Reconfigurable Framework for Resilient Semantic Segmentation for Space Applications

SEBASTIAN SABOGAL and ALAN GEORGE, University of Pittsburgh
GARY CRUM, NASA Goddard Space Flight Center

Deep learning (DL) presents new opportunities for enabling spacecraft autonomy, onboard analysis, and intelligent applications for space missions. However, DL applications are computationally intensive and often infeasible to deploy on radiation-hardened (rad-hard) processors, which traditionally harness a fraction of the computational capability of their commercial-off-the-shelf counterparts. Commercial FPGAs and system-on-chips present numerous architectural advantages and provide the computation capabilities to enable onboard DL applications; however, these devices are highly susceptible to radiation-induced single-event effects (SEEs) that can degrade the dependability of DL applications. In this article, we propose Reconfigurable ConvNet (RECON), a reconfigurable acceleration framework for dependable, high-performance semantic segmentation for space applications. In RECON, we propose both selective and adaptive approaches to enable efficient SEE mitigation. In our selective approach, control-flow parts are selectively protected by triple-modular redundancy to minimize SEE-induced hangs, and in our adaptive approach, partial reconfiguration is used to adapt the mitigation of dataflow parts in response to a dynamic radiation environment. Combined, both approaches enable RECON to maximize system performability subject to mission availability constraints. We perform fault injection and neutron irradiation to observe the susceptibility of RECON and use dependability modeling to evaluate RECON in various orbital case studies to demonstrate a 1.5–3.0× performability improvement in both performance and energy efficiency compared to static approaches.

CCS Concepts: • **Hardware** → **Fault tolerance**; • **Computer systems organization** → *Reconfigurable computing*; *Heterogeneous (hybrid) systems*; *System on a chip*; *Availability*; *Reliability;*

Additional Key Words and Phrases: Environmentally adaptive computing, dependability modeling, space computing, single-event effects

ACM Reference format:
Sebastian Sabogal, Alan George, and Gary Crum. 2021. Reconfigurable Framework for Resilient Semantic Segmentation for Space Applications. *ACM Trans. Reconfigurable Technol. Syst.* 14, 4, Article 22 (September 2021), 32 pages.
https://doi.org/10.1145/3472770

This work was supported by SHREC industry and agency members and by the IUCRC Program of the National Science Foundation under Grant No. CNS-1738783. This work was performed, in part, at the Los Alamos Neutron Science Center (LANSCE), an NNSA User Facility operated for the U.S. Department of Energy (DOE) by Los Alamos National Laboratory (Contract 89233218CNA000001).
Authors' addresses: S. Sabogal and A. George, University of Pittsburgh, NSF SHREC Center, Swanson School of Engineering, Pittsburgh, PA, 15213; emails: {sebastian.sabogal, alan.george}@pitt.edu; G. Crum, NASA Goddard Space Flight Center, Science Data Processing Branch (Code 587), Greenbelt, MD, 20771; email: gary.a.crum@nasa.gov.

1936-7406/2021/09-ART22 $15.00
https://doi.org/10.1145/3472770

1 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 [29]. 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 require real-time capabilities to execute critical spacecraft maneuvers and operations autonomously. **Deep learning (DL)** presents several opportunities for enhancing spacecraft autonomy, onboard data analysis, and intelligent applications for space missions. One example is semantic segmentation, a powerful DL and computer-vision 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 **Earth observations (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 **radiation-hardened (rad-hard)** space processors. The 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.

To address these challenges, space missions continue to adopt **small satellites (SmallSats)**, including CubeSats, as low-SWaP-C platforms [32]. Furthermore, to improve onboard processing capabilities, SmallSat and CubeSat missions frequently employ systems developed with solely commercial-off-the-shelf technology or with a mix of commercial and rad-hard devices including both FPGAs and hybrid **system-on-chips (SoCs)** [16]. 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 SoCs and FPGAs provide superior performance, energy efficiency, and affordability compared to their rad-hard alternatives but are highly susceptible to radiation-induced **single-event effects (SEEs)** that can affect the dependability of the system and application [28]. 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 that 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 article, 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 dataflow 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 dataflow subsystem, which is more vulnerable to SEE-induced **silent data corruption (SDC)** but is faster to repair, is protected using an environmentally adaptive strategy leveraging dynamic partial reconfiguration. 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 partial reconfiguration, RECON can adapt its dataflow 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 dataflow 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 [2], a symmetric autoencoder 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.

The remainder of this article is organized as follows. Section 2 provides a cursory overview of background topics and related work. Section 3 introduces the RECON framework, describes the application of RECON for hybrid and heterogeneous SoCs and systems, and explains the selective and adaptive approaches for efficient SEE mitigation. Section 4 evaluates RECON in terms of performance and dependability. Finally, Section 5 concludes this article with findings of this research.

2 BACKGROUND

This section provides a cursory overview of space-computing trends including SmallSats, CubeSats, DL for space missions, and commercial hybrid and heterogeneous SoCs and systems that enable onboard DL processing. Next, topics in DL including **neural network (NN)** and **convolutional NN (CNN)** basics, semantic segmentation, accuracy metrics, and CNN architectures and optimizations for FPGA acceleration are covered. Furthermore, concepts in dependable computing such as radiation effects on electronic devices, SEE susceptibility of SRAM-based FPGAs, including mitigation techniques and evaluation methods, and environmentally adaptive systems and modeling for near-Earth radiation environments are discussed. Finally, this section provides a discussion of related work in the analysis, evaluation, and mitigation of SEEs in DL applications accelerated on FPGAs.

2.1 SmallSats, CubeSats, and Onboard DL

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 $10 \times 10 \times 10$ 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 [32, 47, 53]. 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. However, 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 [32]. 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 for future missions in EO [34]. The decadal strategy accentuated the need for advanced methodologies to analyze and convert EO 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 addition to advancing science missions, SmallSat and CubeSat technology

are also emerging in future defense missions. The Defense Advanced Research Projects Agency 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 [10]. 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 [9].

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) [57] and Xilinx Zynq UltraScale+ MPSoC (Zynq-MPSoC) [58], combine fixed-logic CPUs with reconfigurable-logic FPGAs in a single device. The Zynq-7000 SoC features single- or dual-core ARM Cortex-A9 **application processor unit (APU)** and an Artix or Kintex 7-Series FPGA fabric. The Zynq-MPSoC features a multiprocessor system, including dual- or quad-core ARM Cortex-A53 APU, dual-core ARM Cortex-R5 real-time processor unit, TMR MicroBlaze platform management unit, and an UltraScale+ Architecture FPGA fabric. In both SoC series, the CPU and FPGA subsystems can interact over the **Advanced eXtensible Interconnect (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. Xilinx SoCs and FPGAs 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.

The **Center for High-Performance Reconfigurable Computing (CHREC) Space Processor (CSP)** [54] and **Space, High-Performance, and Resilient Computing (SHREC) Space Processor (SSP)** [40] are two examples of multifaceted hybrid space computers. CSP was developed by researchers at the National Science Foundation (NSF) CHREC in collaboration with NASA **Goddard Space Flight Center (GSFC)**, and SSP was developed at the NSF Center for 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 SoC (Z7020 or Z7030/Z7035/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. CSP has flight heritage as part of two U.S. Department of Defense **Space Test Program (STP)** Houston missions to the International Space Station, including STP-H5 CHREC Space Processor (STP-H5-CSP) and STP-H6 Spacecraft Supercomputing for Image and Video Processing (STP-H6-SSIVP) [41, 54]. Both CSP and SSP are planned for flight on the STP-H7 Configurable and Autonomous Sensor Processing Research (STP-H7-CASPR) [40]. Derivatives of CSP include the SHREC Hybrid Computer and SpaceCube Mini-Z [7], both developed at NASA and featured on many new science missions.

