks: Neural design and hardware implementation. In *2017 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, pages 1–6, Dec 2017.

[100] Nikolai A. Tsyganenko. A magnetospheric magnetic field model with a warped tail current sheet. *Planetary and Space Science*, 37(1):5–20, 1989.

[101] Allan J. Tylka, James H. Adams, Paul R. Boberg, Buddy Brownstein, William F. Dietrich, Erwin O. Flueckiger, Edward L. Petersen, Margaret A. Shea, Don F. Smart, and Edward C. Smith. CREME96: A revision of the Cosmic Ray Effects on Micro-Electronics Code. *IEEE Transactions on Nuclear Science*, 44(6):2150–2160, Dec 1997.

[102] Dazhi Wang and Kishor S. Trivedi. Reliability analysis of phased-mission system with independent component repairs. *IEEE Transactions on Reliability*, 56(3):540–551, Sep. 2007.

[103] Ingo Wardinski, Diana Saturnino, Hagay Amit, Aude Chambodut, Benoit Langlais, Mioara Mandea, and Thébault Erwan. Geomagnetic core field models and secular variation forecasts for the 13th International Geomagnetic Reference Field (IGRF-13). *Earth, Planets and Space*, 72(1):155, Oct 2020.

[104] Xuechao Wei, Cody Hao Yu, Peng Zhang, Youxiang Chen, Yuxin Wang, Han Hu, Yun Liang, and Jason Cong. Automated systolic array architecture synthesis for high throughput CNN inference on FPGAs. In *Proceedings of the 54th Annual Design Automation Conference 2017*, DAC '17, New York, NY, USA, 2017. Association for Computing Machinery.

[105] Caleb Williams and Stephanie DelPozzo. 2020 nano/microsatellite market forecast, 10th edition. *SpaceWorks Enterprises, Inc*, 1, 2020.

[106] Christopher Wilson and Alan D. George. CSP hybrid space computing. *Journal of Aerospace Information Systems*, 15(4):215–227, Feb 2018.

[107] Christopher Wilson, James MacKinnon, Patrick Gauvin, Sebastian Sabogal, Alan D. George, Gary Crum, and Thomas Flatley. µCSP: A diminutive, hybrid, space processor for smart modules and CubeSats. In *Proceedings of the 30th Annual AIAA/USU Conference on Small Satellites*, pages 6–11, Logan, UT, 2016. AIAA.

[108] Christopher Wilson, Sebastian Sabogal, Alan D. George, and Ann Gordon-Ross. Hybrid, adaptive, and reconfigurable fault tolerance. In *2017 IEEE Aerospace Conference*, pages 1–11, March 2017.

[109] Christopher Wilson, Jacob Stewart, Patrick Gauvin, James MacKinnon, James Coole, Jonathan Urriste, Alan D. George, Gary Crum, Elizabeth Timmons, Jaclyn Beck, Thomas Flatley, Michael Wirthlin, Alex Wilson, and Aaron Stoddard. CSP hybrid space computing for STP-H5/ISEM on ISS. In *Proceedings of the 29th Annual AIAA/USU Conference on Small Satellites*, pages 1–12, Logan, UT, 2015. AIAA.

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

[111] Michael A. Xapsos, Patrick M. O'Neill, and T. Paul O'Brien. Near-Earth space radiation models. *IEEE Transactions on Nuclear Science*, 60(3):1691–1705, June 2013.

[112] Xilinx. *UltraScale Architecture Configurable Logic Block*. Xilinx, v1.5 edition, Feb 2017. Xilinx User Guide (UG574).

[113] Xilinx. *Zynq-7000 SoC Technical Reference Manual*. Xilinx, v1.12.2 edition, Jul 2018. Xilinx User Guide (UG585).

[114] Xilinx. *Libmetal and OpenAMP for Zynq Devices User Guide*. Xilinx, v2019.1 edition, May 2019. Xilinx User Guide (UG1186).

[115] Xilinx. *Zynq UltraScale+ Device Technical Reference Manual*. Xilinx, v2.1 edition, Aug 2019. Xilinx User Guide (UG1085).

[116] Xilinx. ML Caffe segmentation tutorial, Dec 2020. Accessed: 2021-02-01.

[117] Xilinx. *Vitis AI User Guide*. Xilinx, v1.3 edition, Dec 2020. Xilinx User Guide (UG1414).

[118] Xilinx. *Zynq DPU*. Xilinx, v3.3 edition, Dec 2020. Xilinx Product Guide (PG338).

[119] Chen Zhang, Peng Li, Guangyu Sun, Yijin Guan, Bingjun Xiao, and Jason Cong. Optimizing FPGA-based accelerator design for deep convolutional neural networks. In *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '15, pages 161–170, New York, NY, USA, 2015. ACM.

[120] Hongyan Zhang, Michael A. Kochte, Michael E. Imhof, Lars Bauer, Hans-Joachim Wunderlich, and Jörg Henkel. GUARD: Guaranteed reliability in dynamically reconfigurable systems. In *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, June 2014.

[121] Zhuoran Zhao, Dimitris Agiakatsikas, Nguyen T. H. Nguyen, Ediz Cetin, and Oliver Diessel. Fine-grained module-based error recovery in FPGA-based TMR systems. In *2016 International Conference on Field-Programmable Technology (FPT)*, pages 101–108, Dec 2016.

[122] Zhuoran Zhao, Nguyen T. H. Nguyen, Dimitris Agiakatsikas, Ganghee Lee, Ediz Cetin, and Oliver Diessel. Fine-grained module-based error recovery in FPGA-based TMR systems. *ACM Trans. Reconfigurable Technol. Syst.*, 11(1), January 2018.