# ReCoN: A Reconfigurable CNN Acceleration Framework for Hybrid Semantic Segmentation on Hybrid SoCs for Space Applications

Sebastian Sabogal, Alan D. George
*NSF SHREC Center*
*University of Pittsburgh*
*Pittsburgh, PA, USA*
{*sebastian.sabogal,alan.george*}*@pitt.edu*

Gary A. Crum
*Science Data Processing Branch (Code 587)*
*NASA Goddard Space Flight Center*
*Greenbelt, MD, USA*
*gary.a.crum@nasa.gov*

***Abstract*—Recent advancements in deep learning present new opportunities for enhanced scientific methods, autonomous operations, and intelligent applications for space missions. Semantic segmentation is a powerful computer-vision process using convolutional neural networks (CNNs) to classify objects within an image. Semantic segmentation has numerous space-science and defense applications, from semantic labeling of Earth observations for insights about our changing planet, to monitoring natural disasters for damage control, to gathering intelligence for national defense and security. Despite these advantages, CNNs can be computationally expensive and prohibited on traditional radiation-hardened space processors, which are often generations behind their commercial-off-the-shelf counterparts in terms of performance and energy-efficiency. FPGA-based hybrid System-on-Chips (SoCs), which combine fixed-logic CPUs with reconfigurable-logic FPGAs, present numerous architectural advantages well-suited to address the computational capabilities required for high-performance, intelligent spacecraft. To enable semantic segmentation for on-board space processing, we propose a hybrid (hardware/software partitioned) approach using our reconfigurable CNN accelerator (ReCoN) for accelerating CNN inference on hybrid SoCs. When evaluated on the Xilinx Zynq SoC and Xilinx Zynq UltraScale+ MPSoC platforms, our hybrid approach demonstrates an improvement in performance and energy-efficiency up to two orders of magnitude compared to a software-only baseline on the hybrid SoC. Furthermore, fault injection and wide-spectrum neutron beam-testing was performed to characterize the ReCoN architectural response to injected errors and susceptibility to neutron irradiation.**

***Keywords*-Deep Learning; Convolutional Neural Networks; Semantic Segmentation; Hybrid System-on-Chip; Hybrid Space Computing; Fault Injection; Radiation-beam Testing**

## I. Introduction

Recent advancements in deep learning present new opportunities to enhance scientific methods, autonomous operations, and intelligent applications for space missions. The National Academies' Space Studies Board (SSB) issued a report for the 2017-2027 decadal strategy on Earth science and applications from space, providing recommendations for NASA, NOAA, and USGS. In their survey, the SSB highlighted the need for advanced methodologies to analyze and convert data from Earth observations (EO) into scientific knowledge [1]. The SSB also identified machine learning as a scientific and technological opportunity to extend the reach of Earth science through more efficient uses of limited resources. Semantic segmentation is a deep-learning algorithm, based on convolutional neural networks (CNNs), that learns to infer dense labels for every pixel of an image. Semantic segmentation has numerous space applications, from semantic labeling of Earth's features for insights about our changing planet, to monitoring natural disasters, to gathering intelligence for national security.

Due to ongoing innovations in both sensor technology and spacecraft autonomy, on-board space processing continues to be outpaced by the computational demands required for future missions. The application of deep-learning concepts for on-board processing can enable spacecraft to efficiently process immense volumes of raw sensor-data into actionable data to overcome limitations in downlink communication. However, spacecraft designers are challenged to create high-performance, intelligent space computers subject to unique requirements, with stringent constraints in size, weight, power, and cost (SWaP-C), and unique hazards, including radiation, thermal, vibration, and vacuum. Spacecraft often employ radiation-hardened (rad-hard) processors to satisfy reliability constraints and overcome space radiation challenges. However, rad-hard processors are often generations behind their commercial-off-the-shelf (COTS) counterparts, which tend to offer superior performance and energy-efficiency but are more susceptible to space radiation. Despite the high-applicability of deep learning for spaceflight, advanced deep-learning algorithms, such as semantic segmentation, are computationally expensive and prohibited on traditional rad-hard processors. Currently, the application of deep-learning for space missions rely on high-performance computing (HPC) resources, such as GPU clusters, to analyze downlinked data.

Small satellites (SmallSats) and CubeSats are small form-factor spacecraft emerging as high-risk, low-cost platforms enabled by the miniaturization of electronics, sensors, and instruments. In their 2016 report, the SSB identified CubeSats as a disruptive innovation for space-science technology and concluded that CubeSat missions were already meeting

978-1-7281-3194-8/19/$31.00 ©2019 IEEE
DOI 10.1109/SpaceComp.2019.00010

valuable science objectives [2]. SmallSats have proliferated substantially in the academic, commercial, and government sectors, and several missions for a wide variety of science and technology applications have launched or are planned. SmallSat technology is also emerging in future defense missions. DARPA recently released a Broad Agency Announcement for the Blackjack program. In this program, DARPA seeks to develop a next-generation avionics unit, called Pit Boss, which will leverage commodity and commercial technology to enable advanced on-orbit edge computing and mission autonomy. Blackjack will demonstrate that a constellation of autonomous, low-cost, replenishable SmallSats residing in low-Earth orbit can compete with monolithic spacecraft residing in geosynchronous orbit [3].

As both high-risk and low-cost platforms, SmallSats and CubeSats often employ commercial technology, such as embedded system-on-chip (SoC) devices, providing an advantage for improved on-board processing. Hybrid SoCs combine two or more distinct computing architectures (e.g., CPUs, FPGAs, GPUs) into one device to attain the advantages of each. FPGA-based hybrid SoCs, such as the Xilinx Zynq SoC (Zynq7) and Zynq UltraScale+ MPSoC (ZynqMP), combine fixed-logic CPUs with reconfigurable-logic FPGAs. Hybrid SoCs enable hardware/software partitioning of applications, where sequential, control-flow parts are executed in software on the CPU and parallel, data-flow parts are accelerated in hardware on the FPGA. The hardware/software co-design of these hybrid applications can enable significant improvements in performance and energy-efficiency. Hybrid SoCs present numerous architectural advantages that make them well-suited to address the on-board processing challenges for space missions [4].

In this article, we propose a hybrid approach to semantic segmentation for on-board processing. Our hybrid approach combines an adaptive framework and reconfigurable CNN accelerator, called ReCoN, to accelerate semantic segmentation on hybrid SoCs [5]. When evaluated on the Xilinx Zynq SoC and Xilinx Zynq UltraScale+ MPSoC platforms, our hybrid approach demonstrates a substantial improvement in performance and energy-efficiency up to two orders of magnitude compared to a single-threaded, software-only baseline executed on the hybrid SoC. Fault-injection and radiation-beam experiments were also performed to characterize the ReCoN architectural response to injected errors and susceptibility to neutron radiation.

The remainder of this article is organized as follows. Section II provides an overview of hybrid SoC technology for space applications, CNN basics, semantic segmentation, and related works. Section III describes our hybrid approach, which includes a system framework, control-flow software, and ReCoN accelerator architecture. Section IV evaluates our hybrid approach for semantic segmentation in terms of prediction accuracy, resource utilization, performance, and energy-efficiency. Section V provides a description and analysis of fault-injection and radiation-beam test experiments conducted for ReCoN. Finally, Section VI provides concluding statements and insights for future work.

## II. Background

This section provides a cursory overview of space computing challenges, hybrid SoCs and their application to space computing, space radiation effects, fundamentals of convolutional neural networks, and semantic segmentation. Finally, we summarize related works that inspired this research effort.

### A. Hybrid SoCs for Space Systems

The Xilinx Zynq SoC (Zynq7) and Xilinx Zynq UltraScale+ MPSoC (ZynqMP) are two commercial families of hybrid SoCs [6], [7]. The Zynq7 devices feature up to dual-core ARM Cortex-A9 CPU and 28-nm Artix or Kintex 7-Series FPGA fabric, and the ZynqMP devices feature up to quad-core ARM Cortex-A53 CPU and 16-nm UltraScale architecture FPGA fabric. Platforms for the Zynq7 include the Xilinx ZC706 (Z7040), TUL PYNQ-Z2 (Z7020), and Digilent ZedBoard (Z7020), and for the ZynqMP include the Xilinx ZCU102 (ZU9EG) and Avnet UltraZed-EG (ZU3EG). Within each family, these devices share similar characteristics (e.g., performance, power consumption, and susceptibility to radiation) because they have the same architecture and process technology, but varied quantities of available resources. In our evaluation, we use a diverse set of platforms, but the experimental results can be used interchangeably with other devices of the same family. In both families of hybrid SoCs, the CPU and FPGA subsystems can interact over general-purpose and high-performance Advanced eXtensible Interconnect (AXI) interfaces. Both devices are capable of dynamic partial reconfiguration (PR), which enables predefined partitions of the FPGA, called PR regions (PRRs), to be reconfigured with other compatible modules, called PR modules (PRMs), at run-time without interrupting the remainder of the system (e.g., CPU and other FPGA subsystems).

