A Methodology for Evaluating and Analyzing FPGA-Accelerated, Deep-Learning Applications for Onboard Space Processing

Sebastian Sabogal, Alan George
NSF Center for Space, High-Performance, and Resilient Computing (SHREC)
University of Pittsburgh
Pittsburgh, PA, USA
{sebastian.sabogal,alan.george}@pitt.edu

2021 IEEE Space Computing Conference (SCC) | 978-1-6654-2400-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/SCC49971.2021.00022

***Abstract*—Due to continued innovations in onboard data analysis and spacecraft autonomy, enabled by deep learning (DL), modern spacecraft require dependable, high-performance computers to process onboard an immense volume of raw sensor data into actionable information to formulate critical decisions autonomously. To enable compute-intensive DL algorithms, commercial-off-the-shelf processors, including FPGAs and system-on-chips, are often employed for their superior performance, energy-efficiency, and affordability compared to traditional radiation-hardened alternatives; however, these processors are highly susceptible to radiation-induced single-event effects (SEEs) that can degrade the dependability of DL applications.**

Researchers have created a diverse collection of DL models that perform a variety of tasks useful for Earth-observation missions. However, due to characteristic differences between models and accelerators, their tradeoffs can vary in terms of accuracy, area, performance, energy-efficiency, and dependability, which are factors crucial for resource-constrained and mission-critical systems. To select the optimal DL solution that maximizes inference performance, conserves onboard resources, and satisfies mission dependability requirements, a methodology is required to evaluate and compare the tradeoffs between competing options.

In this paper, we propose a methodology for evaluating and analyzing the tradeoffs of FPGA-accelerated DL models, including a hierarchical fault-injection approach to accelerate the characterization of SEE susceptibility of DL solutions in terms of well-established dependability metrics. Furthermore, we identify performance and dependability trends, analyze the impact of SEEs on the inference accuracy, and predict design fault rates for near-Earth orbital environments. To demonstrate the versatility of our methodology, we evaluate and analyze four semantic-segmentation models accelerated on four Xilinx Deep-Learning Processing Unit accelerators.

***Index Terms*—FPGA, deep learning, semantic segmentation, single-event effects, fault injection, space computing**

I. Introduction

Machine learning (ML), particularly deep learning (DL), continues to proliferate in space applications to enhance mission capabilities in onboard data analysis and spacecraft autonomy [1], [2]. DL can enable a variety of complex mission tasks for both science and defense missions such as remote sensing [3], [4], constellation management [5], and terrain-relative navigation [6]. However, despite these advantages, DL algorithms are computationally intensive and require dependable, high-performance space computers capable of converting onboard an immense volume of raw sensor data into actionable information that can be used to formulate critical decisions autonomously. Additionally, constraints in size, weight, power, and cost (SWaP-C) and requirements in mission dependability for the harsh radiation environment further exacerbate the space-computing challenge.

To address these challenges, small satellites (SmallSats), including CubeSats, have proliferated in space applications as low-SWaP-C platforms enabled by miniaturized technologies [2], [7]. To improve onboard processing capabilities, SmallSat missions often employ computers composed partially or solely of commercial-off-the-shelf devices due to their superior performance, energy-efficiency, and affordability compared to traditional radiation-hardened (rad-hard) alternatives. Often, these computers will also include FPGAs and hybrid system-on-chips (SoCs), which synergize multiple, distinct computing architectures within one device to attain the architectural advantages of each. FPGA-based hybrid SoCs, which combine fixed-logic CPUs with reconfigurable-logic FPGAs, provide several architectural advantages suitable for onboard DL acceleration. However, commercial FPGAs and SoCs are highly susceptible to radiation-induced single-event effects (SEEs) that can degrade the dependability of the DL application. Consequently, dependability must be considered for systems that use these commercial devices to deploy DL in mission-critical applications.

Furthermore, researchers have created a broad variety of DL models for applications that will also vary in dependability. Some examples include DL models that perform classification, detection, localization, and segmentation tasks on imagery and can be applied to enhance applications in Earth observation (EO) and remote sensing [8]. However, due to characteristic differences between DL models (e.g., network structure, operations, and trained parameters) and accelerators (e.g., architecture, optimizations, and data-flow), DL solutions can vary broadly in terms of accuracy, resource utilization,

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.

978-1-6654-2400-4/21/$31.00 ©2021 IEEE
DOI 10.1109/SCC49971.2021.00022

Authorized licensed use limited to: University of Pittsburgh. Downloaded on April 08,2022 at 19:08:38 UTC from IEEE Xplore. Restrictions apply.

performance, energy-efficiency, and dependability. All these tradeoffs are crucial for resource-constrained and mission-critical systems. To select an optimal DL solution for a specific task that maximizes inference performance, conserves onboard resources, and satisfies dependability requirements, a methodology is required to evaluate and compare the tradeoffs between competing options.

In this paper, we propose a methodology for evaluating FPGA-accelerated DL models and analyzing their tradeoffs. With an emphasis on the dependability evaluation, we also propose a hierarchical fault-injection approach to accelerate the characterization of fault susceptibility in DL solutions. In our hierarchical approach, fault injection is performed at multiple levels, in order of decreasing application granularity (coarse to fine), and the targeted area at each level is continually reduced by omitting inconsequential bits of the preceding level, thus substantially reducing the number of fault injections required. Furthermore, we propose analytical methods that use our hierarchical fault-injection approach to quantify and examine FPGA-accelerated DL models in terms of well-established dependability metrics and to observe the impact of faults on inference accuracy, profile and map vulnerability to node-level operations, and predict design fault rates for near-Earth orbital environments. To demonstrate the versatility of our methodology, we evaluate four semantic-segmentation DL models accelerated on four Xilinx Deep-Learning Processing Unit (DPU) accelerator configurations implemented on two generations of Xilinx SoCs: Zynq-7000 SoC (Zynq-7000) and Zynq UltraScale+ MPSoC (Zynq-MPSoC).

II. Background

This section provides a cursory overview of hybrid space computing for SmallSat missions. In addition, the section discusses dependable computing for FPGAs, including radiation effects on FPGAs, SEE-mitigation techniques and evaluation methods needed in a space environment. Next, this section presents concepts in semantic segmentation and FPGA acceleration of DL applications, including deep neural network (DNN) architectures and optimizations. Finally, this section provides a discussion of related work in the evaluation and analysis of the SEE susceptibility of FPGA-accelerated DL models.

A. Hybrid and Reconfigurable Space Systems

To enable onboard DL applications, SmallSat missions often employ commercial hybrid SoCs, including the Xilinx Zynq-7000 SoC (Zynq-7000) and Zynq UltraScale+ MPSoC (Zynq-MPSoC). Both SoC series combine fixed-logic CPU cores with reconfigurable-logic FPGA fabric within a single chip, with both subsystems interconnected by high-speed Advanced eXtensible Interface (AXI) interfaces. The Zynq-7000 Z7020 SoC features a dual-core ARM Cortex-A9 CPU with Artix 7-Series FPGA fabric interconnected by 64-bit AXI3 interfaces. The Zynq-MPSoC ZU3EG SoC features a quad-core ARM Cortex-A53 CPU with UltraScale+ Architecture FPGA fabric interconnected by 128-bit AXI4 interfaces. Furthermore, both SoC series include configuration access ports (CAPs) that enable FPGA reconfiguration and configuration memory (CRAM) access. These ports include the processor CAP (PCAP; peripheral in CPU) and internal CAP (ICAP; special block in FPGA).