The Science Data Processing Branch at NASA GSFC is developing the SpaceCube v3.0 VPX (SCv3VPX) [15] and SpaceCube v3.0 Mini (SCv3M) [7] 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

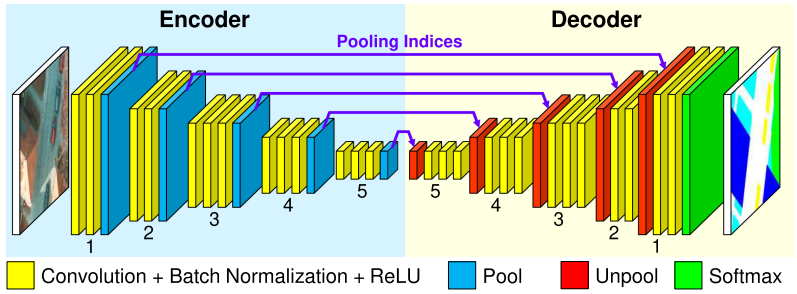


Fig. 1. SegNet semantic-segmentation model.

processor card (e.g., SSP) for external management or self-managed with a soft-core processor (e.g., MicroBlaze or RISC-V).

2.3 Convolutional Neural Networks

CNNs have become increasingly popular in DL and computer-vision applications for classification, detection, localization, and segmentation tasks in imagery [44]. 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 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.

2.4 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 [14, 23]. One semantic-segmentation model proposed by Badrinarayanan et al. [2], called *SegNet*, is illustrated in Figure 1. SegNet 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 that produces two outputs: downsampled FMs and **pooling indices (PIs)**. Each decoder block begins with a max-unpooling layer that 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 pixelwise probability distribution. The argmax of the output layer can also be used to assign the most probable label for each pixel.

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)**, **false positive (FP)**, and **false negative (FN)** using Equation (1). For multi-class segmentation, to determine the **mean IoU (mIoU)** and mean F1, respectively, the IoU and F1 are calculated for each class and are then averaged across all classes,

$$\text{IoU} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}} \quad \text{and} \quad \text{F1} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}. \tag{1}$$

2.5 FPGA Acceleration of DL Applications

The acceleration of DL on FPGAs has been explored extensively in the literature [17, 30]. 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 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. [19] constrains the continuous input set (floating point) 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 [24] 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), 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 $V = B^T dB$, 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 $Y = A^T WA$,

$$Y = A^T \left[(GgG^T) \odot (B^T dB) \right] A. \tag{2}$$

The $F(2 \times 2, 3 \times 3)$ 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 $F(2 \times 2, 3 \times 3)$ form has a $2.25\times$ improvement in multiplication efficiency (number of multiplies to compute one output pixel) at a $2.625\times$ 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.

Finally, architectural optimizations, such as systolic arrays, DSP time-multiplexing, and layer fusion, can further improve CNN acceleration. Wei et al. [52] proposed a two-dimensional (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 [17]. 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 [17]. Miscellaneous, channelwise 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.

2.6 Radiation Effects

Radiation is an environmental hazard that pose several challenges for electronic devices including hybrid SoCs and FPGAs in space. Radiation sources include **galactic cosmic rays (GCRs)**, solar particle events, and charged particles trapped within the Van Allen radiation belts. Radiation effects on electronic devices are typically categorized as long-term cumulative effects or short-term transient effects. Cumulative effects include total ionizing dose, the ionizing radiation dose absorbed by the device material over time causing parametric or functional degradation, and displacement damage dose, the non-ionizing damage caused by particles colliding with atoms of the device lattice structure. SEEs are transient effects that occur when a single radiation particle strike deposits enough charge to cause an effect. SEEs can be destructive or nondestructive. Destructive SEEs include single-event latch-up, single-even burnout, single-event gate rupture, and others. Nondestructive SEEs include **single-event upset (SEU)**, single-event transient, and single-event functional interrupt. Both types of effects are extensively covered by the National Academies' report on the U.S. infrastructure for space radiation effects testing [33]. Additionally, NASA created the **Radiation Hardness Assurance (RHA)**, a multi-step approach to address radiation concerns in spacecraft development [22]. NASA further evolved RHA for SmallSat missions that require high reliability but are too cost-constrained to follow standard RHA practices [8].

2.7 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. 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 [38, 45, 55].

2.7.1 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, whereas coarse-grain replication provides greater area efficiency [48]. A variety of tools have been developed for the automatic triplication and insertion of majority voters 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 academic tool that supports selective replication of designs in a post-synthesis netlist [21].

Hardware redundancy can be combined with PR for **module-based error recovery (MER)** [5]. 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 [1, 59, 60].

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 [4], and the scrubbing approach can be categorized into blind, readback, replacement, or hybrid forms [46].

2.7.2 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. Three essential metrics for quantifying the dependability of a design include the **architectural vulnerability factor (AVF)**, critical area, and **mean-work-to-failure (MWTF)**. In the context of this article, the AVF of a design is the probability that an injected fault

will manifest into an observable event [31]. The critical area of a design combines the design AVF and area to account for differences in the resource utilization [25]. MWTF describes the amount of useful work completed until an observable event is expected [39]. The classification of observable events is user-defined and can vary by design or application (e.g., SDC or hangs). AVF, critical area, and MWTF are calculated using Equations (3), (4), and (5), respectively,

$$\text{AVF} = \frac{\text{Number of Observable Events}}{\text{Number of Fault Injections}}. \tag{3}$$

$$\text{Critical Area} = \text{AVF} \times \text{Area}, \tag{4}$$

$$\text{MWTF} = \frac{\text{Amount of Useful Work Completed}}{\text{Number of Observable Events}}. \tag{5}$$

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 [37]. The cross-section is calculated by dividing the number of observable events by the beam fluence using Equation (6). 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 [37],

$$\sigma = \frac{\text{Number of Observable Events}}{\text{Total Effective Fluence}}. \tag{6}$$

2.8 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 [6, 56]. Jacobs et al. [20] and Sabogal et al. [43] 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 [18] 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 [51] 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 [50], 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 (7). 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.

$$\lambda_{\text{design}}(t) = \sum_{r \in R} \lambda_{r,\text{SEE}}(t) \cdot \text{RU}_r \cdot \text{AVF}_r. \tag{7}$$

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

2.9 Related Work

The evaluation, analysis, and mitigation of SEEs in machine-learning applications accelerated on FPGAs have been moderately explored in the literature [3, 11–13, 26, 27, 42, 49]. An overview of concepts and taxonomy for dependability in FPGA-based NNs, including passive and active methods for fault tolerance, is provided in [49]. A variety of methods using fault injection and radiation-beam testing have been explored to evaluate the dependability of NNs. Du et al. [12] 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. [3] 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. [11] 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. [27] 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 severity, which describes the impact of faults on inference accuracy.

Methods to improve NN dependability using efficient methods for SEE mitigation have also been explored. Libano et al. [26] 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. [13] 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. [42] 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

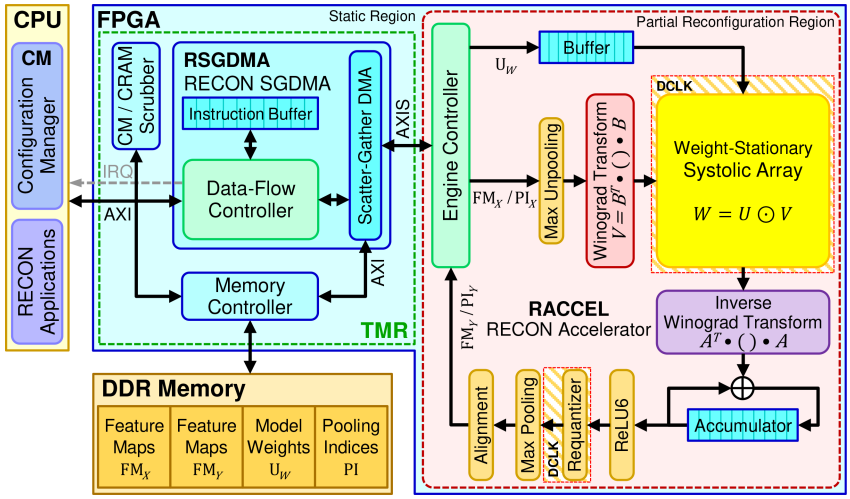


Fig. 2. RECON acceleration framework.

control-flow part of the acceleration framework to minimize the hang rate, and high-performance, unmitigated and low-performance, TMR versions of the dataflow part were evaluated to quantify the SDC rate and the tradeoffs in performance and dependability.