Hybrid SoCs are becoming increasingly adopted for space missions. One example is the CHREC Space Processor v1 (CSPv1), a multifaceted-hybrid space computer developed at the NSF Center for Space, High-performance, and Resilient Computing (SHREC) with close collaboration with NASA Goddard Space Flight Center (GSFC) [8]. The CSPv1 space computer features the Z7020 and combines a novel mix of commercial technology (processor and memory for performance and energy-efficiency benefits), rad-hard technology (monitoring and managing circuits for reliability), and supplementary fault-tolerant computing for extended reliability enhancements. CSPv1 has flight heritage as part of two U.S. Department of Defense Space Test Program (STP) - Houston missions to the International Space Station (ISS). These missions include the STP-H5 CHREC Space Processor

(STP-H5-CSP) and STP-H6 Spacecraft Supercomputing for Image and Video Processing (STP-H6-SSIVP) [9] experiments. CSPv1 was flown on the NASA CeREs heliophysics-science CubeSat and will be featured on the Lockheed-Martin LunIR lunar-flyby CubeSat, the NASA Mass Spectrometer observing lunar operations (MSolo) instrument, and several other planned missions. Other space computers based on the Zynq7 devices include Innoflight's Compact Flight Computer (CFC-300), GomSpace's Nanomind Z7000, and Xiphos' Q7. Space computers based on the ZynqMP devices include Innoflight's Compact Heterogeneous-processor Array for Multi-Parametric Sensing (CHAMPS), and Xiphos' Q8.

In [10], a framework was developed for analyzing potential processor architectures for on-board space computing. Using this framework, the *computational density* (CD), measured in giga operations per second (GOPS), and *computational density per Watt* (CD/W) were calculated for the Z7020 and several state-of-the-art rad-hard processors. In this comparison, the Z7020 demonstrated significant improvements versus the rad-hard processors in both metrics. Due to the immense computation demands required for deep-learning algorithms, which are not achievable by currently available rad-hard processors, semantic segmentation is only viable on space platforms by leveraging the performance benefits of SoCs and relying on fault-tolerant mitigation techniques for radiation effects.

### B. Radiation Effects

In the near-Earth space environment, radiation sources include galactic cosmic rays, solar particle events, and charged particles trapped within the Van Allen radiation belts. Radiation presents numerous challenges for electronic devices in space [11]. Radiation effects include long-term cumulative effects, such as total ionizing dose and displacement damage dose, and transient single-event effects (SEEs). Non-destructive SEEs include upsets, transients, and functional interrupts. These effects are extensively covered in [12]. Radiation-beam testing is often exercised to characterize device susceptibility to radiation, and to determine whether the device is suitable for the space radiation environment.

### C. Convolutional Neural Networks (CNNs)

Convolutional neural networks (CNNs) have become increasingly popular in the computer-vision community for classification, detection, localization, and segmentation applications. CNNs are a form of classical supervised learning algorithms with a feed-forward process for inference and a backpropagation process for training [13]. CNNs typically contain convolutional, activation, pooling, and fully connected layers. Convolutional layers are used for extracting features in the input and producing feature maps for subsequent layers. Each convolution operation contains a set of learnable weights, the kernel and bias, which are

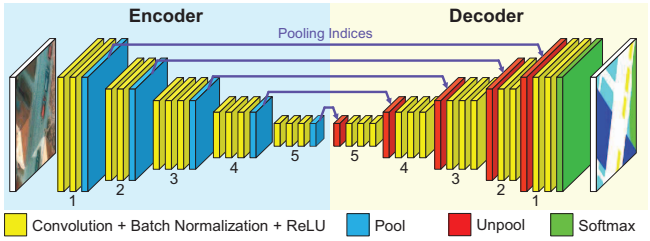


Figure 1. SegNet model.

formulated during training. The initial convolutional layers detect low-level features (e.g., lines, corners, etc.) and the deeper layers extract more complex structures and patterns. Activation layers are used to introduce nonlinearity into the network to allow for the approximation of nonlinear patterns. Examples of activation functions include sigmoid, tanh, and rectified linear unit (ReLU), with ReLU often preferred for faster training [14]. Pooling layers are used to downsample the spatial resolution of the input to reduce the number of parameters and amount of processing. Examples of pooling functions include max-pooling and average-pooling. The fully connected layer, often at the end of the CNN, performs classification and maps features extracted from previous layers into an output vector of classes. The arguments of the maxima (argmax) specify the most probable classification of the input and a class label is assigned. CNNs may append a softmax layer to convert the output vector into a discrete probability distribution vector specifying the confidence of the classification. Batch normalization (batch-norm) is another layer that may be inserted between convolutional and activation layers to accelerate training by normalizing and scaling the inputs to reduce the covariate shift [15].

### D. Semantic Segmentation

Semantic segmentation is a computer-vision process that learns to label each pixel of an image, where pixels with the same label share semantic characteristics. We selected *SegNet* [16] as the baseline model for evaluating our hybrid approach. SegNet uses pooling indices obtained from max-pooling layers for upsampling feature maps in max-unpooling layers, removing the need for fully connected layers. As a result, the SegNet model substantially reduces the number of weights and functions to be accelerated, which is desirable to resource-constrained systems. The SegNet model has been applied to semantic segmentation of EO imagery in [17], which we leverage as a case-study to facilitate the evaluation of our hybrid approach.

SegNet uses an encoder-decoder network architecture, as illustrated in Figure 1. SegNet is symmetrical and contains five encoder and decoder blocks, each with two or three convolutional layers followed by batch-norm and a ReLU operation. Each encoder block is followed by a max-pooling layer which produces two outputs: discretized feature-maps and pooling indices. Each decoder block begins with a

max-unpooling layer which uses the pooling indices of the corresponding encoder block to upsample smaller feature-maps back to the original spatial resolution. An optional softmax layer can be appended at the end of the network to convert the output volume of the final convolution layer into a volume where each pixel contains decimal probabilities about its classification. The argmax of the output layer can also be used to assign the most probable label for each pixel.

### E. Related Works

The acceleration of CNNs on FPGAs has been explored extensively in the literature [18], [19], [20], [21]. Prior works explored data-path optimizations, approximate computing, and batch computing techniques to accelerate CNN functions. CNNs are highly parallelizable, however, they cannot be fully unrolled on FPGAs due to resource limitations [18]. Instead of loop unrolling, the alternative is to map a reusable subset of the CNN operations into the FPGA, and iteratively stream data through them. Finding the optimal accelerator configuration can be reduced into a loop optimization problem [19]. In [19], a roofline model was proposed to explore the design space for possible solutions for a CNN architecture on an FPGA platform, subject to computational resource and memory bandwidth constraints, using loop optimization techniques such as loop unrolling, tiling, and interchange. Batch computing involves processing multiple images in a batch. In [20], batch processing was employed to reuse convolution weights on multiple images to reduce memory accesses for improved bandwidth at the cost of increased latency for each image in the batch. Approximate computing techniques involve quantizing feature maps and trained weights into fixed-point representations for improved performance and energy-efficiency at the minimal cost of decreased CNN accuracy [18].

## III. Hybrid Approach

This section provides a system description for our hybrid approach to accelerating semantic segmentation on hybrid SoCs for on-board processing. The system includes an environmentally adaptive framework and the hybrid semantic segmentation application. The hybrid application includes the ReCoN accelerator and direct memory access (DMA) controller, for accelerating data-flow parts, and the ReCoN control-software, which performs the control-flow parts.