The CHREC Space Processor (CSP) and SHREC Space Processor (SSP) are two examples of multifaceted hybrid space computers developed by researchers at the National Science Foundation (NSF) Center for Space, High-Performance, and Resilient Computing (SHREC) in collaboration with government and industry partners [9]. Both single-board computers feature a Zynq-7000 SoC (Z7020 or Z7030/Z7030/Z7045) and combine a novel mix of commercial and rad-hard technology, supplemented by SEE mitigation enhancements, to achieve the nexus in dependable, high-performance computing. CSP has flight heritage as part of two U.S. Department of Defense Space Test Program (STP) Houston missions to the International Space Station (ISS), including STP-H5 CHREC Space Processor (STP-H5-CSP) and STP-H6 Spacecraft Supercomputing for Image and Video Processing (STP-H6-SSIVP). Both CSP and SSP are planned for flight on the STP-H7 Configurable and Autonomous Sensor Processing Research (STP-H7-CASPR). The Innoflight Compact Flight Computer (CFC-300), GomSpace Nanomind Z7000, and Xiphos Q7 are alternative examples of single-board computers featuring a Zynq-7000. The NASA SpaceCube v3.0 VPX (SCv3VPX) [10], Innoflight CFC-400, and Xiphos Q8 are examples that feature a Zynq-MPSoC.

B. Radiation Effects and FPGA Dependability

Space radiation poses several challenges for electronic devices. Sources of radiation include galactic cosmic rays, solar particle events, and charged particles trapped within the geomagnetic field. Radiation effects on electronic devices are often categorized as cumulative or transient effects. Total ionizing dose and displacement damage dose are both long-term, cumulative effects due to continuous exposure to radiation. Single-event effects (SEEs) are transient, short-term effects that occur when a single radiation particle strikes the device causing an effect. SEEs can be destructive (e.g., latch-up) or nondestructive (upsets, transients, or functional interrupts). Radiation effects testing is extensively covered in [11].

1) FPGA Dependability: SRAM-based FPGAs are high-density, reconfigurable architectures consisting of many resources (e.g., logic blocks, DSPs, BRAM, I/O blocks, etc.) interconnected by a complex, configurable routing network. At runtime, a bitstream is stored in SRAM-based CRAM which configures the resources and network routing to implement a design on the FPGA. This architectural paradigm enables users to create massively parallel datapaths to accelerate compute-intensive algorithms on FPGAs, and the capability for runtime reconfiguration. However, despite these advantages, SRAM-based FPGAs are highly susceptible to radiation-induced SEEs that can affect the dependability of the design. Faults in static CRAM bits, which define the internal architecture, can cause functional changes in the design operation. Faults in dynamic CRAM bits and other on-chip

Authorized licensed use limited to: University of Pittsburgh. Downloaded on April 08,2022 at 19:08:38 UTC from IEEE Xplore. Restrictions apply.

memories can also cause a wide variety of adverse effects. In practice, fault-masking techniques (e.g., triple-modular redundancy), CRAM scrubbing, and error correction code are employed to improve design dependability with tradeoffs in performance, area, and energy-efficiency. A comprehensive overview of the radiation effects on FPGAs, including SEE mitigation techniques, is covered in the literature [12].

2) Dependability Analysis: The dependability of an FPGA design can be measured experimentally using fault-injection or radiation-beam testing. CRAM fault injection involves injecting a bit-flip into CRAM to observe the architectural response to the fault during design operation. The architectural vulnerability factor (AVF) and mean-work-to-failure (MWTF) are two useful metrics for quantifying the dependability of a design. AVF refers to the probability that an injected fault will manifest into an observable event, and MWTF refers to the amount of useful work completed (e.g., number of inferences) until an observable event is expected. Observable events are user-defined and can vary by application. Examples include silent data corruption (SDC) or hangs. AVF and MWTF are calculated using Eq. (1) and Eq. (2), respectively. In practice, the AVF results are reported with the corresponding 95% confidence interval (CI) to provide context for uncertainty in the measurements of the experiment [13].

$$\text{AVF} = \frac{\text{Number of Observable Events}}{\text{Number of Fault Injections}} \tag{1}$$

$$\text{MWTF} = \frac{\text{Amount of Useful Work Completed}}{\text{Number of Observable Events}} \tag{2}$$

For a specific near-Earth orbital environment and observable event, the fault rate of an FPGA design can be approximated using Eq. (3). For each FPGA resource type $r \in R$, the resource SEE rate $\lambda_{r,\text{SEE}}$ is calculated and scaled by the resource utilization RU_r times the resource AVF (AVF_r). To calculate the resource SEE rate, the Cosmic Ray Effects on Micro-Electronics (CRÈME96) tool can be used. CRÈME96, 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 [14]. The resource utilization of the design is generated by the design tools. Finally, the resource AVF can be quantified by fault injection or radiation-beam testing. In cases where determining the AVF of a specific resource type is infeasible, an estimate is made (e.g., assume worst case or use the AVF of another resource). The fault rate of the final design is the summation of the scaled fault rates for all resource types.

$$\lambda_{\text{design}} = \sum_{r \in R} \lambda_{r,\text{SEE}} \cdot \text{RU}_r \cdot \text{AVF}_r \tag{3}$$

C. Semantic Segmentation

Semantic segmentation is a DL task that can label images at the pixel level, where pixels with the same label share the same semantic characteristics [15]. Semantic segmentation can be applied in numerous EO applications (e.g., land use, land cover, and cloud masking). Four examples of DL models that perform semantic segmentation include U-Net [16], Efficient Neural Network (ENet) [17], Feature Pyramid Network (FPN) [18], and Efficient Spatial Pyramid Neural Network (ESPNet) [19], each with unique characteristics. U-Net is a symmetric autoencoder and contains contracting and expanding data pathways. The contracting pathway is a sequence of encoder blocks that perform feature extraction and pooling-based downsampling, and the expanding pathway is a sequence of decoder blocks that perform deconvolutional upsampling using feature maps (FMs) from preceding and lateral blocks. ENet is an asymmetric autoencoder that uses dilated convolution to achieve a larger receptive field of each convolutional filter, stores pooling indices from early pooling layers for unpooling-based upsampling, and factorizes filters to drastically reduce the number of parameters. FPN is a pyramidal structure with bottom-up and top-down pathways. The bottom-up pathway generates FMs with increasing semantic value at decreasing resolution, and the top-down pathway reconstructs higher resolution layers using FMs from the preceding and lateral levels. Finally, ESPNet is an asymmetric autoencoder that uses efficient spatial pyramid convolutional modules, which perform point-wise convolution followed by a spatial pyramid of dilated convolutions that significantly decreases the number of parameters required while maintaining a large receptive field.

A variety of metrics exist to quantify the inference accuracy of a DL model depending upon the task. For semantic segmentation, the mean intersection-over-union (mIoU) and mean F1 (Dice) score are two standard metrics used to quantify the accuracy of a segmented output compared to a ground-truth label mask [15].

D. Deep Learning on FPGAs

Researchers have explored a broad variety of architectures, optimizations, and design-exploration methods to maximize DL acceleration capabilities for FPGAs that enable a diverse range of DL applications for edge systems [20]. One example is the Xilinx Deep-Learning Processing Unit (DPU) [21], the hardware component of the Vitis AI stack [22], illustrated in Fig. 1. The DPU is a general-purpose DNN accelerator that uses a coprocessing architecture. The DPU includes (1) an instruction unit, which performs instruction fetching and scheduling of node-level operations on FM data, (2) a global memory pool, which manages on-chip and off-chip memory buffers, and (3) a hybrid computing array, which is composed of a scalable number of processing engines (PEs) that perform multiply-accumulate (MAC) and other miscellaneous operations for node processing.

