HAL open science

Impact of High-Level-Synthesis on Reliability of Artificial Neural Network Hardware Accelerators

Marcello Traiola, Fernando Fernandes dos Santos, Paolo Rech, Carlo Cazzaniga, Olivier Sentieys, Angeliki Kritikakou

To cite this version:

Marcello Traiola, Fernando Fernandes dos Santos, Paolo Rech, Carlo Cazzaniga, Olivier Sentieys, et al.. Impact of High-Level-Synthesis on Reliability of Artificial Neural Network Hardware Accelerators. IEEE Transactions on Nuclear Science, 2024, pp.1-9. 10.1109/TNS.2024.3377596 . hal-04514579

HAL Id: hal-04514579

https://inria.hal.science/hal-04514579v1

Submitted on 21 Mar 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Impact of High-Level-Synthesis on Reliability of Artificial Neural Network Hardware Accelerators

Marcello Traiola, Fernando Fernandes dos Santos, Paolo Rech, Carlo Cazzaniga
Olivier Sentieys and Angeliki Kritikakou

***Abstract*—Dedicated hardware is required to efficiently execute the highly resource-demanding modern Artificial Neural Networks (ANNs). The high complexity of ANNs systems has motivated the use of High-Level Synthesis (HLS) tools, which increase design abstraction. The higher abstraction reduces the implementation of FPGA hardware details visible to the designer, making an accurate reliability evaluation challenging. When ANN hardware accelerators are used in safety-critical systems, reliability becomes paramount, and in order to have a realistic reliability evaluation, physical fault injection, such as beam testing, is mandatory. Existing reliability analysis approaches focus on specific ANN hardware accelerator designs, but when HLS tools are used, the tool flow and design decisions can impact reliability. Therefore, we evaluate the error rate of ANN hardware accelerators generated by HLS tools under high-energy neutrons and explore the impact of HLS parameters on reliability. Our results show that by tweaking hardware parameters, such as the reuse of resources, can increase the error rate linearly. Furthermore, the generated ANN hardware accelerator with the best tradeoff of area and execution cycles can deliver 15× more correct executions than the least optimized one, despite its increased error rate.**

I. Introduction

Artificial Neural Networks (ANNs) are one of the most intensively and widely used predictive models in the field of Machine Learning (ML) [1]. ANNs have proven outstanding results for many complex tasks and applications, such as object recognition in images/videos, natural language processing, satellite image recognition, robotics, aerospace, smart healthcare, and autonomous driving. Their incredibly high effectiveness and flexibility come at the price of enormous algorithmic complexity, thus requiring huge computational power. To support energy-intensive data movement, speed of computation, and large memory resources required by AI to achieve its full potential, custom and embedded ANN accelerators are required [2]. SRAM-based Field Programmable Gate Arrays (FPGAs), thanks to their flexibility and efficiency, are heavily used to prototype and implement dedicated ANN accelerators.

Unfortunately, the increased complexity of modern ANN systems makes explicit dedicated low-level hardware design approaches less effective [3]. To mitigate the design complexity, high-level specification languages, such as C/C++ and Chisel, are used to design hardware circuits in higher abstraction. With the help of High-Level Synthesis (HLS) tools, such high-level descriptions can be interpreted to create digital hardware that implements the same functionality, as it automatically compiles the functional description of a design into a Register-Transfer Level (RTL) implementation that can be synthesized on FPGAs. Using high-level specification languages and HLS tools makes the circuit design less complex compared to implementations based on Hardware Description Languages (HDL) [4]. Indeed, with this approach, the circuit can easily be modified, expanded, and verified, speeding up significantly the hardware design process. As a result, custom and complex NN accelerators can be designed faster through HLS.

Reliability becomes paramount when ANN accelerators are employed in safety-critical systems, such as autonomous vehicles, aerospace, or military applications. The abstraction introduced by HLS, while providing unquestionable benefit in terms of design time and flexibility, has the drawback of hiding from the programmer details about the translation to lower hardware layers and the final implementation, possibly hiding weak configurations and limiting the possibility of predicting the design reliability. To evaluate the ANN accelerator's reliability, faults have to be injected into the system. Fault injection can be achieved through simulation [5]–[9] and irradiation experiments [10]–[14]. Although fault injection through simulation provides early soft error assessment, it is less accurate than irradiation experiments [15]. Therefore, testing ANN accelerators under radiation is crucial in providing information about the hardware error model and its impact on safety-critical systems.