### A. Reconfigurable Framework

Our hybrid approach leverages the Hybrid Adaptive Reconfigurable Fault Tolerance (HARFT) architecture, an environmentally adaptive framework for hybrid SoCs, as illustrated in Figure 2. The HARFT architecture provides an infrastructure for system-configuration management and hardware acceleration [22]. HARFT uses PR to dynamically reconfigure between various accelerators at run-time without

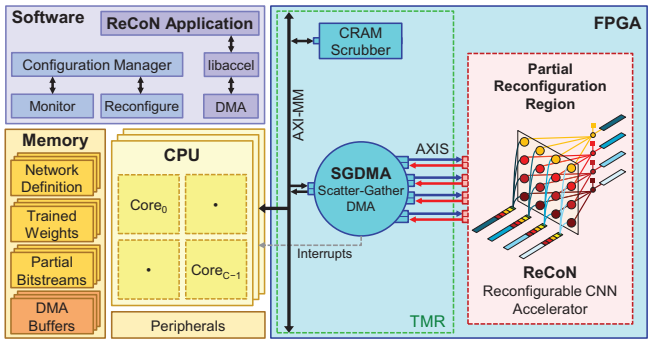


Figure 2. HARFT architecture.

interrupting system uptime. For space applications, the subsystems in the static region are protected with triple-modular redundancy (TMR), which triplicates circuits in the FPGA with majority voting for fault masking. The configuration memory (CRAM), which stores the configuration bitstream at run-time to realize the FPGA design, is also protected with CRAM scrubbing, which periodically scans static CRAM bits to correct and prevent the accumulation of errors. The PRRs can be configured with simplex, high-performance or TMR, low-performance accelerators, both also protected by CRAM scrubbing.

The acceleration framework provides a full hardware/software stack enabling userspace software applications with shared access to FPGA hardware accelerators. The FPGA hardware portion of this framework includes DMAs residing in the static region, and accelerators residing in the PRRs. The software portion includes the Linux device drivers for the DMA, *libaccel* (custom userspace library for interfacing with DMA), and userspace applications. For our hybrid semantic segmentation application, ReCoN resides in a single PRR and interfaces with a custom scatter-gather DMA (SGDMA) via AXI-Stream interconnects. The ReCoN control-software runs on the CPU and interfaces with *libaccel* to operate the SGDMA.

### B. ReCoN Accelerator Architecture

ReCoN is designed for scalability and parameterization to accommodate various hybrid SoC platforms and application domains. ReCoN is a streaming accelerator that interfaces to the SGDMA to accelerate CNN functions on multi-dimensional data in parallel and also integrates multiple CNN functions into one accelerator.

*1) Scalability and Parameterization:* ReCoN supports numerous pre-synthesis and run-time parameters to accommodate various target platforms and application domains. Pre-synthesis parameters include the *scale* and *quantization*. The scale parameter specifies the parallelization factor of the accelerator, which refers to the number of parallel channels packed into the stream and the number of parallel accelerator functions. Increasing the scale parameter improves performance and energy-efficiency but increases the area overhead.

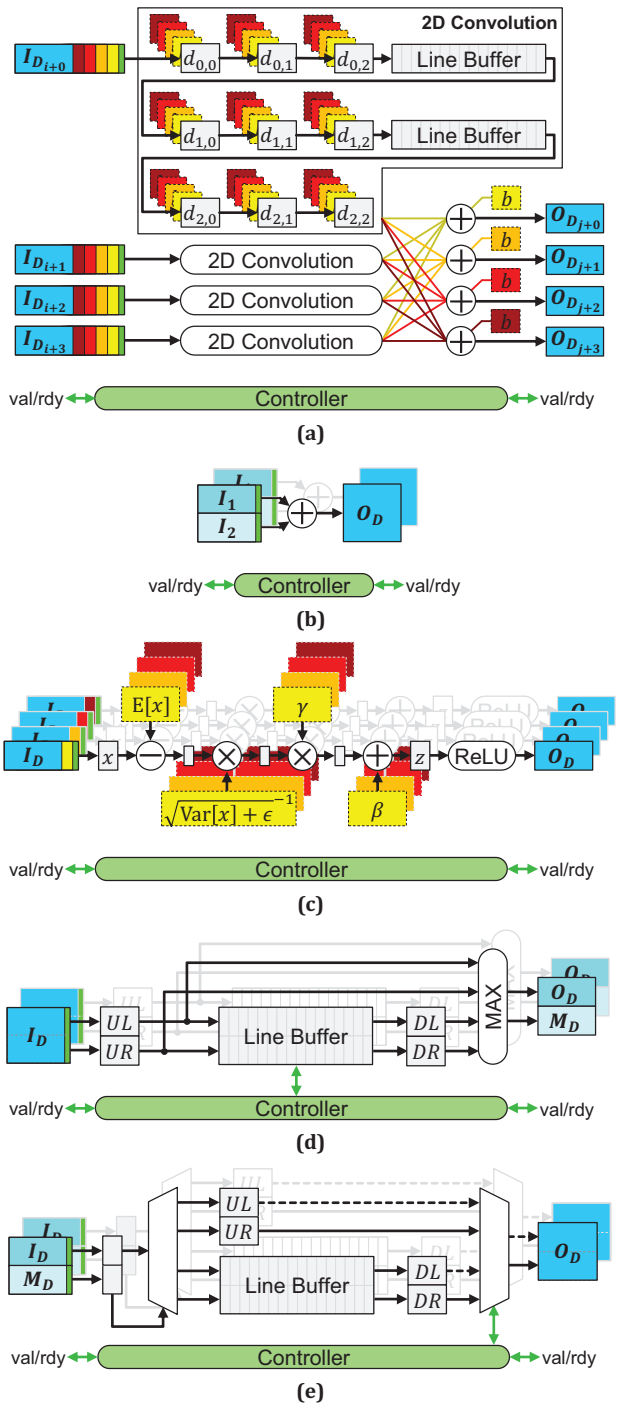


Figure 3. $\text{ReCoN}_4$ accelerator functions: (a) convolution, (b) vector sum, (c) batch-norm and ReLU, (d) max-pooling, and (e) max-unpooling.

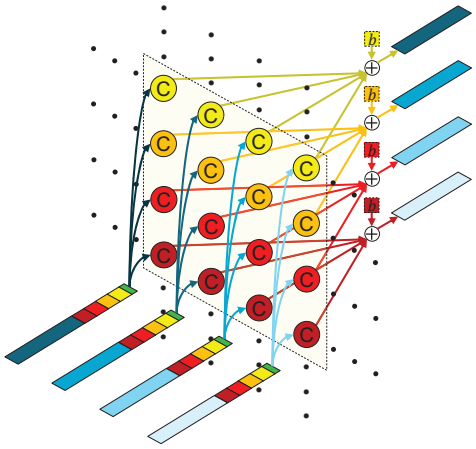

Figure 4. $4\times4$ convolution function.

For the remainder of this article, we use the subscript notation ($\text{ReCoN}_N$) to denote that the ReCoN accelerator is scaled by a factor of $N$. The quantization parameter specifies the data-type representation used by ReCoN, which includes single-precision floating-point or arbitrary-precision fixed-point. Arbitrary-precision fixed-point provides substantial improvements in area for a minimal trade-off in inference accuracy. Quantization optimizations are discussed later in this section. ReCoN is generated using Vivado High-Level Synthesis (HLS), which is a high-productivity tool for translating synthesizable functions written in high-level programming languages (such as C or C++) into a register-level transfer (RTL) representation for FPGAs. Using Vivado HLS, numerous compiler directives are available to further tune ReCoN, such as trading between resource sharing, improved timing, and area.

ReCoN is also designed to support numerous run-time parameters to accommodate various network shapes, trained weights, and data volumes. ReCoN operates on AXI-Stream packets. The packet structure includes three sections: an *accelerator header*, with arguments for input resolution and accelerator function, a *function-specific section*, with trained weights for convolution and batch-norm functions only, and finally the *data section* containing feature maps.