The Vitis AI stack features a variety of model-compression, algorithmic, and architectural optimizations to efficiently map DL models onto the FPGA for acceleration. Model-compression optimizations include parameter pruning and data quantization that improve efficiency at the expense of minimally decreased accuracy. Pruning removes parameters with minimal representation in the model to reduce the model size, and quantization replaces resource-intensive floating-point data with low-precision integer data to

Authorized licensed use limited to: University of Pittsburgh. Downloaded on April 08,2022 at 19:08:38 UTC from IEEE Xplore. Restrictions apply.

reduce area, bandwidth, energy, and storage requirements. Next, algorithmic optimizations include loop optimizations (unrolling, tiling, and interchange) to maximize the data-flow efficiency and cache performance of on-chip memory. Finally, architectural optimizations include DSP time multiplexing and node fusion. The DPU operates DSP resources at twice the frequency of surrounding logic to accomplish the same amount of computations with only half of the DSP resources. Node fusion involves fusing multiple node operations into one supernode to improve latency and efficiency.

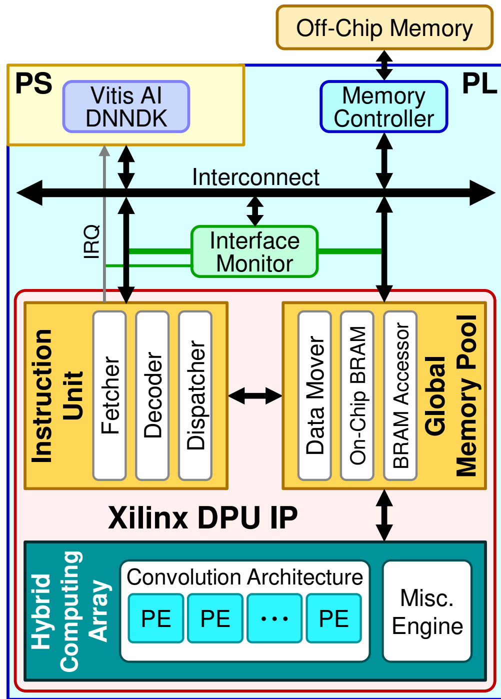


Fig. 1. Xilinx deep-learning processing unit (DPU) architecture.

E. Related Work

The dependability of FPGA-accelerated ML models, including methods for evaluation and mitigation, has been explored in the literature [23]. A variety of approaches for evaluating ML dependability using fault injection and radiation-beam testing have also been proposed. In [24], single-bit and multi-bit fault injection were performed in both static and dynamic CRAM to observe the architectural response of a binary neural network to both single and multi-bit upsets. In [25], [26], the dependability tradeoffs between mixed-precision float-point and binary quantization data types of a CNN were analyzed. In [27], fault injection and neutron irradiation were performed on multi-layer perceptron with layers assigned to separate FPGA partitions to analyze the ML model at the model and layer levels.

The dependability analysis of FPGA-accelerated ML models has also enabled efficient methods for SEE mitigation. In [28], [29], fault injection was performed to identify the most vulnerable layers and channels, respectively, for selective replication to improve overall dependability with minimal overhead due to replication. In [30], a CNN was disaggregated into a static, replicated control-flow subset and a runtime-reconfigurable data-flow subset that can be exchanged with unmitigated, high-performance and mitigated, low-performance versions of the accelerator. In this paper, we propose an efficient fault-injection approach to accelerate the evaluation of DL solutions for FPGAs to enable a rapid tradespace analysis between DL models and accelerators for optimal selection, and to quickly identify vulnerable parts of the DL solution for selective or adaptive SEE mitigation.

III. Approach

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

A. Hierarchical Fault-Injection Approach

Our hierarchical approach involves fault injection at multiple levels, in order of decreasing granularity (e.g., application $\rightarrow$ phases $\rightarrow$ subphases), and focuses on continually narrowing the size of targeted CRAM between levels by omitting bits with noncritical representation in the preceding level. At each level transition, the subset of tested CRAM that manifests into observable events becomes the target CRAM for the subsequent level, and the remaining bits (noncritical and untested) are omitted. The continual omission of inconsequential bits can substantially reduce the number of fault injections required to analyze an FPGA-accelerated application at low levels of granularity. For our evaluation of FPGA-accelerated DL models, two levels are sufficient to evaluate a DL model at the model and node levels. The approach is illustrated in Fig. 2.

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

Authorized licensed use limited to: University of Pittsburgh. Downloaded on April 08,2022 at 19:08:38 UTC from IEEE Xplore. Restrictions apply.

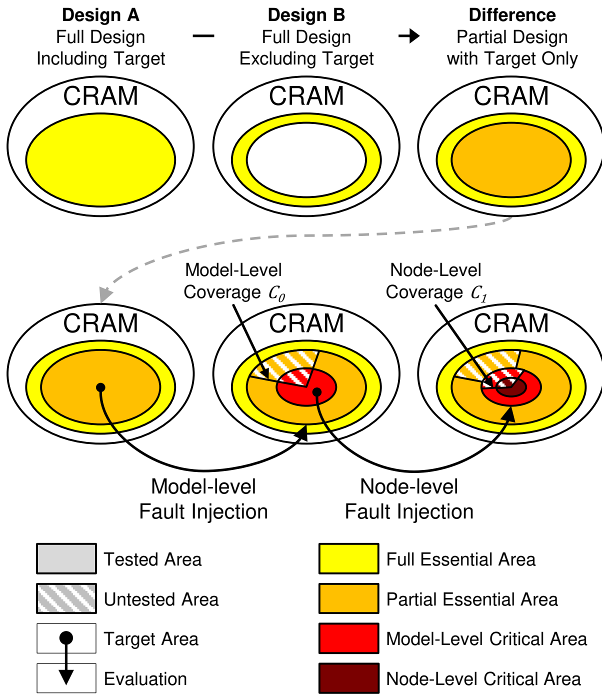


Fig. 2. Hierarchical fault-injection approach.

In our hierarchical approach, fault injection is first performed at the model level (highest granularity), where injected faults are present during the execution of the entire model (all nodes). Area_E is the target area, and fault injection is performed to generate the model-level critical area ($\text{Area}_\text{M,C}$), which is the subset of Area_E that is vulnerable to model-level faults that will manifest into observable events. The coverage factor C_0 refers to the fraction of the target area that has been tested. At this level, the model-level dependability metrics AVF_M, MWTF_M, and $\text{Area}_\text{M,C}$ are measured, and $\text{Area}_\text{M,C}$ can be approximated using Eq. (4).

$$\text{Area}_\text{M,C} = \text{AVF}_\text{M} \times \text{Area}_\text{E} \tag{4}$$

Next, fault injection is performed at the node level, where injected faults are present only during the execution of a randomly selected node. $\text{Area}_\text{M,C}$ is the targeted area, and fault injection is performed to generate the node-level critical area ($\text{Area}_\text{N,C}$), which is the subset of $\text{Area}_\text{M,C}$ that is vulnerable for node-level faults that cause observable events. At this level, node-level dependability metrics are measured, and $\text{Area}_\text{N,C}$ can be approximated using Eq. (5). The coverage factor C_0 inversely scales the $\text{Area}_\text{M,C}$ to account for any untested, critical bits potentially missed in the previous level of fault injection. The accuracy of $\text{Area}_\text{N,C}$ increases when the coverage factor of the model-level process increases ($C_0 \rightarrow 1$).

$$\text{Area}_\text{N,C} = \frac{1}{C_0} \times \text{AVF}_\text{N} \times \text{Area}_\text{M,C} \tag{5}$$

B. Fault-Injection Procedure

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

In our evaluation of the DPU, a custom interface monitor (IM), illustrated in Fig. 1, was created to monitor the interrupt and all AXI interfaces of the DPU (one instruction and two data interfaces). Since the DPU interrupt fires at the end of each node operation, the IM can calculate checksums of each interface at the node level. Consequently, the IM can identify errors at the node level, including harmless faults that may corrupt intermediate data but may be masked by a later node (e.g., faulty FMs generated from convolutional nodes masked by later pooling or activation nodes).