In this article, we extend upon our previous work in Reference [42] 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 maximize inference performance with efficient hardware. Second, we propose the partitioning of control-flow and dataflow 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 dataflow 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.

3 ARCHITECTURE OVERVIEW

This section provides an architectural overview of the RECON framework, which is illustrated in Figure 2. 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 full reconfiguration. The CM exists as software on a CPU or as a controller residing in the SR.

The RSGDMA and RACCEL modules constitute the control-flow and dataflow 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 dataflow 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.

3.1 Approaches for Efficient SEE Mitigation

In this article, we use a transient-fault model to address the dependability of RECON due to SEE-induced faults, and we focus on the mitigation of two SEE-induced events: *SDC* and *hangs*. SDC refers to an erroneous outcome of the application due to errors that are neither detectable nor correctable without dependable-computing techniques. SDC usually occurs when faults affect the dataflow parts of the design (i.e., datapath) and manifest into data errors. 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 [49]. 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 and corrupt finite-state machines (e.g., entry into invalid states), disrupt flow-control processes, or adversely activate/inhibit control signals that prevent completion of the execution. 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 partial reconfiguration are fast, nondisruptive recovery mechanisms to repair faults. Full reconfiguration 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 dataflow (RACCEL) subsystems, and the selective and adaptive approaches are applied to the RSGDMA and RACCEL subsystems, respectively.

3.1.1 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 dataflow-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, then the CM activates the decoupler and performs PR to recover the RACCEL with minimal system interruption.

3.1.2 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 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 3 illustrates an example of this adaptive approach for a RECON framework with mitigated $\text{RSGDMA}_{\text{4-TMR}}$, which supports RACCEL_N with $N \leq 4$. 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, we

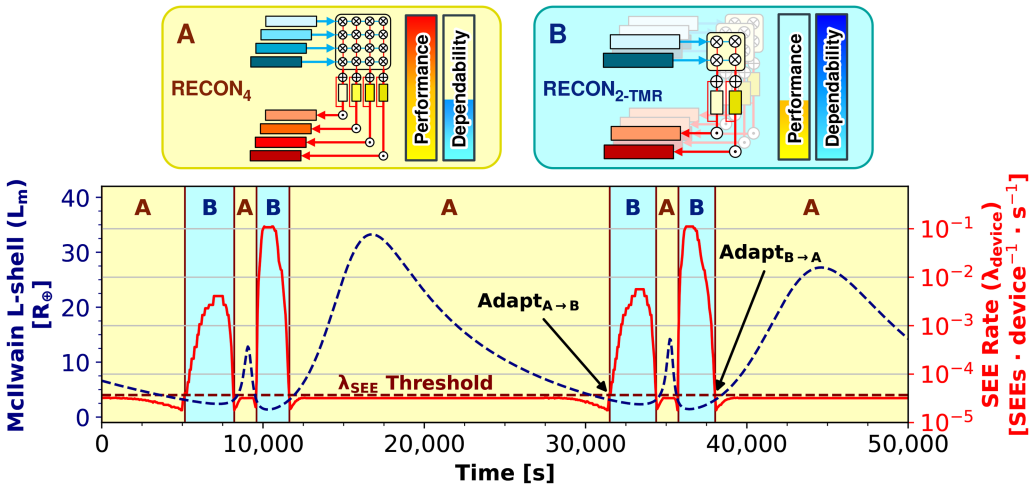


Fig. 3. Adaptive approach for SEE mitigation selects between high-performance $Mode_A$ ($RECON_4$) and high-dependability $Mode_B$ ($RECON_{2\text{-}TMR}$) in response to the current SEE rate of the orbital environment.

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.

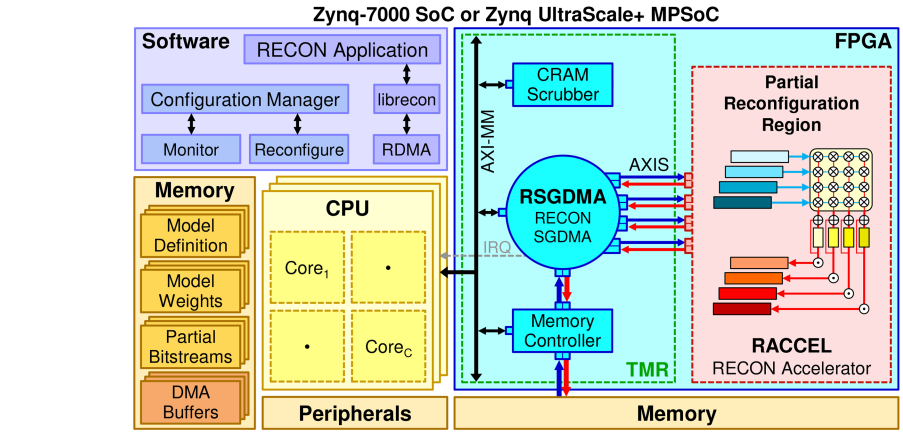
3.2 Architectures for Space Computers

The RECON framework can be deployed on space computers featuring a hybrid and heterogeneous SoC and system architecture. Space computers, such as the SpaceCube Mini-Z (Z7020) [7], CSP (Z7020) [54], and SSP (Z7030/Z7035/Z7045) [40] 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 4(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.

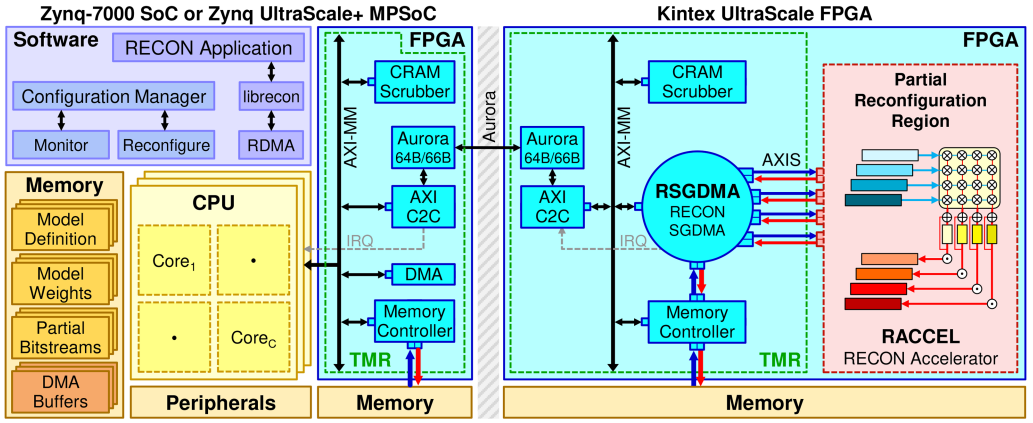
RECON can also be deployed on space computers featuring heterogeneous, disaggregated CPU-FPGA systems, which often combine a large FPGA coprocessor to a relatively low-profile CPU or SoC interfaced by a high-speed interconnect. One example is the space single-board computer architecture exemplified by the SCv3VPX design (Zynq-MPSoC and KU-FPGA) [15]. 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) [7] connected to a SoC (e.g., SSP). In both examples, illustrated in Figure 4(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 4, 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



(a)



(b)

Fig. 4. RECON architecture for (a) hybrid SoCs and (b) heterogeneous SoC/FPGA systems.

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.

3.3 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.

3.3.1 Model-Compression Optimizations. RECON uses the INT8 quantization scheme in Reference [19] 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 floating point (FP32)**. RECON uses INT8 quantization to attain the hardware-efficiency benefits associated with low-precision hardware at the expense of a generally slight decrease in accuracy due to precision error.

3.3.2 Algorithmic Optimizations. Using the $F(2 \times 2, 3 \times 3)$ 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.

3.3.3 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, channelwise max-pooling layers are optional postprocess operations that follow the convolutional layers. Inversely, the decompressive, channelwise 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 dataflow is illustrated in Figure 2.