*2) Accelerator Functions:* ReCoN consolidates multiple functions into one accelerator, with each one having equal data-widths for both input and output streaming interfaces. $\text{ReCoN}_N$ includes:

- One $N\times N$ convolution function with $N$ single-pixel input and $N$ single-pixel output channels
- $\frac{N}{2}$ vector-sum functions, each with two single-pixel input and one double-pixel output channels
- $N$ batch-norm and ReLU functions, each with one single-pixel input and one single-pixel output channels
- $\frac{N}{2}$ max-pool functions, each with one double-pixel input and two single-pixel output channels
- $\frac{N}{2}$ max-unpool functions, each with two single-pixel input and one double-pixel output channels

*3) Convolutional Layer:* Each convolutional layer converts an input volume ($H \times W \times D_{\text{in}}$) into an output volume ($H \times W \times D_{\text{out}}$), with $D_{\text{in}} \times D_{\text{out}}$ convolutions are required to produce $D_{\text{in}} \times D_{\text{out}}$ *intermediate* convolution outputs. Next, each output dimension (of $D_{\text{out}}$) requires $D_{\text{in}} - 1$ vector sums to add $D_{\text{in}}$ convolution outputs and produce one complete output, so each convolutional layer also requires $D_{\text{out}} \times (D_{\text{in}} - 1)$ vector sums.

The convolution function contains $N \times N$ convolutions, operates on $N$ inputs, and produces $N$ outputs, as illustrated in Figures 3(a) and 4. The convolution function reuses each of the $N$ input channels across $N$ convolutions, each with a different set of trained weights (convolution kernel and bias), to produce a total of $N^2$ intermediate outputs. For each of the $N$ output channels, the intermediate outputs are added to produce $N$ *partial* vector-sums. Next, the vector-sum function, as illustrated in Figure 3(b), is used to add all partial vector-sums to produce the complete convolutional layer output. Since each convolutional layer requires $D_{\text{in}} \times D_{\text{out}}$ convolutions and $D_{\text{out}} \times (D_{\text{in}} - 1)$ vector sums, the convolution and vector-sum functions must be invoked $\lceil \frac{D_{\text{in}}}{N} \rceil \times \lceil \frac{D_{\text{out}}}{N} \rceil$ and $\lceil 2\frac{D_{\text{out}}}{N} \rceil \times (\lceil \frac{D_{\text{in}}}{N} \rceil - 1)$ times, respectively, to process one convolutional layer.

The $N \times N$ convolution function design has a quadratic relationship between the number of channels and the number of convolutions. When $N$ doubles, the number of channels doubles and the number of convolutions quadruples, or equivalently, the number of invocations is quartered. Therefore, when $N$ doubles, the processing capability of the convolution function improves by a factor of four. However, this improvement only holds if the interconnect bandwidth can satisfy the doubled bandwidth requirement.

*4) Batch Normalization and ReLU Layers:* The batch-norm and ReLU operations each convert an input volume ($H \times W \times D$) into an output volume ($H \times W \times D$). Both operations are merged into one function, as illustrated in Figure 3(c), because both have the same access-pattern and one always precedes the other. This function requires four additional weights: running mean (E[$x$]), running variance (Var[$x$]), scale ($\gamma$), and shift ($\beta$), for the batch-norm operation. Since the batch-norm and ReLU function depends on a single dimension of the input volume, the $N$ instances of the function can run in parallel to use the full input and output interconnect bandwidths. This function requires $\lceil \frac{D}{N} \rceil$ invocations to perform all operations.

*5) Pooling Layers:* The pooling layer performs max-pooling with a filter size of $2 \times 2$ and converts an input volume ($H \times W \times D$) into two output volumes: the maxima ($\frac{H}{4} \times \frac{W}{4} \times D$) and the pooling indices ($\frac{H}{4} \times \frac{W}{4} \times D$). Max-pooling quarters the input spatial-resolution and produces a combined output volume that is half the input volume. The max-pooling function, as illustrated in Figure 3(d), contains $\frac{N}{2}$ instances of the max-pool operation. Each max-pool operation converts one input (as two channels packed into the input stream) into two outputs. Collectively, the max-pool function uses the full input bandwidth but can only use half the output bandwidth because the output stream size is half the input stream size.

*6) Unpooling Layers:* The unpooling layer performs max-unpooling with a filter size of $2 \times 2$ and converts two input volumes, feature maps ($H \times W \times D$) and indices ($H \times W \times D$), into one output volume ($4H \times 4W \times D$). Max-unpooling quadruples the input spatial-resolution and produces an output volume that is double the combined input volume. The max-unpooling function, as illustrated in Figure 3(e), contains $\frac{N}{2}$ instances of the max-unpool operation. Each max-unpool operation converts two inputs (feature maps and pooling indices) into one output (as two channels packed into the output stream). Collectively, the max-unpool function uses the full output bandwidth but can only use half the input bandwidth because the input stream size is half the output stream size.

*7) Quantization Optimizations:* ReCoN supports two data-type representations: single-precision floating-point and arbitrary-precision fixed-point. When configured to use floating-point, the ReCoN output is identical to the software output. However, floating-point arithmetic incurs a high area overhead and may require resource sharing to fit ReCoN into resource-constrained FPGAs at the cost of decreased performance. To improve performance, quantization can be used to constrain feature map and trained weights to arbitrary-precision fixed-point values. Generally, fixed-point arithmetic is substantially more area-efficient than floating-point arithmetic, and can allow for a greater scaling factor. Quantization also has the benefit of reducing the storage size of trained weights, which is useful for resource-constrained systems. However, due to loss of precision in arbitrary-precision fixed-point, the ReCoN output may deviate slightly compared to the software output as the precision error accumulates. However, the average error is negligible and may justify the trade-off for area, performance, and energy-efficiency benefits.

The digital signal processing (DSP) slices in the Zynq7 (DSP48E1) feature 25-bit$\times$18-bit multipliers. ReCoN uses the 25-bit operand for feature maps using the Q9.16 (9 signed-integer bits and 16 fractional bits) fixed-point format and the 18-bit operand for trained weights. The fixed-point format for trained weights varies by type (e.g., convolutional weights, convolutional bias, etc.) and is selected by fitting the minima and maxima of each type to an arbitrary-precision fixed-point format that maximizes precision. In the ZynqMP, the DSP slices (DSP48E2) feature 27-bit$\times$18-bit multipliers and can use the Q9.18 fixed-point format for a slight improvement in precision.

*8) Scatter-Gather Streaming Data-Flow Optimizations:* Although ReCoN will accelerate the processing of CNN layers, the communication cost associated with streaming data between the CPU and FPGA subsystems is also essential for

reducing the overall execution time. We developed a scatter-gather DMA (SGDMA) to facilitate the parallel streaming of multi-dimensional data through ReCoN. The SGDMA provides three major functions: scatter-gather streaming, stream-size parameterization, and decoupling logic for PR.

The SGDMA is full-duplex and converts an AXI interface into two AXI-Stream interfaces: one for DMA-to-accelerator (D2A) streaming and one for accelerator-to-DMA (A2D) streaming. For each direction, the SGDMA supports three run-time parameters: the number of channels to use, the data-width of the channels, and the length of the stream. The stream-size parameterization capability allows the SGDMA to interface to each accelerator function in ReCoN.

The scatter-gather streaming data-flow provided by the SGDMA has the advantage of creating an interleaving architecture. During a scatter-gather transfer, the SGDMA will rotate between AXI descriptors and complete one AXI burst transfer per channel before proceeding to the next one. Each AXI descriptor points to a DMA buffer, allowing the SGDMA to access multiple DMA buffers. Since the SGDMA effectively rotates between DMA buffers, the scatter-gather flow will seamlessly interleave buffer data in the D2A direction and deinterleave stream data in the A2D direction. The principal benefit is that multi-dimensional data can remain deinterleaved in DMA buffers, and the SGDMA will automatically perform the data-interleaving preprocess and data-deinterleaving postprocess in hardware. As a result, the SGDMA completely eliminates the overhead of software memory interleaving and deinterleaving.

Furthermore, since all accelerator functions operate on data streams, the AXI descriptors can be configured to reuse DMA buffers to perform the accelerator functions in-place. This access pattern has the advantage of reducing the memory overhead required for DMA buffers and significantly reduces the amount of software memory copies. The only software memory copies required are those that specify the header and function-specific section of the stream packet, which are negligible in terms of size compared to the data.

### C. ReCoN Control-Software

The ReCoN control-software provides the control-flow operations required for hybrid semantic segmentation. The control software allocates DMA buffers, loads input image data and trained weights, and invokes the SGDMA to asynchronously stream buffer data through ReCoN.