At the end of each fault-injection iteration, the execution output and intermediate data are analyzed to classify the outcome. We focus upon two classifications of observable events, including SDC and hangs, and the criticality of the faults. SDC events are erroneous and normally undetectable outcomes of an application execution due to faults. SDC events usually occur when faults affect the data-flow subset of a design (e.g., corrupting datapaths). In our fault-injection procedure, SDC events are observed if the checksum of the predicted output does not equal the checksum of the golden (fault-free) output. SDC events can also be observed if any intermediate checksums generated by the IM are not equal to the checksums of a golden execution. Hang events refer to the nonperformance of the application execution due to faults. Hang events usually occur when faults affect the control-flow subset of a design (e.g., corrupting finite-state machines). In our fault-injection procedure, hang events are observed if a model execution is preempted to abort by timeout prior to completion. The DPU runtime software was configured to timeout after 3 seconds if the execution did not finish.

Criticality is associated with SDC events and refers to the negative impact a fault has on inference accuracy. The criticality of SDC events can vary broadly, from low criticality (e.g., few bad pixels) to high criticality (e.g., severe distortions). Depending upon mission requirements, SDC with low criticality may be acceptable. SDC events with criticality below a user-specified tolerance threshold can be classified as tolerable SDC (SDC_T), and SDC events above this threshold can be classified as critical SDC (SDC_C). SDC_C events are the primary classification used to analyze dependability. Due to high redundancy in the parameters of a model, DL algorithms have been demonstrated to have an inherent tolerance to faults [23]. Therefore, the SDC criticality is essential for an accurate dependability assessment. When an SDC event is

Authorized licensed use limited to: University of Pittsburgh. Downloaded on April 08,2022 at 19:08:38 UTC from IEEE Xplore. Restrictions apply.

observed, the criticality can be measured as the difference in accuracy between the predicted output and golden output using a standard metric depending upon the DL task (e.g., mIoU or F1 metrics for semantic segmentation).

IV. Evaluation

In our evaluation, we quantify and analyze the accuracy, area, performance, energy-efficiency, and dependability characteristics of four semantic-segmentation models (ENet, ESPNet, FPN, and U-Net, shown in Table I) on four configurations of the DPU (B512, B800, B1024, and B1152, shown in Table II) on two generations of Xilinx SoC platforms: the TUL PYNQ-Z2 (PYNQ-Z2) and UltraZed-EG (UZED-EG), which feature a Z7020 and ZU3EG, respectively. The DL models are based on the Caffe models in [31] and were modified to use the Potsdam dataset [32] in 512×512 RGB image patches. The pixel parallelism (PP), input channel parallelism (ICP), and output channel parallelism (OCP) parameters correspond to the convolution architecture of the DPU, and the peak number of operations per cycle is equal to 2×PP×ICP×OCP. The DPU exposes 64-bit AXI3 and 128-bit AXI4 interfaces for the Zynq-7000 and Zynq-MPSoC, respectively. The DPU v3.1 IP is configured with one DPU core, low RAM and DSP usage, and extras enabled (channel augmentation, depth-wise convolution, average pool, ReLU, Leaky ReLU, and ReLU6). The trained DL models are quantized and compiled for the DPU using the DNN Development Kit (DNNDK; Vitis AI predecessor) v3.1 flow, with PetaLinux v2019.2 and Vivado v2020.1.

TABLE I
Evaluated DL Models Compiled for the DPU

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

TABLE II
DPU Convolution Architectures

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

A. Accuracy

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

TABLE III
Model Accuracy

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

B. Resource Utilization

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

TABLE IV
DPU Resource Utilization

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

C. Performance and Energy-Efficiency

The inference performance, in frames per second (FPS), and energy-efficiency, in FPS per Watt (FPS/W), for each DL solution are shown in Table VI. Both performance and energy-efficiency are dependent upon both the DL model (e.g., network, operations, size, and data-flow) and accelerator (e.g., number of PEs and other computational units, bandwidth and caching, and operating frequency). Vitis AI (and DNNDK) provide runtime tools and libraries that can profile the inference performance and DPU utilization of a DL model. Furthermore, a power meter was used to measure

Authorized licensed use limited to: University of Pittsburgh. Downloaded on April 08,2022 at 19:08:38 UTC from IEEE Xplore. Restrictions apply.

the board power when the DL solution was (1) active and (2) unloaded (i.e., the FPGA is programmed with the design with all cells and nets associated with the DPU removed). The difference is approximately the power consumption exclusive to the DPU. The power of the unloaded DPU was measured at approximately 2.1W and 6.5W for the PYNQ-Z2 and UZED-EG, respectively.

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

Next, we compare DL models in order of performance. ENet achieves the best performance due to low parameter count and workload MACs despite a medium DPU utilization (51%). Next, FPN is second due to a high DPU utilization (83%) despite high parameter count and workload MACs. Next, ESPNet is third due to having the lowest DPU utilization (22%) despite also having the lowest parameter count, workload MACs. Next, U-Net is fourth due to having the highest parameter count, workload MACs despite also having the highest DPU utilization (94%).

When compared cycle-per-cycle, the Z7020 and ZU3EG both have similar inference performance. However, the maximum frequency is substantially higher for the ZU3EG than the Z7020, possibly due to a combination of (1) generational differences between both FPGA architectures (i.e., combined configurable logic blocks, added control sets, distribution RAM control, and flip-flop I/O, upgraded tile-based columnar architecture to improve flexibility and logic and routing efficiency in UltraScale and UltraScale+ FPGAs compared to 7-Series FPGAs [33]) and (2) higher bandwidth in the UZED-EG compared to the PYNQ-Z2 (128-bit AXI4 interface and DDR4 memory versus 64-bit AXI3 and DDR3 memory).

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

D. Dependability

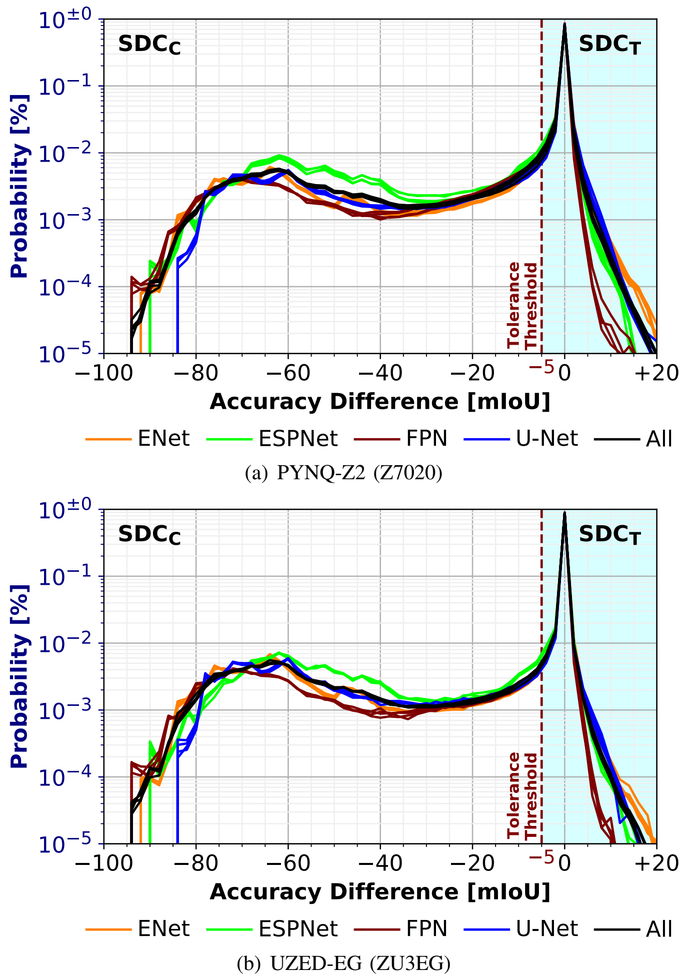
In this section, we apply our hierarchical CRAM fault-injection approach to evaluate and analyze the susceptibility of each DL solution (model, configuration, and SoC) to CRAM faults. Dependability metrics are quantified at both the model and node levels and are used to compare tradeoffs between options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