Finally, the RSGDMA uses an AXI-based **scatter-gather DMA (SGDMA)**, 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 EVALUATION

This section describes the performance and dependability evaluations for RECON. RECON is configured to use the optimizations discussed in Section 3.3.3 and is implemented for the Z7020 (PYNQ-Z2 and Zybo Z7-20) and ZU3EG (Ultra96-V2 and UltraZed-EG) that serve as emulators for hybrid space computers (similar to those described in Sections 2.2 and 3.2). The Z7020 and ZU3EG devices use the configuration illustrated in Figure 4(a) but use CPU-interfaced DDR memory for DMA buffers. In our performance evaluation, both platforms are evaluated in terms of accuracy, resource utilization, performance, 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 operating system. FG-TMR is applied using the BL-TMR tool [21]. The Potsdam dataset of the ISPRS commission II/4 benchmark for 2D semantic labeling [36] is used for this evaluation. This dataset uses EO imagery in 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).

4.1 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 that 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.

4.1.1 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 (1), the interface accuracy of the segmented images is measured in terms of the mIoU and F1 metrics, and the results are shown in Table 1. For this evaluation, an mIoU difference of −1.7% to −0.7% was observed with loss decreasing as the model size increased. Although the INT8 version deviates in accuracy compared

Table 1. 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

Table 2. RECON 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%)
$RACCEL_{1\text{-TMR}}$	10,604 (19.93%)	10,939 (10.28%)	21 (15.00%)	30 (13.64%)	2,614,220 (10.20%)
$RACCEL_{2\text{-TMR}}$	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%)
$RSGDMA_{4\text{-TMR}}$	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%)
$RACCEL_{1\text{-TMR}}$	11,321 (16.04%)	11,674 (8.27%)	21 (9.72%)	30 (8.33%)	4,344,837 (14.10%)
$RACCEL_{2\text{-TMR}}$	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%)
$RSGDMA_{4\text{-TMR}}$	26,379 (37.39%)	19,452 (13.78%)	123 (56.94%)	0 (0.00%)	10,630,561 (34.48%)

to FP32 due to low-precision hardware, the loss in accuracy is a small and acceptable tradeoff for the hardware efficiency benefits of INT8.

4.1.2 Resource Utilization. The resource utilization of several implemented RECON modules (RACCEL and RSGDMA) are shown separately in Table 2. 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 channelwise datapaths. Furthermore, the application of FG-TMR in $RACCEL_N$ incurs a 3–5× increase in resource utilization compared to $RACCEL_{N\text{-TMR}}$. For this evaluation, the amount of OCM used for tiling and accumulation is set to 8,192 pixels per channel.

4.1.3 Performance and Energy-Efficiency. Performance and energy efficiency, quantified in **frames-per-second (FPS)** and FPS-per-watt (FPS/W), respectively, are measured for several configurations of SegNet executed as software on the SoC CPU or accelerated on RECON. RECON operates at the maximum frequencies (noted as logic/DSP) constrained by the dividers of the APU clock source. In the software versions, the FPGA 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 3 shows the performance and energy-efficiency measurements. In all situations,

Table 3. 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
$RECON_{1\text{-}TMR}$	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
$RECON_{2\text{-}TMR}$	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
$RECON_{1\text{-}TMR}$	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
$RECON_{2\text{-}TMR}$	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

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. The Z7020 and ZU3EG have 64-bit AXI3 and 128-bit AXI4 interconnects, respectively, so the memory bandwidth of each device saturates when $N > 2$ and $N > 4$, respectively. Furthermore, as the model size decreases quartically ($Net_A \rightarrow Net_B \rightarrow Net_C$), the performance also increases quartically.

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 1.97 W and 5.20 W 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.

4.2 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 2.8, for three orbital case studies, including the Jason-3 in LEO, NOAA-20 in sun-synchronous orbit, and Molniya 1-88 in highly elliptical orbit. The selected orbital case studies represent the dynamic radiation environment of three distinct orbital regimes to demonstrate the

versatility of RECON. Spacecraft in **geostationary orbit (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 analyze the fault-injection results and the impact of CRAM faults on the inference accuracy to determine the AVF in terms of SDC_T, SDC_C, and hang events. 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.

4.2.1 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 [21]. 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 s). 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

Table 4. RECON Model-level CRAM Fault Injection Test Results on PYNQ-Z2 (Z7020)

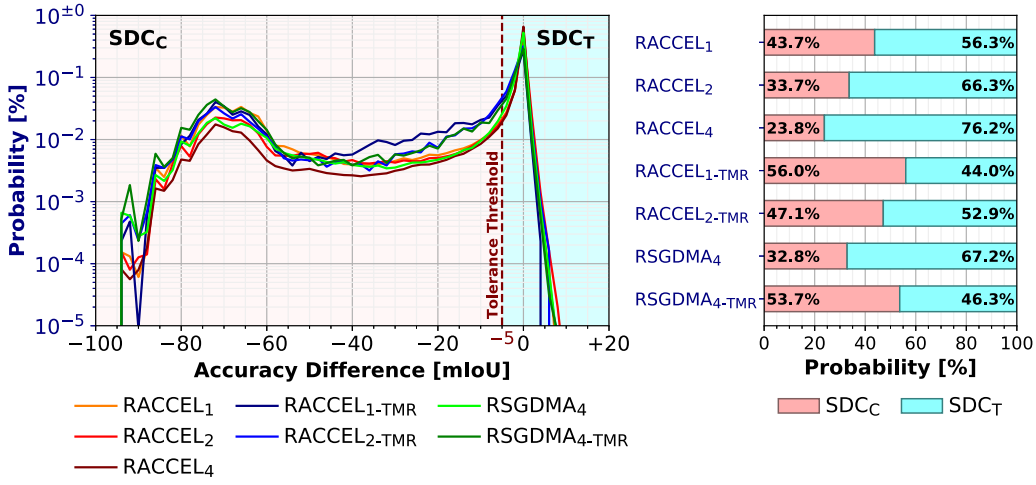
Module	Injections	AVF (%)			Critical CRAM Bits ±95% CI Error		
		SDC_T	SDC_C	Hangs	SDC_T	SDC_C	Hangs
$RACCEL_1$	2,459,068	16.14	12.55	6.13	124,292.3 ± 386.7	96,622.3 ± 341.0	47,237.9 ± 238.4
$RACCEL_{1\text{-TMR}}$	6,936,806	0.09	0.11	0.06	2,335.9 ± 58.2	2,977.5 ± 65.7	1,490.5 ± 46.5
Improvement					53.2×	32.5×	31.7×
$RACCEL_2$	4,851,293	23.38	11.86	3.43	341,524.5 ± 628.5	173,267.7 ± 447.7	50,095.6 ± 240.7
$RACCEL_{2\text{-TMR}}$	9,876,758	0.14	0.13	0.05	6,889.1 ± 114.7	6,144.5 ± 108.3	2,259.9 ± 65.7
Improvement					49.6×	28.2×	22.2×
$RACCEL_4$	8,819,211	33.02	10.33	1.85	1,087,874.6 ± 1,249.5	340,387.8 ± 698.9	61,107.2 ± 296.1
$RSGDMA_4$	2,773,146	13.75	6.73	10.03	160,463.3 ± 509.3	78,475.0 ± 356.1	117,014.6 ± 434.9
$RSGDMA_{4\text{-TMR}}$	4,573,435	0.17	0.19	0.04	7,960.9 ± 177.8	9,216.3 ± 191.3	2,101.8 ± 91.4
Improvement					20.2×	8.5×	55.7×

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, are performed to minimize the 95% CI error. To accelerate this process, the Xilinx design tools are used to generate a list of essential CRAM bits, which are CRAM bits actively used by the design, to target exclusively [25]. Furthermore, several PYNQ-Z2 boards are deployed to parallelize the fault-injection campaign.