The control software is parameterizable to support arbitrary image volumes (spatial-resolution and dimension) and network shapes of the SegNet model to accommodate various space applications and imaging sensors (e.g., multispectral, hyperspectral, etc.). When initialized, the control software references two resources: the *network definition*, which specifies the network shape and the arrangement of layers, and the corresponding *trained weights*. Both resources are obtained after network development (testing, analysis, and training) and are uploaded to the spacecraft for deployment. For training, the dataset can be constructed using downlinked sensor data or approximated by using or modifying existing datasets.

Table I
EVALUATION PLATFORMS.

| | Xilinx ZC706 (Z7045) | Xilinx ZCU102 (ZU9EG) |
|---|---|---|
| **Processing System (PS)** | | |
| CPU | ARM Cortex-A9 (dual-core) | ARM Cortex-A53 (quad-core) |
| L1 cache | 32KB/32KB I/D per core | 32KB/32KB I/D per core |
| L2 cache | 512KB unified | 1MB unified |
| Frequency | 667MHz | 1.2GHz |
| **Programmable Logic (PL)** | | |
| FPGA | Kintex 7 (28 nm) | UltraScale architecture (16 nm) |
| LUTs | 218600 | 274080 |
| FFs | 437200 | 548160 |
| BRAM | 545 | 912 |
| DSPs | 900 | 2520 |
| Frequency | 100MHz/200MHz | 100MHz/300MHz |
| **PS-PL Interface** | | |
| Interface | AXI3 (64-bit/16-beat burst) | AXI4 (128-bit/256-beat burst) |
| **Acceleration Framework** | | |
| DMA | 8-channel SGDMA | 8-channel SGDMA |
| Accelerator | $\text{ReCoN}_{2/2\text{-TMR}/4/8}$ | $\text{ReCoN}_{2/2\text{-TMR}/4/8}$ |
| Quantization | fixed-point (Q9.16) | fixed-point (Q9.18) |

## IV. EVALUATION

To evaluate our hybrid approach for semantic segmentation, we experimentally recorded accuracy, resource utilization, performance, and energy-efficiency metrics by running our hybrid architecture on two hardware platforms at various configurations. In this section, we describe our experimental setup, target platforms, and application case-study, and analyze our results.

### A. Platforms

Our framework was realized on two hardware platforms, including the Xilinx ZC706 (Z7045) and Xilinx ZCU102 (ZU9EG). The system specifications for these platforms are detailed in Table I [6], [7]. For both platforms, Vivado 2018.2 was used to synthesize ReCoN and generate system bitstreams (using default synthesis and implementation settings), and Petalinux 2018.2 was used to deploy an embedded Linux operating system.

For our semantic segmentation application, we selected the Potsdam dataset from the ISPRS commission II/4 benchmark for 2D semantic labeling [23]. This dataset provides EO imagery in RGB (red-green-blue) and IRRG (infrared-red-green) formats, with ground-truth labels including six classes: roads, buildings, low vegetation, trees, automobiles, and clutter. We resized the dataset to 512×512 images, and then partitioned this dataset into 70% for training and 30% for testing. We trained three different network shapes: Net (86 layers, 7376806 weights), Net½ (86 layers,

Table II
INFERENCE ACCURACY.

| Inference Accuracy/Error | Net | Net½ | Net¼ |
|---|---|---|---|
| Inference Accuracy (RGB) | 90.17% | 89.63% | 88.30% |
| Inference Accuracy (IRRG) | 90.00% | 89.95% | 88.92% |
| Accelerator Error (floating-point) | 0.00% | 0.00% | 0.00% |
| Accelerator Error (Q9.16) | 0.73% | 0.40% | 0.30% |
| Accelerator Error (Q9.18) | 0.72% | 0.39% | 0.29% |

Table III
RESOURCE UTILIZATION.

| **Xilinx ZC706 (Z7045)** | | | | |
|---|---|---|---|---|
| Subsystem | Slices (218600) | FFs (437200) | BRAM (545) | DSPs (900) |
| Framework | 3.87% | 1.14% | 6.33% | 0.00% |
| $\text{ReCoN}_2$ | 1.62% | 2.03% | 1.84% | 4.44% |
| $\text{ReCoN}_{\text{2-TMR}}$ | 6.78% | 6.05% | 5.50% | 13.33% |
| $\text{ReCoN}_4$ | 3.81% | 5.94% | 3.49% | 16.89% |
| $\text{ReCoN}_8$ | 12.02% | 20.39% | 6.79% | 65.78% |

| **Xilinx ZCU102 (ZU9EG)** | | | | |
|---|---|---|---|---|
| Subsystem | Slices (274080) | FFs (548160) | BRAM (912) | DSPs (2520) |
| Framework | 4.23% | 1.00% | 7.57% | 0.04% |
| $\text{ReCoN}_2$ | 0.93% | 1.24% | 1.09% | 1.59% |
| $\text{ReCoN}_{\text{2-TMR}}$ | 4.59% | 3.70% | 3.29% | 4.76% |
| $\text{ReCoN}_4$ | 2.14% | 3.57% | 2.08% | 6.03% |
| $\text{ReCoN}_8$ | 6.45% | 11.64% | 4.05% | 23.49% |

1849814 weights), and Net¼ (86 layers, 465262 weights), where Net½ and Net¼ halves or quarters the dimension of each layer in Net, respectively. We use the single-threaded, software-only results as the baseline for our comparisons.

### B. Accuracy

In the context of semantic segmentation, accuracy refers to the prediction rate in which pixels of an image are assigned the correct label. Accuracy depends on several factors (e.g., network shape, training method, dataset, etc.). Using the test set, we calculated the accuracy for all three sample networks for each image format (RGB and IRRG), as shown in Table II. As noted previously, the floating-point version produces an output identical to the software version, but the fixed-point version has minor deviations due to accumulation of precision error. The average error for ZynqMP platforms using the Q9.18 format is slightly improved compared to Zynq7 platforms using the Q9.16 format.

### C. Resource Utilization

The resource utilization of the HARFT framework and ReCoN are separately shown in Table III. These numbers were obtained using the Vivado design tools post-implementation using default synthesis and implementation settings. When configured for efficient quantization, the scalability of ReCoN is bounded by the number of DSP slices available in the FPGA. $\text{ReCoN}_N$ requires $9N^2 + 2N$ DSP slices. The ZC706 and ZCU102 platforms provide enough DSP slices to support the 8-channel SGDMA and $\text{ReCoN}_8$ accelerator. Our CSPv1 space computer (Z7020) provides enough DSP slices for the 4-channel SGDMA and $\text{ReCoN}_4$ accelerator.

### D. Performance

For performance, the average execution times were measured for several configurations of the software and hybrid versions of the semantic segmentation application. The software-only version was compiled using GCC with O2 optimizations and NEON single-instruction, multiple-data intrinsics enabled. For multi-threading, OpenMP, an application programming interface for shared-memory multiprocessing, was used to parallelize all CNN functions. For the hybrid version, the performance was measured for varied scaling factors and the FPGA operating frequencies, as detailed in Table I. Table IV lists the execution times and the performance improvements compared to the baseline. In all situations, the hybrid version outperforms the software version by up to two orders of magnitude, depending on the network and system configuration.

### E. Energy Efficiency

Power and energy consumption are essential metrics for space systems. For a fair comparison, the FPGA was not programmed when running the software versions to assume a CPU-only system. Using a power meter, the overall system power was measured when idled (7.13W for the ZC706 and 21.60W for the ZCU102) and when actively processing the application. The dynamic-power and dynamic-energy consumption can be calculated by using the following equations:

$$\text{Dynamic Power} = \text{Active Power} - \text{Idle Power}$$

$$\text{Dynamic Energy} = \text{Execution Time} \times \text{Dynamic Power}$$

Table IV lists the dynamic-power and dynamic-energy consumptions, and the energy-efficiency improvements compared to the baseline. Although the hybrid versions often have a higher peak power-consumption, the reduced execution times result in significant improvements in overall dynamic-energy consumption, up to two orders of magnitude compared to the baseline. To accommodate space applications with stricter power requirements, the FPGA operating frequency and ReCoN configuration can be reduced at the cost of decreased performance.

## V. RELIABILITY EXPERIMENTS

This section describes the fault-injection and radiation-beam experiments performed to analyze the architectural response of ReCoN to both injected and radiation-induced errors. The objective of these experiments was to analyze the vulnerability of two designs: simplex, quad-channel $\text{ReCoN}_4$ and TMR, dual-channel $\text{ReCoN}_{\text{2-TMR}}$. Both accelerators have similar resource utilizations but introduce a trade-off

Table IV
PERFORMANCE AND ENERGY-EFFICIENCY.