1) Model-Level Analysis: At the model level, fault injection is performed on the full DL solution, and the model-level AVF, MWTF, and critical area are calculated using Eq. (1), Eq. (2), and Eq. (4), respectively, and are shown in Table VI. In Fig. 3, SDC criticality is represented as a histogram to illustrate the probabilities of losses in inference accuracy due to CRAM faults. Fig. 4 illustrates experimental samples including the input image, ground-truth label mask, golden output, and SDC outputs with varied criticality. To demonstrate the significance of SDC criticality in our dependability analysis, we assume a tolerance threshold of -5%, where an mIoU difference greater than or equal to -5% is tolerable.

Most fault injections resulted as correct outputs or SDC_T with minor defects (mIoU difference near 0%). This observation is supported by related works that have demonstrated an inherent fault tolerance in DL algorithms [23]. A few fault injections resulted in SDC_T that improved the inference accuracy considerably (mIoU 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. The remaining fault injections resulted in SDC_C (mIoU difference less than the -5% tolerance threshold). Most notably, for each model, the shape of the distribution is similar across all tested DPUs for both SoCs. This observation indicates that SDC criticality is possibly more dependent on the characteristics of the DL model than the characteristics of the DPU.

Fig. 5 shows the SDC-critical area averaged across all DL solutions in terms of SDC_T and SDC_C. For both SoCs, the SDC_T and SDC_C-critical areas both increase as the DPU scales. However, both the AVF and critical areas increase more rapidly in the Z7020 compared to the ZU3EG, possibly due to (1) generational differences between both FPGA architectures or (2) substantially fewer resources in the Z7020 resulting in higher FPGA design resource utilization. This trend is not observed for the average hang-critical area, which increases slightly as the DPU scales. For both SoCs, the ratio of SDC criticality (SDC_T to SDC_C) increases as the DPU scales, possibly due to the increasing ratio of functional to faulty PEs. For example, each PE in B512 will process more FM data and parameters than each PE in B1152 due

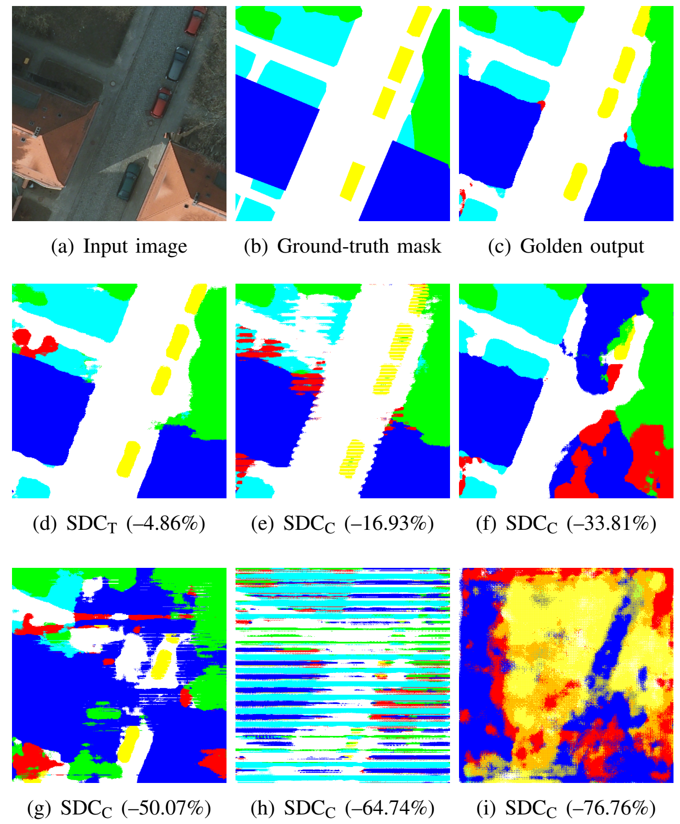
Authorized licensed use limited to: University of Pittsburgh. Downloaded on April 08,2022 at 19:08:38 UTC from IEEE Xplore. Restrictions apply.



(a) PYNQ-Z2 (Z7020)

(b) UZED-EG (ZU3EG)

Fig. 3. Impact of CRAM faults on mIoU with distribution of mIoU difference in SDC events by DL solution for the (a) PYNQ-Z2 (Z7020) and (b) UZED-EG (ZU3EG). Multiple histograms represented as line plots are overlaid. 60 bins with widths of 2% mIoU loss per bin.



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

(d) SDC$_T$ (–4.86%) (e) SDC$_C$ (–16.93%) (f) SDC$_C$ (–33.81%)

(g) SDC$_C$ (–50.07%) (h) SDC$_C$ (–64.74%) (i) SDC$_C$ (–76.76%)

Fig. 4. Experiment samples including input image, ground-truth label mask, golden output (86.33% mIoU), and variety of SDC outputs (with mIoU difference).

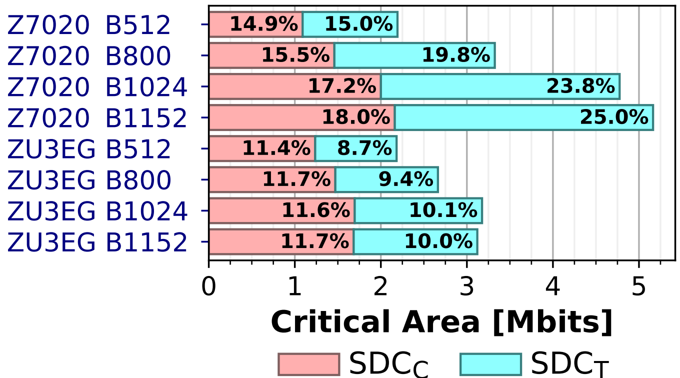


Fig. 5. Average SDC-critical area of DPUs in terms of SDC$_T$ and SDC$_C$. Percentage with respect to total area of DPU.

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

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

Using CRÈME96, the SEE fault rates for the low-Earth orbit (LEO; particularly ISS orbit) and geostationary orbit (GEO) environments were predicted for various resource types of the Z7020, assuming solar-minimum conditions and 100 mils of spherical, aluminum shielding. Using Eq. (3), the predicted SEE rates are scaled by the DPU resource utilization (Table IV) and AVF of SDC (Table VI) to approximate the SDC rates of each DPU for both orbital environments, as shown in Table VI. The SDC rates are approximately

Authorized licensed use limited to: University of Pittsburgh. Downloaded on April 08,2022 at 19:08:38 UTC from IEEE Xplore. Restrictions apply.

proportional to the SDC-critical area of the DL solution. The substantially lower SDC rates in ZU3EG compared to the Z7020 designs are due to the reduced SEE susceptibility in UltraScale+ FPGAs compared to 7-Series FPGAs [34].