Table 4 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 4, 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 2) and 10% decrease in energy efficiency. Similarly, the application of FG-TMR substantially reduces the critical area of both $RACCEL_{1\text{-TMR}}$ and $RACCEL_{2\text{-TMR}}$ compared with their unmitigated counterparts in terms of SDC and hangs, also at the expense of increased area and energy overhead.

Figure 5(a) illustrates a histogram that shows the impact of model-level CRAM faults on the inference accuracy (mIoU) of SDC events. This 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% to −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%). A benign SDC event occurs when a faulty execution results in the correct classification of pixels that would otherwise be mislabeled. For this evaluation, we assume a tolerance threshold of −5% mIoU, where a loss in accuracy ≥ −5% mIoU is considered acceptable.

Using this tolerance threshold, SDC events can be classified as SDC_T or SDC_C events to evaluate RECON in terms of both. The probabilities of SDC events being either SDC_T or SDC_C for each



(a) (b)

Fig. 5. Impact of CRAM faults on mIoU with (a) 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.

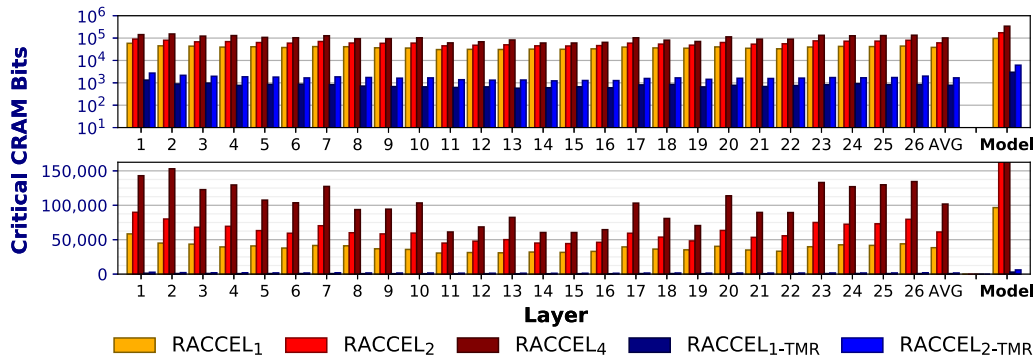


Fig. 6. Critical CRAM bits vulnerable to SDC_C by layer and model for each RACCEL configuration in (top) logarithmic and (bottom) linear scale.

module can be determined, as shown in Figure 5(b). For both $RACCEL_N$ and $RACCEL_{N\text{-TMR}}$ 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 ($RACCEL_{N\text{-TMR}}$) have greater SDC severity compared to their unmitigated counterparts ($RACCEL_N$); however, the overall reduced critical area negates this increase. For example, $RACCEL_{2\text{-TMR}}$ has greater severity than $RACCEL_2$ (47% versus 34%) but a 37× reduction in critical CRAM bits vulnerable to SDC_C.

Furthermore, a dependability evaluation of RECON at the layer-level can be useful to analyze and identify how characteristics of the model and architecture can affect the susceptibility of RECON to faults. Figure 6 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.

4.2.2 Time-varying Fault Rate Prediction. Using the SEE rate prediction methodology of Reference [43], 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 (7), 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 7 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 fault-masking 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 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.

4.2.3 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 5 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

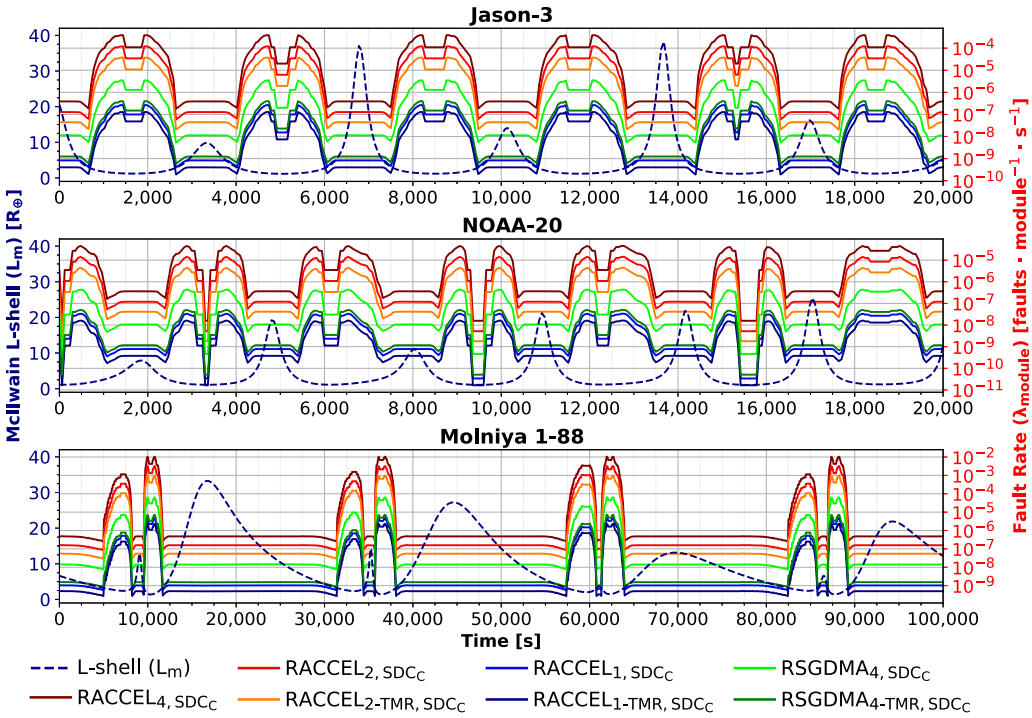


Fig. 7. Predicted McIlwain L-shell (L_m) and fault rates (λ_{module}) of multiple RECON modules for Z7020 over time for the Jason-3, NOAA-20, and Molniya 1-88 orbital case studies.

Table 5. RECON Static Modes and Adaptive Strategy

Static Mode	Configuration	Performance (FPS)	Energy Efficiency (FPS/W)	SDC_C MWTF (FPS)	SDC_C MWTF (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 $\alpha_1 \leq \alpha_2 \leq \alpha_3 \leq \cdots \leq \alpha_{N-1}$, and $[\lambda_{\min}, \lambda_{\max}]$ are the extrema of the expected range of SEE rates.

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-s intervals over a

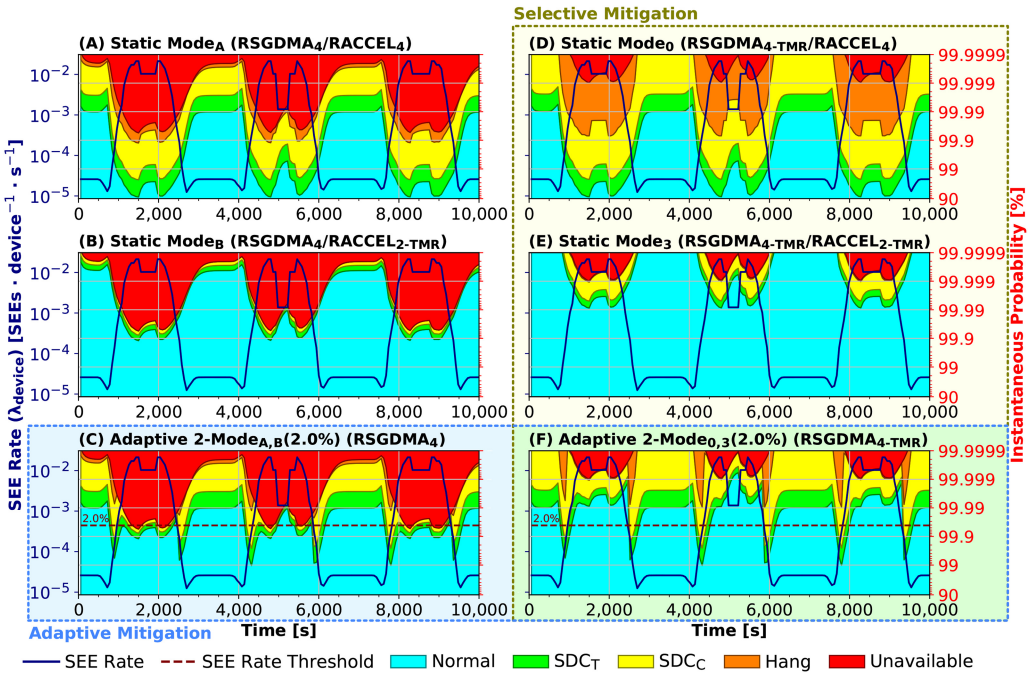


Fig. 8. Instantaneous probability of system operation for six strategies in Jason-3 orbit over time. Strategies *D*, *E*, and *F* (right column) use selective mitigation and strategies *C* and *F* (bottom row) use adaptive mitigation.