**Xilinx ZC706 (Z7045)**

| Version (configuration) | Performance | | | | | | Energy-Efficiency | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Execution Time [s] | | | Improvement | | | Dynamic | Dynamic Energy [J] | | | Improvement | | |
| | Net | Net½ | Net¼ | Net | Net½ | Net¼ | Power [W] | Net | Net½ | Net¼ | Net | Net½ | Net¼ |
| **Software** | | | | | | | | | | | | | |
| 1 thread | 2345.1 | 543.2 | 115.7 | 1.0 | 1.0 | 1.0 | 1.46 | 3423.9 | 793.1 | 169.0 | 1.0 | 1.0 | 1.0 |
| 2 threads | 900.9 | 184.6 | 40.6 | 2.6 | 2.9 | 2.9 | 1.51 | 1360.3 | 278.8 | 61.3 | 2.5 | 2.8 | 2.8 |
| **Hybrid** (100MHz) | | | | | | | | | | | | | |
| $ReCoN_2$ | 66.0 | 16.8 | 4.6 | 35.5 | 32.3 | 25.3 | 1.60 | 105.3 | 26.9 | 7.3 | 32.5 | 29.5 | 23.2 |
| $ReCoN_{2\text{-}TMR}$ | 66.3 | 17.1 | 4.9 | 35.4 | 31.8 | 23.6 | 1.70 | 112.7 | 29.1 | 8.3 | 30.4 | 27.3 | 20.3 |
| $ReCoN_4$ | 27.6 | 7.3 | 2.1 | 85.0 | 73.7 | 54.5 | 1.62 | 44.8 | 12.0 | 3.4 | 76.4 | 66.2 | 49.0 |
| $ReCoN_8$ | 13.3 | 3.7 | 1.2 | 175.8 | 145.1 | 96.2 | 1.86 | 24.8 | 7.0 | 2.2 | 138.0 | 113.9 | 75.5 |
| **Hybrid** (200MHz) | | | | | | | | | | | | | |
| $ReCoN_2$ | 46.3 | 12.1 | 3.3 | 50.7 | 45.1 | 35.2 | 2.18 | 100.9 | 26.3 | 7.2 | 34.0 | 30.2 | 23.6 |
| $ReCoN_4$ | 18.1 | 5.0 | 1.5 | 129.3 | 109.5 | 76.8 | 2.28 | 41.2 | 11.3 | 3.4 | 83.0 | 70.3 | 49.2 |
| $ReCoN_8$ | 8.5 | 2.5 | 0.9 | 275.8 | 215.3 | 129.4 | 2.70 | 23.0 | 6.8 | 2.4 | 148.9 | 116.2 | 69.9 |

**Xilinx ZCU102 (ZU9EG)**

| Version (configuration) | Performance | | | | | | Energy-Efficiency | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Execution Time [s] | | | Improvement | | | Dynamic | Dynamic Energy [J] | | | Improvement | | |
| | Net | Net½ | Net¼ | Net | Net½ | Net¼ | Power [W] | Net | Net½ | Net¼ | Net | Net½ | Net¼ |
| **Software** | | | | | | | | | | | | | |
| 1 thread | 1973.4 | 370.5 | 70.7 | 1.0 | 1.0 | 1.0 | 2.65 | 5229.6 | 981.9 | 187.3 | 1.0 | 1.0 | 1.0 |
| 2 threads | 526.2 | 112.1 | 28.7 | 3.8 | 3.3 | 2.5 | 2.79 | 1468.2 | 312.8 | 80.2 | 3.6 | 3.1 | 2.3 |
| 4 threads | 274.3 | 57.8 | 14.9 | 7.2 | 6.4 | 4.8 | 3.20 | 879.0 | 185.2 | 47.6 | 6.0 | 5.3 | 3.9 |
| **Hybrid** (100MHz) | | | | | | | | | | | | | |
| $ReCoN_2$ | 55.2 | 13.5 | 3.8 | 35.7 | 27.5 | 18.8 | 2.70 | 148.8 | 36.3 | 10.1 | 35.1 | 27.1 | 18.5 |
| $ReCoN_{2\text{-}TMR}$ | 55.1 | 14.3 | 3.9 | 35.8 | 25.9 | 18.3 | 2.78 | 153.1 | 39.7 | 10.7 | 34.1 | 24.7 | 17.5 |
| $ReCoN_4$ | 16.6 | 4.7 | 1.5 | 118.9 | 78.7 | 46.6 | 2.88 | 47.8 | 13.6 | 4.4 | 109.4 | 72.4 | 42.9 |
| $ReCoN_8$ | 8.4 | 2.6 | 1.0 | 236.1 | 141.2 | 72.1 | 2.97 | 24.8 | 7.8 | 2.9 | 210.6 | 126.0 | 64.3 |
| **Hybrid** (300MHz) | | | | | | | | | | | | | |
| $ReCoN_2$ | 23.9 | 6.6 | 2.0 | 82.4 | 56.5 | 35.7 | 3.29 | 78.8 | 21.6 | 6.5 | 66.4 | 45.5 | 28.8 |
| $ReCoN_4$ | 8.7 | 2.7 | 1.0 | 227.8 | 137.2 | 68.6 | 3.49 | 30.2 | 9.4 | 3.6 | 173.0 | 104.2 | 52.1 |
| $ReCoN_8$ | 4.6 | 1.7 | 0.7 | 427.2 | 220.5 | 95.0 | 3.70 | 17.1 | 6.2 | 2.8 | 306.0 | 157.9 | 68.1 |

in performance and reliability. For both experiments, we used a reconfigurable-system design, with TMR-protected 4-channel SGDMA and CRAM scrubber residing in the static region, and either $ReCoN_4$ or $ReCoN_{2\text{-}TMR}$ residing in the PRR. The BL-TMR tool, a highly user-configurable tool for selective replication of FPGA designs, was used to apply low-level TMR [24]. The SGDMA was modified to compute XOR-based checksums on both the D2A and A2D streams. Since the execution of the SegNet model is deterministic, the output and intermediate checksums can be compared against golden checksums to determine the execution outcome and to identify exactly which layers were affected due to injected or radiation-induced errors, respectively.

### A. Fault Injection

Fault injection was performed to observe the architectural response of each design to errors injected into CRAM. For this experiment, the objective was to measure the architectural vulnerability factor (AVF) and mean-work-to-failure (MWTF) metrics for each design, and the tolerance of erroneous outputs. In this context, the AVF is the probability that an injected error will manifest into an erroneous output [26], and MWTF describes the average number of correct executions until an erroneous output is expected, capturing the trade-off between performance and reliability [27]. AVF and MWTF are calculated as follows:

$$\text{AVF} = \frac{\text{Number of Erroneous Executions}}{\text{Number of Fault Injections}}$$

$$\text{MWTF} = \frac{\text{Number of Correct Executions}}{\text{Number of Erroneous Executions}}$$

To avoid modifying the design, the Processor Configuration Access Port (PCAP) is used for injecting faults into the CRAM. The CRAM scrubber is inactive for this experiment because the fault-injection procedure is controlled (i.e., one error per iteration). In our fault-injection procedure, each iteration begins with the random selection of a layer and CRAM bit location (frame address, word, and bit). Next, the application is executed until it reaches the randomly selected layer, where the execution is halted, the fault is injected via the PCAP, and the execution is resumed. When the randomly selected layer is complete, the execution is halted, the fault is repaired, and the execution is resumed until completion. The error is restricted to the execution of the selected layer to focus on the vulnerability of that layer, as well as to represent the behavior of the CRAM scrubber

Table V
FAULT INJECTION AND WIDE-SPECTRUM NEUTRON-BEAM TEST RESULTS.