2) Node-Level Analysis: At the node level, fault injection is performed on randomly selected nodes of each DL model for each DL solution, and fault injection is performed exclusively on the model-level critical area vulnerable to both SDC and hangs. The node-level critical areas are calculated using Eq. (5). Fig. 6 illustrates the sequence of nodes executed by the DPU for each DL model, including the SDC_C-critical area (tolerance threshold of -5%) and set of operations performed by each node. Nodes with more than one operation are referred to as supernodes which have fused multiple node operations into one to improve efficiency. Convolution-based operations include convolution (ConvNd) and deconvolution (DeConvNd), and miscellaneous operations include concatenation (Concat), pooling (Pooling), elementwise (Eltwise), scale (Scale), and rectified linear unit (ReLU).

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

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

3) Fault-Injection Evaluation: To demonstrate the efficiency and accuracy of our hierarchical fault-injection approach, we compare with a direct fault-injection approach targeting solely the essential area of the DPU for both model and node levels. First, we compare the fault-injection efficiency and speed-up. The total number of possible fault injections for both direct and hierarchical approaches is represented by Eq. (6) and Eq. (7), respectively, where N_m is the number of nodes for each model $m \in M$.

$$\text{Injections}_{\text{Direct}} = \sum_{m \in M} (N_m + 1) \times \text{Area}_{\text{E}} \quad (6)$$

$$\text{Injections}_{\text{Hierarchical}} = \sum_{m \in M} (\text{Area}_{\text{E}} + N_m \times \text{Area}_{\text{M,C}}) \quad (7)$$

For example, in the B512 configuration for the Z7020, the partial essential area (Area_{E}) is 7.3 Mbits, and the aggregated critical area ($\text{Area}_{\text{M,C}}$), which is the union of critical areas across all evaluated DL models, is 2.6 Mbits (AVF_{M} = 35%). To evaluate all four models (397 nodes) at the node level, a direct approach has 2.91 billion possible injections, whereas our hierarchical approach has 1.04 billion possible injections, thus a maximum efficiency improvement of 2.7×. Similarly, the B512 configuration for the ZU3EG, with Area_{E} and aggregated $\text{Area}_{\text{M,C}}$ equal to 10.9 Mbits and 2.5 Mbits, respectively, there is a maximum efficiency improvement of 4.2×. Efficiency is dependent upon the ratio of $\text{Area}_{\text{M,C}}$ to Area_{E}. As shown in Fig. 5, this ratio increases rapidly as the DPU scales for the Z7020, resulting in efficiency decreasing from 2.7× (B512) to 2.1× (B1152). However, this ratio increases relatively slightly for the ZU3EG, so efficiency decreases slightly from 4.2× (B512) to 4.1× (B1152).

TABLE V
FAULT-INJECTION ACCURACY FOR B512 CONFIGURATION ON Z7020

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

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

V. CONCLUSION

Modern spacecraft increasingly require more computational capabilities to enable compute-intensive, deep-learning (DL) methods that can enhance onboard analysis, spacecraft autonomy, and intelligent applications. Commercial-off-the-shelf FPGAs and hybrid SoCs, which provide the architectural capabilities for the onboard acceleration of DL algorithms, can address these requirements. However, commercial FPGAs and SoCs are highly susceptible to radiation-induced single-event effects (SEEs) that can affect dependability.

Authorized licensed use limited to: University of Pittsburgh. Downloaded on April 08,2022 at 19:08:38 UTC from IEEE Xplore. Restrictions apply.

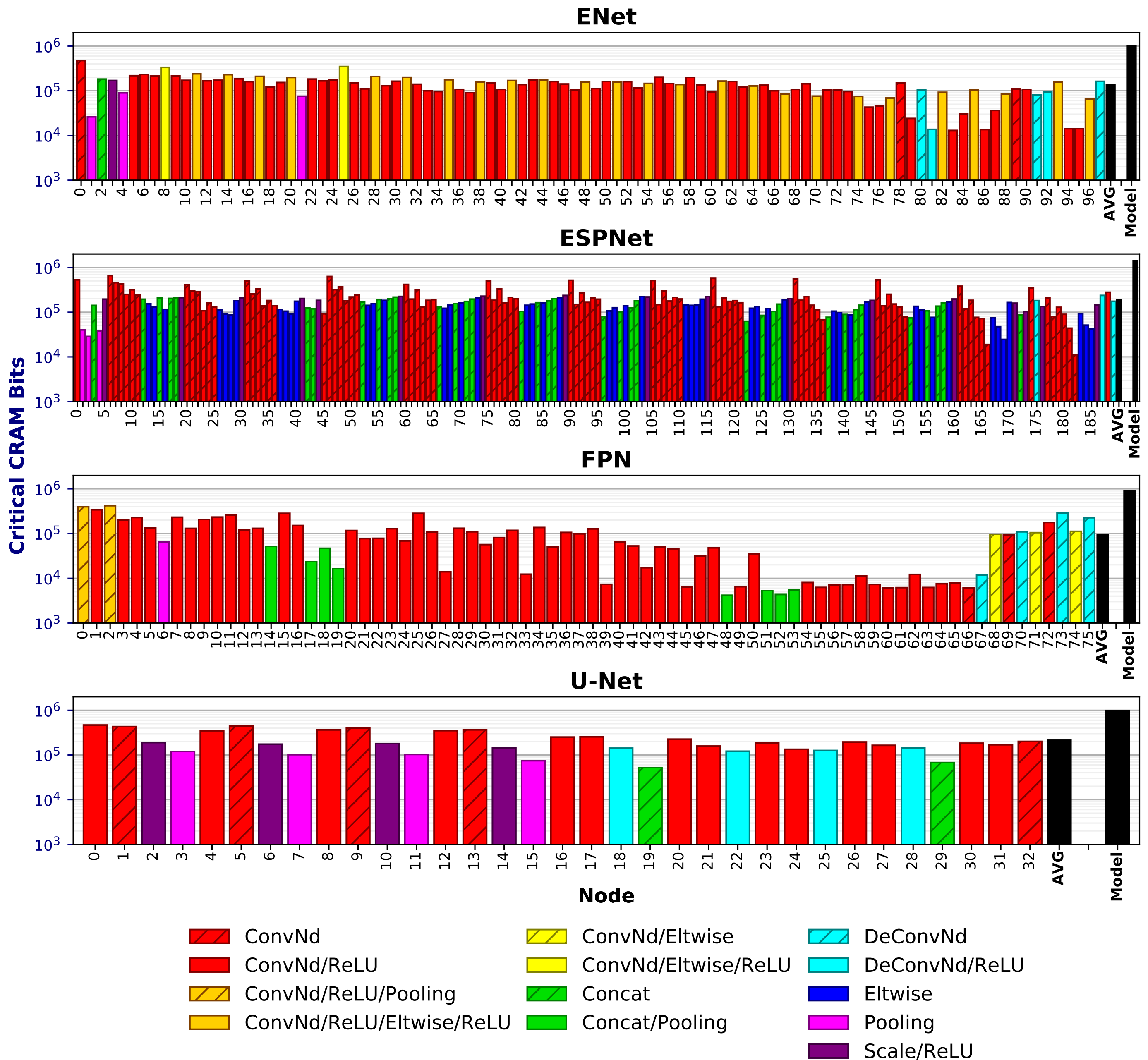


Fig. 6. SDC_C-critical area and operations by node for each DL model for the B512 configuration of the DPU on the Z7020.

Furthermore, with a broad variety of DL tasks and a diverse collection of DL models, each with its own accuracy, resource utilization, performance, energy-efficiency, and dependability characteristics, a methodology is required to evaluate and compare the tradeoffs between DL models and accelerators to select the optimal design.