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.

Figure 8 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 8).

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

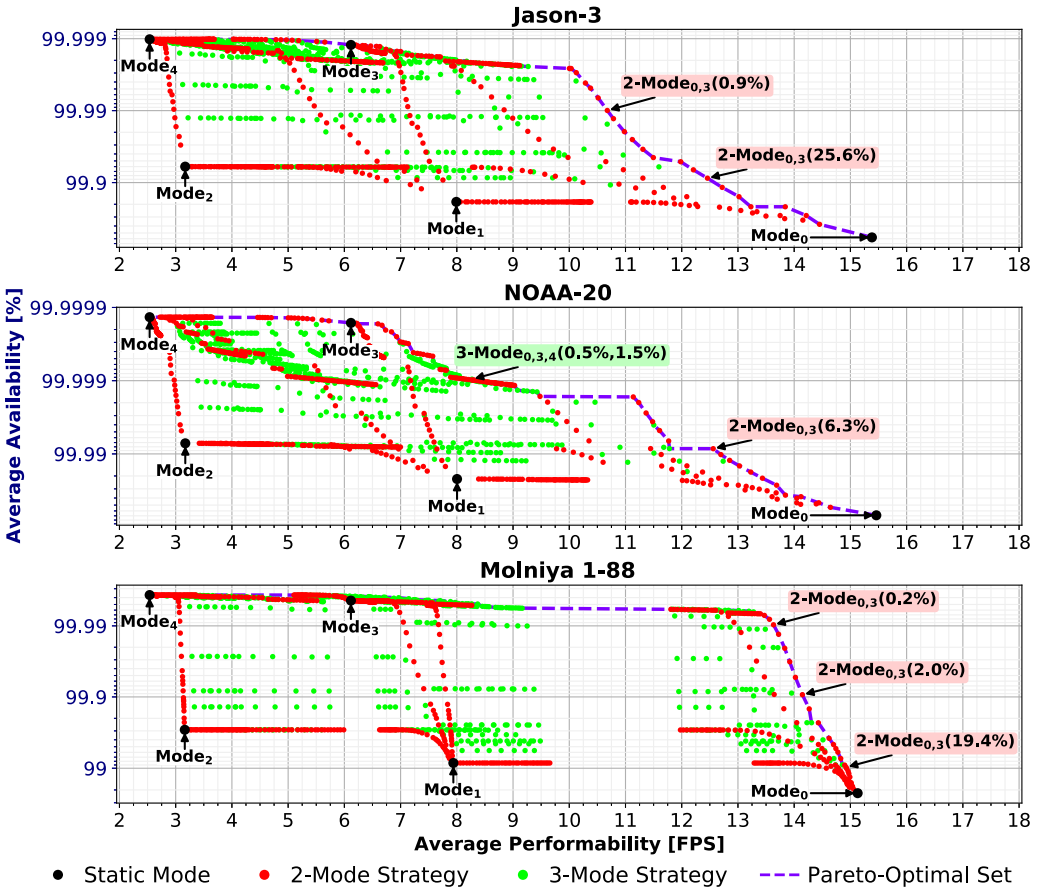


Fig. 9. RECON design tradespace in terms of performability and availability and with Pareto-optimal curves for static and adaptive strategies.

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 9 shows the design tradespace with 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 6 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 $\text{2-Mode}_{0,3}(0.9\%)$ satisfy the availability

Table 6. 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-$Mode_{0,3}$(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-$Mode_{0,3}$(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-$Mode_{1,4}$(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-$Mode_{1,4}$(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-$Mode_{0,3}$(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-$Mode_{0,3,4}$(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-$Mode_{1,4}$(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-$Mode_{1,4}$(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-$Mode_{0,3}$(19.4%)	≥99	3.54×10^{0}	1.81×10^{1}	14.92 (1.88 × $Mode_1$)	5.34 (1.52 × $Mode_1$)
2-$Mode_{0,3}$(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-$Mode_{0,3}$(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-$Mode_{1,4}$(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-$Mode_{1,4}$(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.

constraint of ≥99.99% (four nines), but 2-$Mode_{0,3}$(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-$Mode_{1,4}$(3.7%) satisfy the availability constraint of ≥99.9%, but 2-$Mode_{1,4}$(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 selected availability constraints achieved performability improvements of 1.5–3.0× for all orbital case studies.

4.2.4 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 [35] 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: $RSGDMA_{2\text{-}TMR}/RACCEL_2$ ($Mode_1$) and $RSGDMA_{2\text{-}TMR}/RACCEL_{1\text{-}TMR}$ ($Mode_4$). The experimental setup is illustrated in Figure 10. Four Zybo Z7-20 (Z7020) and two UltraZed-EG (ZU3EG) DUTs were placed in the beam to parallelize

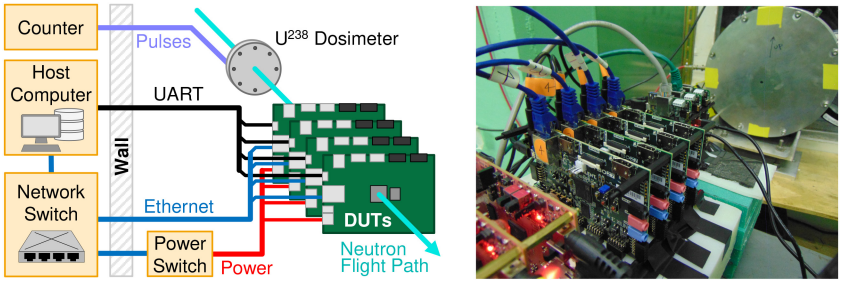


Fig. 10. Experimental setup at LANSCE.

the fluence each design was exposed 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. Because the RSGDMA uses noncoherent AXI that bypass the caches altogether, the performance characteristics are equivalent to those in Section 4.1. 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. 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 7. For both sets of DUTs, the 5× (Zynq-7000) and 2× (Zynq-MPSoC) improvement in the neutron cross-section reaffirms the dependability advantage of $\text{RACCEL}_{1\text{-TMR}}$ 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.

5 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

Table 7. RECON Wide-spectrum Neutron-beam Test Results

Platform Design	Effective Fluence ($n \cdot cm^{-2}$)	Total Runs	Observable Events SDC_T	Observable Events SDC_C	Observable Events 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} $[3.06 \times 10^{-10}, 4.33 \times 10^{-10}]$
$RSGDMA_{2\text{-}TMR}/RACCEL_{1\text{-}TMR}$	4.17×10^{11}	100,702	48	29	7	7.56×10^{-11} $[4.82 \times 10^{-11}, 1.05 \times 10^{-10}]$
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} $[7.90 \times 10^{-13}, 2.19 \times 10^{-11}]$
$RSGDMA_{2\text{-}TMR}/RACCEL_{1\text{-}TMR}$	1.50×10^{11}	135,224	0	0	1	8.01×10^{-12} $[0.00 \times 10^{-00}, 1.57 \times 10^{-11}]$
Improvement						1.97 ×

[a] Assuming one event when no events were detected [37].

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 article, 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 partial reconfiguration to alternate between configurations of the dataflow 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 performance and energy efficiency, respectively, compared to static approaches. Our evaluation, which was conducted on emulators 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.

ACKNOWLEDGMENTS

The authors thank Christopher Wilson of NASA Goddard Space Flight Center for providing guidance for this research effort, Daniel Sabogal for developing the low-level Linux software, Steve Wender and Kranti Gunthoti of LANSCE for providing guidance on ICE-II and collecting dosimetry, and the ISPRS for making available the Potsdam dataset used in this research.