**Fault-Injection Results for TUL PYNQ-Z2 (Z7020)**

| Accelerator | CRAM [bits] | Injections | Errors | Hangs | MWTF | AVF | 95% Confidence Interval |
|---|---|---|---|---|---|---|---|
| $ReCoN_4$ | 2785403 | 9542673 | 2633904 | 58509 | 2.60 | 27.772% | [27.738%, 27.805%] |
| $ReCoN_{2\text{-TMR}}$ | 3513869 | 4124570 | 3362 | 2167 | 1225.18 | 0.082% | [ 0.079%, 0.084%] |
| Improvement | | | | | 471.08 | 340.71 | |

**Wide-Spectrum Neutron-beam Test Results for Digilent ZedBoard (Z7020)**

| Accelerator | Effective Fluence [n · cm$^{-2}$] | Total Executions | Errors | Hangs | MWTF | Cross-section [cm$^2$] | 95% Confidence Interval |
|---|---|---|---|---|---|---|---|
| $ReCoN_4$ | $9.91\times10^{11}$ | 76459 | 2769 | 165 | 26.55 | $2.79\times10^{-9}$ | $[2.69\times10^{-9}$ , $2.90\times10^{-9}$ ] |
| $ReCoN_{2\text{-TMR}}$ | $1.73\times10^{11}$ | 5882 | 53 | 14 | 109.72 | $3.06\times10^{-10}$ | $[2.24\times10^{-10}$, $3.89\times10^{-10}]$ |
| Improvement | | | | | 4.13 | 9.12 | |

**Wide-Spectrum Neutron-beam Test Results for Avnet UltraZed-EG (ZU3EG)**

| Accelerator | Effective Fluence [n · cm$^{-2}$] | Total Executions | Errors | Hangs | MWTF | Cross-section [cm$^2$] | 95% Confidence Interval |
|---|---|---|---|---|---|---|---|
| $ReCoN_4$ | $3.49\times10^{11}$ | 75527 | 25 | 1 | 3020.04 | $7.17\times10^{-11}$ | $[4.65\times10^{-11}$, $1.06\times10^{-10}]$ |
| $ReCoN_{2\text{-TMR}}$ | $1.27\times10^{11}$ | 14782 | 0[1] | 0 | 14781.00 | $7.84\times10^{-12}$ | $[7.84\times10^{-13}$, $4.39\times10^{-11}]$ |
| Improvement | | | | | 4.89 | 9.14 | |

[1] Assuming one error when no errors were detected [25].

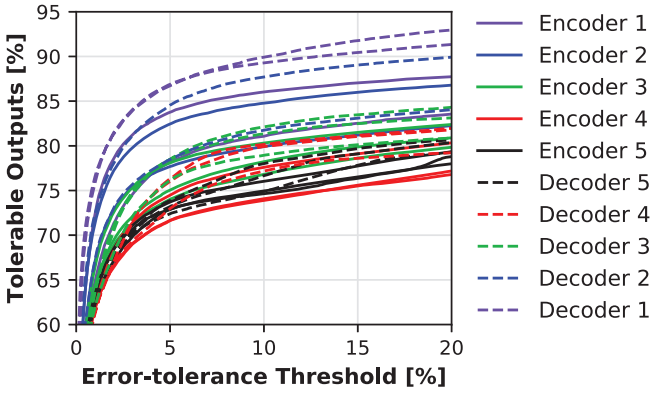


Figure 5. Percentage of tolerable outputs from $ReCoN_4$ accelerator for varied error-tolerance thresholds.

present in the actual flight system which would correct the error. To accelerate our fault-injection procedure, only essential CRAM bits are targeted. Essential CRAM bits are the set of bits that are actively used in the FPGA design, and errors in these bits will affect the design. The Xilinx design tools can generate the locations of these essential CRAM bits [28]. Eight PYNQ-Z2 (Z7020) devices-under-test (DUTs) were used to parallelize the fault-injection procedure.

The fault-injection results are detailed in Table V. The AVF and MWTF improvements indicate that the fault tolerance provided by $ReCoN_{2\text{-TMR}}$ allows this design to reliably execute more inferences than $ReCoN_4$, despite reduced performance and energy-efficiency. The AVF of $ReCoN_4$ can be improved if some error can be tolerated. For example, a few erroneous pixels may be tolerable, but severe distortions may not be. An error-tolerance threshold, which denotes tolerable percentage of erroneous pixels compared to the golden output, can be used to determine the percentage of tolerable outputs throughout the entire fault-injection procedure. Using the XOR-based checksums generated by the SGDMA, the error tolerance can be analyzed at the layer granularity. Figure 5 illustrates the percentage of tolerable outputs by convolutional layer, and the encoder or decoder block it resides in, for varied error-tolerance thresholds. As evident in Figure 5, errors in the inner convolutional-layers are more susceptible to producing intolerable outputs, compared to the outer layers. This architectural response may be due to the inner layers of the SegNet model operating on high-dimensional and highly discretized feature maps representing complex abstractions, which may facilitate the propagation of errors. The reconfigurable-system architecture can alternate between both versions of ReCoN (e.g., simplex for outer layers, and TMR for inner layers) to maximize performance subject to reliability constraints.

### B. *Radiation-beam Testing*

ReCoN was irradiated by a wide-spectrum neutron beam at the Los Alamos Neutron Science Center (LANSCE) Weapons Neutron Research (WNR) facility, using the 4FP30R/ICE-II instrument [29]. In this experiment, the neutron-induced, application-error cross-section and MWTF metrics were calculated for both designs. In this context, the cross-section is the sensitive area of the DUT in which neutron-induced errors will manifest into an erroneous output (i.e., silent data-corruption). Four Digilent ZedBoard (Z7020) and two Avnet UltraZed-EG (ZU3EG) DUTs were placed in the beam to parallelize the fluence each design was exposed to. The experimental setup is shown in Figure 6. Because the error rate is uncontrolled, the CRAM scrubber

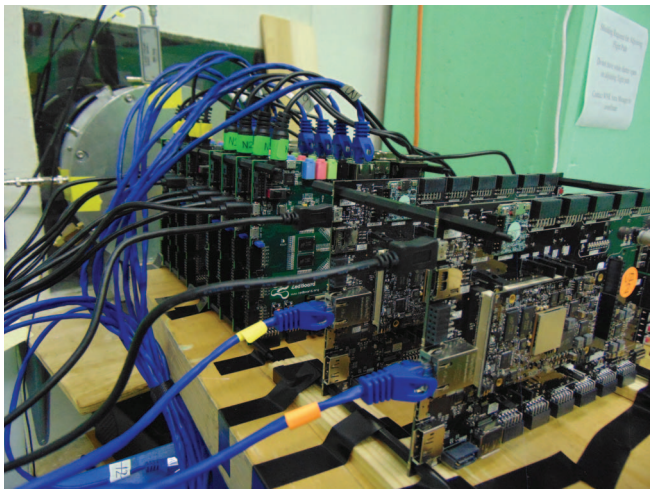

Figure 6. Experimental setup at LANSCE 4FP30R/ICE-II.

was used to prevent the accumulation of errors in CRAM. However, the radiation beam can expose the DUTs to error modes that cannot be directly compared with our fault-injection procedure (e.g., multi-bit upsets, CPU or memory errors, overwhelmed scrub-rate, etc.).

In our radiation-beam test procedure, the DUTs continuously ran semantic segmentation on either version of ReCoN. DUT-management software was used to automate the power-cycling of DUTs when it was detected that the DUT had hanged (failed to signal a heartbeat before timeout), reported consecutive errors (counted as one error), or detected a failure of the CRAM scrubber. Golden checksums were used to test the execution outcomes (correct, error, or hang), which were recorded with timestamps. The 4FP30R/ICE-II instrument contains a dosimeter that records the integrated neutron flux (above 10 MeV) with timestamps. The neutron fluence (above 10 MeV) can be calculated by integrating the neutron flux over the time interval that the DUT was active. The cross-section and corresponding 95% confidence interval are calculated as specified in [25]:

$$\text{Cross-section} = \frac{\text{Number of Erroneous Executions}}{\text{Effective Fluence}}$$

For the ZedBoard 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 CPU-related errors and failures. For the UltraZed-EG DUTs, the DDR memory was also configured with ECC enabled but the caches were kept enabled as the ZynqMP CPU demonstrates high resilience to SEUs [30]. The designs were alternated between DUTs and the recorded fluence was adjusted to account for the distance between the DUT and beam source. The experimental results are detailed in Table V. For both sets of DUTs, the cross-section and MWTF improvement reaffirms the advantage of $\text{ReCoN}_{\text{2-TMR}}$ to reliably execute more inferences than $\text{ReCoN}_4$, despite performance and energy-efficiency trade-offs. The dissimilarity in the cross-section magnitudes between both sets of DUTs can be attributed to architecture and process-technology differences between both sets of DUTs.