In this paper, we proposed a comprehensive methodology to evaluate the tradeoffs between DL models and accelerators. With an emphasis on dependability, we proposed a hierarchical fault-injection approach that continually narrows the set of targeted bits by removing bits with noncritical representation and thereby accelerating the fault-injection process. Compared to direct fault injection, our approach achieves an efficiency improvement of 2.1-2.7× and 4.1-4.2× for the Zynq-7000 and Zynq-MPSoC, respectively, to evaluate all DL solutions in this study. Furthermore, we describe methods to analyze the dependability of DL models and accelerators at both the model and node levels and in terms of the architectural vulnerability factor, mean-work-to-failure, and critical area. Finally, using this methodology, we evaluated, analyzed, and compared the tradeoffs and trends of four semantic-segmentation models across four configurations of the Xilinx DPU for two generations of Xilinx SoCs (Zynq-7000 SoC and Zynq UltraScale+ MPSoC). Our evaluation, which was conducted on emulators

Authorized licensed use limited to: University of Pittsburgh. Downloaded on April 08,2022 at 19:08:38 UTC from IEEE Xplore. Restrictions apply.

of flight hardware that is currently deployed in space missions, demonstrates that compute-intensive DL applications can be dependably executed onboard for next-generation missions.

Opportunities for future work include (1) applying this methodology to evaluate DL models for other DL tasks (e.g., classification, detection, and localization) and alternative DL accelerators, (2) devising and evaluating adaptive and selective methods for efficient SEE mitigation (e.g., adapting between DL models or accelerators at runtime, or applying SEE mitigation to the most vulnerable nodes), and (3) exploring extrapolation methods using fault-injection results of one DL model and DPU to predict the results for other configurations of the DPU.

REFERENCES

[1] B. DARPA, "Blackjack Pit Boss (BAA HR001119S0012)," April 2019.

[2] 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.

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

[4] M. Esposito, S. S. Conticello, M. Pastena, and B. C. Domínguez, "In-orbit demonstration of artificial intelligence applied to hyperspectral and thermal sensing from space," in *CubeSats and SmallSats for Remote Sensing III*, vol. 11131, International Society for Optics and Photonics. SPIE, 2019, pp. 88 – 96. [Online]. Available: https://doi.org/10.1117/12.2532262

[5] C. Scolese, "2020 Small Satellite Conference keynote address," in *Proceedings of the 34th Annual AIAA/USU Conference on Small Satellites*. Logan, UT: AIAA, August 2020.

[6] A. E. Johnson, Y. Cheng, J. F. Montgomery, N. Trawny, B. Tweddle, and J. X. Zheng, *Real-Time Terrain Relative Navigation Test Results from a Relevant Environment for Mars Landing*. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2015-0851

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

[8] J. E. Ball, D. T. Anderson, and C. S. C. Sr., "Comprehensive survey of deep learning in remote sensing: theories, tools, and challenges for the community," *Journal of Applied Remote Sensing*, vol. 11, no. 4, pp. 1 – 54, 2017. [Online]. Available: https://doi.org/10.1117/1.JRS.11.042609

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

[10] A. Geist, C. Brewer, M. Davis, N. Franconi, S. Heyward, T. Wise, G. Crum, D. Petrick, R. Ripley, C. Wilson, and T. Flatley, "SpaceCube v3.0 NASA next-generation high-performance processor for science applications," in *Proceedings of the 33rd Annual AIAA/USU Conference on Small Satellites*. Logan, UT: AIAA, 2019, pp. 1–9.

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

[12] F. Siegle, T. Vladimirova, J. Ilstad, and O. Emam, "Mitigation of radiation effects in SRAM-based FPGAs for space applications," *ACM Comput. Surv.*, vol. 47, no. 2, pp. 37:1–37:34, 2015. [Online]. Available: http://doi.acm.org/10.1145/2671181

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

[14] A. J. Tylka, J. H. Adams, P. R. Boberg, B. Brownstein, W. F. Dietrich, E. O. Flueckiger, E. L. Petersen, M. A. Shea, D. F. Smart, and E. C. Smith, "CREME96: A revision of the Cosmic Ray Effects on Micro-Electronics Code," *IEEE Transactions on Nuclear Science*, vol. 44, no. 6, pp. 2150–2160, December 1997.

[15] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, P. Martinez-Gonzalez, and J. Garcia-Rodriguez, "A survey on deep learning techniques for image and video semantic segmentation," *Applied Soft Computing*, vol. 70, pp. 41–65, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494618302813

[16] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Cham: Springer International Publishing, 2015, pp. 234–241.

[17] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "ENet: A deep neural network architecture for real-time semantic segmentation," *CoRR*, vol. abs/1606.02147, 2016. [Online]. Available: http://arxiv.org/abs/1606.02147

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

[19] S. Mehta, M. Rastegari, A. Caspi, L. Shapiro, and H. Hajishirzi, "ESPNet: Efficient spatial pyramid of dilated convolutions for semantic segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[20] S. Mittal, "A survey of FPGA-based accelerators for convolutional neural networks," *Neural Computing and Applications*, vol. 32, no. 4, pp. 1109–1139, February 2020. [Online]. Available: https://doi.org/10.1007/s00521-018-3761-1

[21] Xilinx, *Zynq DPU*, v3.3 ed., Xilinx, December 2020, Xilinx Product Guide (PG338).

[22] ——, *Vitis AI User Guide*, v1.3 ed., Xilinx, December 2020, Xilinx User Guide (UG1414).

[23] C. Torres-Huitzil and B. Girau, "Fault tolerance in neural networks: Neural design and hardware implementation," in *2017 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, December 2017, pp. 1–6.

[24] B. Du, S. Azimi, C. de Sio, L. Bozzoli, and L. Sterpone, "On the reliability of convolutional neural network implementation on SRAM-based FPGA," in *2019 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, October 2019, pp. 1–6.

[25] F. F. dos Santos, C. Lunardi, D. Oliveira, F. Libano, and P. Rech, "Reliability evaluation of mixed-precision architectures," in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, February 2019, pp. 238–249.

[26] F. Libano, B. Wilson, M. Wirthlin, P. Rech, and J. Brunhaver, "Understanding the impact of quantization, accuracy, and radiation on the reliability of convolutional neural networks on FPGAs," *IEEE Transactions on Nuclear Science*, vol. 67, no. 7, pp. 1478–1484, July 2020.

[27] F. Benevenuti, F. Libano, V. Pouget, F. L. Kastensmidt, and P. Rech, "Comparative analysis of inference errors in a neural network implemented in SRAM-based FPGA induced by neutron irradiation and fault injection methods," in *2018 31st Symposium on Integrated Circuits and Systems Design (SBCCI)*, August 2018, pp. 1–6.

[28] F. Libano, B. Wilson, J.-P. Anderson, M. J. Wirthlin, C. Cazzaniga, C. Frost, and P. Rech, "Selective hardening for neural networks in FPGAs," *IEEE Transactions on Nuclear Science*, vol. 66, no. 1, pp. 216–222, January 2019.

[29] G. Gambardella, J. Kappauf, M. Blott, C. Doehring, M. Kumm, P. Zipf, and K. Vissers, "Efficient error-tolerant quantized neural network accelerators," in *2019 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, October 2019, pp. 1–6.

[30] S. Sabogal, A. George, and G. Crum, "ReCoN: A reconfigurable CNN acceleration framework for hybrid semantic segmentation on hybrid SoCs for space applications," in *2019 IEEE Space Computing Conference (SCC)*, July 2019, pp. 41–52.

[31] Xilinx, "ML Caffe segmentation tutorial," Xilinx, December 2020, accessed: 2021-02-01.