REFERENCES

[1] Dimitris Agiakatsikas, Nguyen T. H. Nguyen, Zhuoran Zhao, Tong Wu, Ediz Cetin, Oliver Diessel, and Lingkan Gong. 2016. Reconfiguration control networks for TMR systems with module-based recovery. In *Proceedings of the 2016 IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM'16)*. 88–91. https://doi.org/10.1109/FCCM.2016.30

[2] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. 2017. SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* 39, 12 (2017), 2481–2495.

[3] Fabio Benevenuti, Fabiano Libano, Vincent Pouget, Fernanda Lima Kastensmidt, and Paolo Rech. 2018. Comparative analysis of inference errors in a neural network implemented in SRAM-based FPGA induced by neutron irradiation and fault injection methods. In *Proceedings of the 2018 31st Symposium on Integrated Circuits and Systems Design (SBCCI'18)*. 1–6. https://doi.org/10.1109/SBCCI.2018.8533235

[4] Melanie Berg, Christian Poivey, David Petrick, Daniel Espinosa, Austin Lesea, Kenneth A. LaBel, Mark Friendlich, Hak Kim, and Anthony Phan. 2008. Effectiveness of internal versus external SEU scrubbing mitigation strategies in a Xilinx FPGA: Design, test, and analysis. *IEEE Trans. Nucl. Sci.* 55, 4 (Aug. 2008), 2259–2266. https://doi.org/10.1109/TNS.2008.2001422

[5] Cristiana Bolchini, Antonio Miele, and Marco D. Santambrogio. 2007. TMR and partial dynamic reconfiguration to mitigate SEU faults in FPGAs. In *Proceedings of the 22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT'07)*. 87–95. https://doi.org/10.1109/DFT.2007.25

[6] Sébastien Bourdarie and Michael Xapsos. 2008. The near-earth space radiation environment. *IEEE Trans. Nucl. Sci.* 55, 4 (Aug 2008), 1810–1832. https://doi.org/10.1109/TNS.2008.2001409

[7] Cody Brewer, Nicholas Franconi, Robin Ripley, Alessandro Geist, Travis Wise, Sebastian Sabogal, Gary Crum, Sabrena Heyward, and Christopher Wilson. 2020. 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*. AIAA, 1–6.

[8] Michael J. Campola and Jonathan A. Pellish. 2019. Radiation hardness assurance: Evolving for newspace. In *Proceedings of the 2019 RADiations Effects on Components and Systems (RADECS'19) Short Course, Part V*. 1–35.

[9] BAA DARPA. 2018. Blackjack (BAA HR001118S0032). DARPA.

[10] BAA DARPA. 2019. Blackjack Pit Boss (BAA HR001119S0012). DARPA.

[11] Fernando Fernandes dos Santos, Caio Lunardi, Daniel Oliveira, Fabiano Libano, and Paolo Rech. 2019. Reliability evaluation of mixed-precision architectures. In *Proceedings of the 2019 IEEE International Symposium on High Performance Computer Architecture (HPCA'19)*. 238–249. https://doi.org/10.1109/HPCA.2019.00041

[12] Boyang Du, Sarah Azimi, Corrado de Sio, Ludovica Bozzoli, and Luca Sterpone. 2019. On the reliability of convolutional neural network implementation on SRAM-based FPGA. In *Proceedings of the 2019 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT'19)*. 1–6. https://doi.org/10.1109/DFT.2019.8875362

[13] Giulio Gambardella, Johannes Kappauf, Michaela Blott, Christoph Doehring, Martin Kumm, Peter Zipf, and Kees Vissers. 2019. Efficient error-tolerant quantized neural network accelerators. In *Proceedings of the 2019 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT'19)*. 1–6. https://doi.org/10.1109/DFT.2019.8875314

[14] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, Pablo Martinez-Gonzalez, and Jose Garcia-Rodriguez. 2018. A survey on deep learning techniques for image and video semantic segmentation. *Appl. Soft Comput.* 70 (2018), 41–65. https://doi.org/10.1016/j.asoc.2018.05.018

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

[16] Alan D. George and Christopher M. Wilson. 2018. Onboard processing with hybrid and reconfigurable computing on small satellites. *Proc. IEEE* 106, 3 (Mar. 2018), 458–470. https://doi.org/10.1109/JPROC.2018.2802438

[17] Kaiyuan Guo, Shulin Zeng, Jincheng Yu, Yu Wang, and Huazhong Yang. 2019. [DL] A survey of FPGA-based neural network inference accelerators. *ACM Trans. Reconfig. Technol. Syst.* 12, 1, Article 2 (Mar. 2019), 26 pages. https://doi.org/10.1145/3289185

[18] Felix R. Hoots and Ronald L. Roehrich. 1980. *Models for Propagation of NORAD Element Sets*. Technical Report. Aerospace Defense Command Peterson AFB, Office of Astrodynamics.

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

[20] Adam Jacobs, Grzegorz Cieslewski, Alan D. George, Ann Gordon-Ross, and Herman Lam. 2012. Reconfigurable fault tolerance: A comprehensive framework for reliable and adaptive FPGA-based space computing. *ACM Trans. Reconfig. Technol. Syst.* 5, 4, Article 21 (Dec. 2012), 30 pages. https://doi.org/10.1145/2392616.2392619

[21] Jonathan M. Johnson and Michael J. Wirthlin. 2010. 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)*. ACM, New York, NY, 249–258. https://doi.org/10.1145/1723112.1723154