## VI. Conclusions

Despite the high-applicability of deep learning for spaceflight, deep-learning algorithms such as CNNs are computationally expensive and prohibited on traditional rad-hard processors. Commercial hybrid SoCs present numerous architectural advantages that address on-board processing challenges. However, effective use of both the CPU and FPGA subsystems is required to reliably maximize the benefits provided by the hybrid architecture.

In this article, we introduced our hybrid approach for semantic segmentation on hybrid SoCs. When evaluated on the Xilinx Zynq SoC and Xilinx Zynq UltraScale+ MPSoC platforms, our hybrid approach demonstrates an improvement in performance and energy-efficiency up to two orders of magnitude compared to a software-only baseline on the hybrid SoC. Due to significant performance speedup and reduced energy consumption, our hybrid approach can be an enabling technology for applying semantic segmentation and other CNN algorithms to future space missions. For future work, we will investigate new optimizations in ReCoN to further enhance performance and energy-efficiency.

Additionally, fault-injection and radiation-beam testing was performed to characterize the architectural response of two versions of ReCoN (one simplex, high-performance and one TMR, low-performance) to injected and neutron-induced errors. In our CRAM fault-injection experiment, we measured the AVF and MWTF of both designs, and identified a pattern in error tolerance across layers of the SegNet model. In our radiation-beam test, we measured the cross-section and MWTF for both designs under wide-spectrum neutron irradiation. These experiments are the basis for future work in adaptive CNNs, which alternate between high-performance and high-reliability versions of ReCoN across layers with varied susceptibilities, to maximize inference performance subject to reliability constraints.

The HARFT SoC reliability framework is currently integrated into the STP-H6-SSIVP experiment on board the ISS [9]. Using EO imagery captured by SSIVP, a dataset can be constructed or approximated, and a CNN based on the SegNet model can be developed and trained using the new dataset. Finally, the network definition, trained weights, and ReCoN (PR bitstream) can be uploaded to flight-qualify hybrid semantic segmentation for on-board processing.

## Acknowledgments

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. The authors would like to thank Christopher Wilson, James MacKinnon, and Daniel Sabogal. The authors would also

like to thank the U.S. Department of Energy and LANSCE for the invaluable beam time.

## REFERENCES


[1] National Academies of Sciences, Engineering, and Medicine, *Thriving on Our Changing Planet: A Decadal Strategy for Earth Observation from Space*. Washington, DC: The National Academies Press, 2018. [Online]. Available: https://www.nap.edu/catalog/24938/thriving-on-our-changing-planet-a-decadal-strategy-for-earth

[2] National Academies of Sciences, Engineering, and Medicine, *Achieving Science with CubeSats: Thinking Inside the Box*. Washington, DC: The National Academies Press, 2016. [Online]. Available: https://www.nap.edu/catalog/23503/achieving-science-with-cubesats-thinking-inside-the-box

[3] DARPA BAA, "Blackjack (BAA HR001118S0032)," 2018.

[4] A. D. George and C. M. Wilson, "Onboard processing with hybrid and reconfigurable computing on small satellites," *Proceedings of the IEEE*, vol. 106, no. 3, pp. 458–470, March 2018.

[5] S. Sabogal, A. D. George, and G. Crum, "Hybrid semantic image segmentation using deep learning for on-board space processing," in *NASA Goddard Workshop on Artificial Intelligence*, 2018. [Online]. Available: https://asd.gsfc.nasa.gov/conferences/ai/program/031-NASA-GSFC-AI-Workshop-ssabogal.pdf

[6] Xilinx, *Zynq-7000 All Programmable SoC Technical Reference Manual*, Xilinx, Dec 2017, xilinx User Guide (UG585).

[7] Xilinx, *Zynq UltraScale+ Device Technical Reference Manual*, Xilinx, Dec 2017, xilinx User Guide (UG1085).

[8] C. Wilson and A. George, "CSP hybrid space computing," *Journal of Aerospace Information Systems*, vol. 15, no. 4, pp. 215–227, Feb 2018. [Online]. Available: https://doi.org/10.2514/1.I010572

[9] S. Sabogal, P. Gauvin, B. Shea, D. Sabogal, A. Gillette, C. Wilson, A. Barchowsky, A. D. George, G. Crum, and T. Flatley, "SSIVP: Spacecraft supercomputing experiment for STP-H6," in *Proceedings of the 31st Annual AIAA/USU Conference on Small Satellites*. Logan, UT: AIAA, 2017, pp. 1–12.

[10] T. M. Lovelly and A. D. George, "Comparative analysis of present and future space-grade processors with device metrics," *Journal of Aerospace Information Systems*, vol. 14, no. 3, pp. 184–197, Mar 2017. [Online]. Available: https://doi.org/10.2514/1.I010472

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

[12] R. H. Maurer, M. E. Fraeman, M. N. Martin, and D. R. Roth, "Harsh environments: Space radiation environment, effects, and mitigation," *Johns Hopkins APL Technical Digest*, vol. 28, no. 1, pp. 17–29, 2008.

[13] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, Oct 1986. [Online]. Available: http://dx.doi.org/10.1038/323533a0

[14] D. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," *CoRR*, vol. abs/1511.07289, 2015. [Online]. Available: http://arxiv.org/abs/1511.07289

[15] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[16] V. Badrinarayanan, A. Handa, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling," *CoRR*, vol. abs/1505.07293, 2015. [Online]. Available: http://arxiv.org/abs/1505.07293

[17] N. Audebert, B. L. Saux, and S. Lefèvre, "Beyond RGB: Very high resolution urban remote sensing with multimodal deep networks," *ISPRS Journal of Photogrammetry and Remote Sensing*, 2017.

[18] K. Abdelouahab, M. Pelcat, J. Serot, and F. Berry, "Accelerating CNN inference on FPGAs: A survey," 2018.

[19] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing FPGA-based accelerator design for deep convolutional neural networks," in *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '15. New York, NY, USA: ACM, 2015, pp. 161–170. [Online]. Available: http://doi.acm.org/10.1145/2684746.2689060

[20] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan 2017.

[21] C. Farabet, B. Martini, B. Corda, P. Akselrod, E. Culurciello, and Y. Lecun, "NeuFlow: A runtime reconfigurable dataflow processor for vision," in *2011 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPRW 2011*, 2011.

[22] C. Wilson, S. Sabogal, A. George, and A. Gordon-Ross, "Hybrid, adaptive, and reconfigurable fault tolerance," in *2017 IEEE Aerospace Conference*, March 2017, pp. 1–11.

[23] I. Potsdam, "2d semantic labeling dataset," 2018. [Online]. Available: http://www2.isprs.org/commissions/comm3/wg4/2d-sem-label-potsdam.html

[24] J. M. Johnson and M. J. Wirthlin, "Voter insertion algorithms for FPGA designs using triple modular redundancy," in *Proceedings of the 18th Annual ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, ser. FPGA '10. New York, NY, USA: ACM, 2010, pp. 249–258. [Online]. Available: http://doi.acm.org/10.1145/1723112.1723154

[25] H. Quinn, "Challenges in testing complex systems," *IEEE Transactions on Nuclear Science*, vol. 61, no. 2, pp. 766–786, April 2014.

[26] S. S. Mukherjee, C. Weaver, J. Emer, S. K. Reinhardt, and T. Austin, "A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor," in *Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36.*, Dec 2003, pp. 29–40.

[27] G. A. Reis, J. Chang, N. Vachharajani, S. S. Mukherjee, R. Rangan, and D. I. August, "Design and evaluation of hybrid fault-detection systems," in *32nd International Symposium on Computer Architecture (ISCA'05)*, June 2005, pp. 148–159.

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

[29] S. F. Nowicki, S. A. Wender, and M. Mocko, "The Los Alamos Neutron Science Center spallation neutron sources," *Physics Procedia*, vol. 90, pp. 374 – 380, 2017.

[30] J. D. Anderson, J. C. Leavitt, and M. J. Wirthlin, "Neutron radiation beam results for the Xilinx UltraScale+ MPSoC," in *2018 IEEE Nuclear Space Radiation Effects Conference (NSREC 2018)*, July 2018, pp. 1–7.