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

[33] Xilinx, *UltraScale Architecture Configurable Logic Block*, v1.5 ed., Xilinx, February 2017, xilinx User Guide (UG574).

[34] D. S. Lee, M. King, W. Evans, M. Cannon, A. Pérez-Celis, J. Anderson, M. Wirthlin, and W. Rice, "Single-event characterization of 16 nm FinFET Xilinx UltraScale+ devices with heavy ion and neutron irradiation," in *2018 IEEE Nuclear Space Radiation Effects Conference (NSREC 2018)*, July 2018, pp. 1–8.

Authorized licensed use limited to: University of Pittsburgh. Downloaded on April 08,2022 at 19:08:38 UTC from IEEE Xplore. Restrictions apply.

TABLE VI

Evaluation results for performance, energy-efficiency, and model-level dependability.

SoC	DPU	Model	Performance			MWTF		AVF (%)				SDC-Critical Area	SDC Rates ($SDC \cdot dev^{-1} \cdot day^{-1}$)	
			FPS	FPS/W	Util (%)	FPS	FPS/W	SDC	SDC_T	SDC_C	Hangs	±95% CI Error	LEO	GEO
Z7020 (PYNQ-Z2)	**B512** 250/500MHz	ENet	13.2	4.9	41.9	63.4	23.6	31.0	17.2	13.8	2.8	2,270,773 ± 3,954	0.19	0.11
		ESPNet	5.5	2.3	15.9	18.6	7.8	31.4	12.0	19.3	3.2	2,299,408 ± 4,180	0.19	0.11
		FPN	6.0	2.1	80.5	33.4	12.0	27.9	15.5	12.4	2.7	2,045,829 ± 4,227	0.17	0.10
		U-Net	1.2	0.5	88.3	5.9	2.6	28.8	15.4	13.5	2.7	2,114,796 ± 4,466	0.18	0.10
		Average	6.5	2.5	56.7	29.4	11.6	29.8	15.0	14.8	2.8	2,182,701 ± 2,100	0.18	0.10
	B800 250/500MHz	ENet	13.3	4.1	26.9	58.8	18.0	36.2	22.3	13.9	2.3	3,406,655 ±12,562	0.33	0.18
		ESPNet	6.2	2.2	11.4	18.0	6.4	36.6	15.8	20.8	2.6	3,445,554 ±12,450	0.33	0.18
		FPN	7.8	2.0	67.3	39.2	10.2	33.6	20.9	12.7	2.3	3,165,791 ±13,581	0.30	0.16
		U-Net	1.5	0.5	73.6	6.9	2.2	34.5	20.6	13.9	2.3	3,247,737 ±11,928	0.31	0.17
		Average	7.2	2.2	44.8	29.2	9.0	35.2	19.9	15.3	2.4	3,316,434 ± 6,298	0.32	0.17
	B1024 200/400MHz	ENet	14.8	5.0	29.3	53.5	18.2	42.5	27.3	15.2	2.6	4,958,499 ±15,683	0.50	0.27
		ESPNet	6.5	2.6	11.8	14.8	5.9	42.2	18.0	24.1	2.9	4,917,392 ±16,489	0.50	0.27
		FPN	8.9	2.3	75.3	36.1	9.5	39.3	25.0	14.3	2.5	4,587,410 ±16,751	0.46	0.25
		U-Net	1.7	0.6	81.2	6.5	2.2	39.7	24.4	15.3	2.5	4,627,702 ±14,580	0.47	0.25
		Average	8.0	2.6	49.4	26.2	8.5	40.9	23.7	17.2	2.6	4,772,751 ± 7,907	0.48	0.26
	B1152 167/333MHz	ENet	12.7	4.2	26.9	44.7	14.7	43.9	28.6	15.3	2.3	5,276,269 ±18,517	0.55	0.29
		ESPNet	5.8	2.3	11.3	12.4	4.9	44.4	19.5	24.9	2.7	5,344,894 ±18,015	0.56	0.29
		FPN	8.6	2.3	77.8	32.8	8.7	41.3	26.4	14.9	2.3	4,964,799 ±19,871	0.52	0.27
		U-Net	1.5	0.5	75.1	5.3	1.8	41.8	25.9	15.9	2.3	5,026,491 ±17,671	0.52	0.28
		Average	7.2	2.3	47.8	22.2	7.2	42.8	25.1	17.7	2.4	5,153,113 ± 9,236	0.54	0.28
ZU3EG (UZED-EG)	**B512** 375/750MHz	ENet	23.3	7.4	49.3	158.5	50.2	20.6	9.3	11.3	2.2	2,240,818 ± 5,569	0.03	0.08
		ESPNet	9.7	3.4	18.8	51.7	17.8	21.7	7.4	14.3	2.5	2,357,773 ± 9,253	0.03	0.09
		FPN	9.2	3.1	82.5	80.9	27.4	18.1	9.0	9.0	2.2	1,966,019 ± 7,946	0.03	0.07
		U-Net	1.8	0.7	91.4	12.8	5.1	19.3	8.2	11.2	2.2	2,100,344 ± 8,769	0.03	0.08
		Average	11.0	3.8	60.5	74.5	25.9	19.9	8.5	11.5	2.3	2,166,238 ± 3,714	0.03	0.08
	B800 375/750MHz	ENet	24.8	6.6	33.5	165.8	43.9	21.3	9.9	11.5	2.0	2,692,359 ±22,366	0.05	0.13
		ESPNet	11.5	3.5	14.2	60.4	18.3	22.2	7.8	14.4	2.4	2,803,479 ±22,630	0.05	0.13
		FPN	12.2	3.1	70.6	102.7	26.3	20.0	10.7	9.3	2.0	2,526,930 ±21,299	0.04	0.12
		U-Net	2.4	0.7	77.0	16.0	4.9	20.8	9.4	11.5	2.0	2,631,772 ±22,479	0.05	0.13
		Average	12.7	3.6	48.9	84.3	23.7	21.1	9.4	11.7	2.1	2,663,635 ±11,095	0.05	0.13
	B1024 300/600MHz	ENet	25.2	7.5	33.2	168.7	50.1	22.5	11.2	11.3	1.8	3,301,512 ±26,157	0.07	0.18
		ESPNet	11.7	3.8	14.2	60.9	19.5	22.9	8.4	14.5	2.0	3,363,317 ±26,821	0.07	0.18
		FPN	14.1	3.5	79.6	119.5	29.5	20.2	11.0	9.2	1.7	2,968,996 ±24,144	0.06	0.16
		U-Net	2.7	0.8	85.6	18.5	5.6	20.9	9.5	11.4	1.7	3,071,384 ±27,529	0.06	0.17
		Average	13.4	3.9	53.1	89.1	25.7	21.6	10.0	11.6	1.8	3,176,302 ±13,051	0.06	0.17
	B1152 300/600MHz	ENet	25.0	6.6	29.4	170.3	45.1	21.9	10.7	11.2	1.7	3,150,305 ±25,030	0.07	0.18
		ESPNet	11.4	3.5	12.2	59.6	18.5	22.5	8.1	14.4	2.0	3,240,436 ±26,008	0.07	0.19
		FPN	16.3	3.7	81.5	133.4	29.9	20.9	11.5	9.4	1.7	3,011,382 ±26,530	0.06	0.18
		U-Net	2.8	0.8	77.2	18.6	5.3	21.2	9.8	11.4	1.7	3,052,462 ±28,955	0.07	0.18
		Average	13.9	3.7	50.1	91.3	24.4	21.6	10.0	11.6	1.8	3,113,646 ±13,264	0.07	0.18

Authorized licensed use limited to: University of Pittsburgh. Downloaded on April 08,2022 at 19:08:38 UTC from IEEE Xplore. Restrictions apply.