[22] Kenneth A. LaBel and Jonathan A. Pellish. 2014. National radiation hardness assurance (RHA) planning for NASA missions: Updated guidance. NASA Electronic Parts and Packaging Program (NEPP) (March 2014).
[23] Fahad Lateef and Yassine Ruichek. 2019. Survey on semantic segmentation using deep learning techniques. *Neurocomputing* 338 (2019), 321–348. https://doi.org/10.1016/j.neucom.2019.02.003
[24] Andrew Lavin and Scott Gray. 2016. Fast algorithms for convolutional neural networks. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'16)*. 4013–4021.
[25] Robert Le. 2012. Soft error mitigation using prioritized essential bits. Xilinx XAPP538 (v1. 0).
[26] Fabiano Libano, Brittany Wilson, Jon-Paul Anderson, Michael J. Wirthlin, Carlo Cazzaniga, Christopher Frost, and Paolo Rech. 2019. Selective hardening for neural networks in FPGAs. *IEEE Trans. Nucl. Sci.* 66, 1 (Jan. 2019), 216–222. https://doi.org/10.1109/TNS.2018.2884460
[27] Fabiano Libano, Brittany Wilson, Michael Wirthlin, Paolo Rech, and John Brunhaver. 2020. Understanding the impact of quantization, accuracy, and radiation on the reliability of convolutional neural networks on FPGAs. *IEEE Trans. Nucl. Sci.* 67, 7 (Jul. 2020), 1478–1484. https://doi.org/10.1109/TNS.2020.2983662
[28] Tyler M. Lovelly and Alan D. George. 2017. Comparative analysis of present and future space-grade processors with device metrics. *J. Aerosp. Inf. Syst.* 14, 3 (01 Mar. 2017), 184–197. https://doi.org/10.2514/1.I010472
[29] David J. Miranda. 2020. 2020 NASA technology taxonomy: 2015 Technology areas to 2020 taxonomy areas crosswalk (HQ-E-DAA-TN76653). NASA.
[30] Sparsh Mittal. 2020. A survey of FPGA-based accelerators for convolutional neural networks. *Neural Comput. Appl.* 32, 4 (01 Feb. 2020), 1109–1139. https://doi.org/10.1007/s00521-018-3761-1
[31] Shubhendu S. Mukherjee, Christopher Weaver, Joel Emer, Steven K. Reinhardt, and Todd Austin. 2003. A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor. In *Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-36)*. 29–40. https://doi.org/10.1109/MICRO.2003.1253181
[32] National Academies of Sciences, Engineering, and Medicine. 2016. *Achieving Science with CubeSats: Thinking Inside the Box*. The National Academies Press, Washington, DC. https://doi.org/10.17226/23503
[33] National Academies of Sciences, Engineering, and Medicine. 2018. *Testing at the Speed of Light: The State of U.S. Electronic Parts Space Radiation Testing Infrastructure*. The National Academies Press, Washington, DC. https://doi.org/10.17226/24993
[34] National Academies of Sciences, Engineering, and Medicine. 2018. *Thriving on Our Changing Planet: A Decadal Strategy for Earth Observation from Space*. The National Academies Press, Washington, DC. https://doi.org/10.17226/24938
[35] Suzanne F. Nowicki, Stephen A. Wender, and Michael Mocko. 2017. The Los Alamos Neutron Science Center spallation neutron sources. *Phys. Proc.* 90 (2017), 374–380. https://doi.org/10.1016/j.phpro.2017.09.035
[36] ISPRS Potsdam. 2018. 2D Semantic Labeling Dataset. Retrieved from http://www2.isprs.org/commissions/comm3/wg4/2d-sem-label-potsdam.html.
[37] Heather Quinn. 2014. Challenges in testing complex systems. *IEEE Trans. Nucl. Sci.* 61, 2 (Apr. 2014), 766–786. https://doi.org/10.1109/TNS.2014.2302432
[38] Heather Quinn. 2017. Radiation effects in reconfigurable FPGAs. *Semiconduct. Sci. Technol.* 32, 4 (Mar. 2017), 1–8. https://doi.org/10.1088/1361-6641/aa57f6
[39] George A. Reis, Jonathan Chang, Neil Vachharajani, Shubhendu S. Mukherjee, Ram Rangan, and David I. August. 2005. Design and evaluation of hybrid fault-detection systems. In *Proceedings of the 32nd International Symposium on Computer Architecture (ISCA'05)*. 148–159. https://doi.org/10.1109/ISCA.2005.21
[40] Seth Roffe, Theodore Schwarz, Thomas Cook, Noah Perryman, Justin Goodwill, Evan Gretok, Aidan Phillips, Mitchell Moran, Tyler Garrett, and Alan George. 2020. CASPR: Autonomous sensor processing experiment for STP-H7. In *Proceedings of the 34th Annual AIAA/USU Conference on Small Satellites*. AIAA, 1–11.
[41] Sebastian Sabogal, Patrick Gauvin, Brad Shea, Daniel Sabogal, Antony Gillette, Christopher Wilson, Ansel Barchowsky, Alan D. George, Gary Crum, and Thomas Flatley. 2017. SSIVP: Spacecraft supercomputing experiment for STP-H6. In *Proceedings of the 31st Annual AIAA/USU Conference on Small Satellites*. AIAA, 1–12.
[42] Sebastian Sabogal, Alan George, and Gary Crum. 2019. ReCoN: A reconfigurable CNN acceleration framework for hybrid semantic segmentation on hybrid SoCs for space applications. In *Proceedings of the 2019 IEEE Space Computing Conference (SCC'19)*. 41–52. https://doi.org/10.1109/SpaceComp.2019.00010
[43] Sebastian Sabogal, Alan George, and Christopher Wilson. 2020. Reconfigurable framework for environmentally adaptive resilience in hybrid space systems. *ACM Trans. Reconfig. Technol. Syst.* 13, 3, Article 14 (Jul. 2020), 1–32. https://doi.org/10.1145/3398380
[44] Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural Netw.* 61 (2015), 85–117. https://doi.org/10.1016/j.neunet.2014.09.003
[45] Felix Siegle, Tanya Vladimirova, Jørgen Ilstad, and Omar Emam. 2015. Mitigation of radiation effects in SRAM-based FPGAs for space applications. *ACM Comput. Surv.* 47, 2, Article 37 (Jan. 2015), 34 pages. https://doi.org/10.1145/2671181

[46] Aaron Stoddard, Ammon Gruwell, Peter Zabriskie, and Michael J. Wirthlin. 2017. A hybrid approach to FPGA configuration scrubbing. *IEEE Trans. Nucl. Sci.* 64, 1 (Jan. 2017), 497–503. https://doi.org/10.1109/TNS.2016.2636666

[47] Michael A. Swartout. 2020. CubeSats mission assurance trends. In *Proceedings of the NASA Electronic Parts and Packaging (NEPP) Electronics Technology Workshop (ETW'20)*. NASA GSFC, Greenbelt, MD.

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

[49] César Torres-Huitzil and Bernard Girau. 2017. Fault tolerance in neural networks: Neural design and hardware implementation. In *Proceedings of the 2017 International Conference on ReConFigurable Computing and FPGAs (ReConFig'17)*. 1–6. https://doi.org/10.1109/RECONFIG.2017.8279793

[50] 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. 1997. CREME96: A revision of the cosmic ray effects on micro-electronics code. *IEEE Trans. Nucl. Sci.* 44, 6 (Dec. 1997), 2150–2160. https://doi.org/10.1109/23.659030

[51] Ingo Wardinski, Diana Saturnino, Hagay Amit, Aude Chambodut, Benoit Langlais, Mioara Mandea, and Thébault Erwan. 2020. Geomagnetic core field models and secular variation forecasts for the 13th International Geomagnetic Reference Field (IGRF-13). *Earth Planets Space* 72, 1 (22 Oct. 2020), 1–155. https://doi.org/10.1186/s40623-020-01254-7

[52] Xuechao Wei, Cody Hao Yu, Peng Zhang, Youxiang Chen, Yuxin Wang, Han Hu, Yun Liang, and Jason Cong. 2017. Automated systolic array architecture synthesis for high throughput CNN inference on FPGAs. In *Proceedings of the 54th Annual Design Automation Conference (DAC'17)*. Association for Computing Machinery, New York, NY, Article 29, 6 pages. https://doi.org/10.1145/3061639.3062207

[53] Caleb Williams and Stephanie DelPozzo. 2020. 2020 Nano/microsatellite market forecast2020 Nano/Microsatellite Market Forecast, 10th ed. SpaceWorks Enterprises, Inc.

[54] Christopher Wilson and Alan D. George. 2018. CSP Hybrid space computing. *J. Aerosp. Inf. Syst.* 15, 4 (2 Feb. 2018), 215–227. https://doi.org/10.2514/1.I010572

[55] Michael Wirthlin. 2015. High-reliability FPGA-based systems: Space, high-energy physics, and beyond. *Proc. IEEE* 103, 3 (Mar. 2015), 379–389. https://doi.org/10.1109/JPROC.2015.2404212

[56] Michael A. Xapsos, Patrick M. O'Neill, and T. Paul O'Brien. 2013. Near-earth space radiation models. *IEEE Trans. Nucl. Sci.* 60, 3 (June 2013), 1691–1705. https://doi.org/10.1109/TNS.2012.2225846

[57] Xilinx. 2018. *Zynq-7000 SoC Technical Reference Manual* (v1.12.2 ed.). Xilinx User Guide (UG585).

[58] Xilinx. 2019. *Zynq UltraScale+ Device Technical Reference Manual* (v2.1 ed.). Xilinx User Guide (UG1085).

[59] Zhuoran Zhao, Dimitris Agiakatsikas, Nguyen T. H. Nguyen, Ediz Cetin, and Oliver Diessel. 2016. Fine-grained module-based error recovery in FPGA-based TMR systems. In *Proceedings of the 2016 International Conference on Field-Programmable Technology (FPT'16)*. 101–108. https://doi.org/10.1109/FPT.2016.7929433

[60] Zhuoran Zhao, Nguyen T. H. Nguyen, Dimitris Agiakatsikas, Ganghee Lee, Ediz Cetin, and Oliver Diessel. 2018. Fine-grained module-based error recovery in FPGA-based TMR systems. *ACM Trans. Reconfig. Technol. Syst.* 11, 1, Article 4 (Jan. 2018), 1–23 pages. https://doi.org/10.1145/3173549

Received February 2021; revised June 2021; accepted June 2021