Recent studies have evaluated the reliability of machine learning on high-energy neutron beams and have demonstrated that ANN-based systems do not meet the reliability criteria for safety-critical systems. Existing works focus on platforms with generic Graphics Processing Units (GPUs), processors, and ANN accelerators. For instance, reliability analysis is performed for CNNs on NVIDIA GPUs [10] and Arm microprocessors [11]. Existing works on analyzing dedicated hardware for ANN under high-energy neutron beams focus on specialized accelerators and FPGA implementations. For instance, a neuromorphic computer architecture is analyzed in [12], a commercial-off-the-shelf EdgeAI device in [13], and Google Tensor Processing Units in [14]. FPGA implementations are designed either with HDL, such as ANN and MNIST [16], [17], Support Vector Machine [18] and approximated CNNs [19], or HLS tools for image classification CNN [20], and quantized ZynqNet [21]. However, to the

Marcello Traiola, Fernando Fernandes dos Santos, Angeliki Kritikakou, and Olivier Sentieys are with Univ Rennes, Irisa, INRIA, CNRS, France; Angeliki Kritikakou is also with Institut Universitaire de France (IUF), France
Paolo Rech is with University of Trento, Italy
Carlo Cazzaniga is with ISIS Facility, Rutherford Appleton Laboratory, UK

best of our knowledge, no existing work has evaluated how HLS tools and different design optimizations can impact the reliability of obtained ANN accelerators under high-energy neutrons. The closest work is [22], [23], where a set of HLS optimizations through directives (pipeline, array partition, inline, unroll) are analyzed regarding the Single Event Upset (SEU) susceptibility of the obtained design for common benchmarks, e.g., matrix multiplication, advanced encryption standard, and adaptive differential pulse-code modulation, under heavy ions.

This work explores the impact of the HLS on the reliability of the HLS-generated designs synthesized on FPGAs. More precisely, our goal is to analyze how the design choices available through HLS tools impact the reliability of the generated hardware accelerator. To achieve that, we create five different implementations of the same ANN architecture, exploring different resource reuse parameters through HLS and mapping them to PYNQ-Z2 Xilinx boards. Then, we test their resilience properties by placing the SoC under a high-energy neutron beam. We show that the HLS-generated accelerators' error rates can increase linearly with the reuse parameters.

Moreover, we compare the reliability of bare-metal implementations with executions on top of an operating system. During our experiments, while the devices are being irradiated, we run ANNs and check the mismatches in the application output without applying any detection or mitigation techniques. To study the effects and propagation of faults, we perform post-processing to identify the faults that are more likely to reduce an ANN's reliability and affect the system critically. Our experimental data shows that even if the error rate increases based on the HLS parameters and accelerator size, the fastest accelerators can still produce 15× more correct classifications than the slowest ones. Notably, these results were obtained without implementing any fault detection or mitigation mechanisms, and no built-in fault detection features, like scrubbing or Error Correction Codes (ECC), were activated throughout all the conducted experiments. The main contributions of this work are:

- Assessing the reliability of HLS-generated HW accelerators for our neural network use case and analyzing the extent to which HLS parameters can affect the error rates of such neural networks.
- Evaluating the number of correct classifications that the highest-performing accelerator can execute before experiencing a failure without any fault detection or mitigation.
- Discuss how the operating system can impact the error rate of the hardware accelerators.

II. Background and Contributions

A. Radiation Induced Errors on ANN Accelerators

Due to the shrinking of transistors size and lower supply voltages of modern technologies [24], systems are becoming more and more sensitive to faults caused by environmental sources [25], such as radiation, leading to temporary reliability violations, called transient faults. Transient faults can affect the system behavior by changing the hardware's correct execution, manifesting in events such as corrupting data stored in memories or modifying the output of functional units, and changing the behavior of the system.

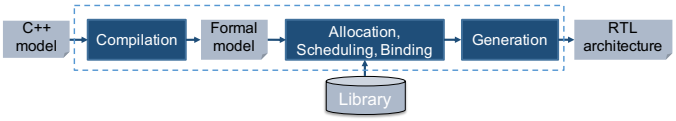


Fig. 1: HLS's design flow. The process starts with high-level language input (i.g., C++) with a standard compilation procedure. Then, the tool will generate, based on the constraints and goals of the circuit, an architecture (RTL) ready to be synthesized into the platform.

ANN accelerators, similar to traditional computing hardware, are subject to transient faults. ANNs have some inherent resilience to faults, similar to biological neural networks. This is because the statistical behavior of neural network architectures and their high space redundancy and over-provisioning naturally provide certain fault tolerance. Therefore, ANNs can partially circumvent hardware faults during the learning process. However, faults can still occur after training. Recent studies in the literature have shown that AI is not always immune. Inference can be significantly affected, leading to ANN prediction failures that are likely to have a detrimental effect on the application [6], [26].

A transient fault in the dedicated ANN accelerators may propagate through the system and lead to failure. We can classify a failure as one of the following outcomes: (1) **No effect** on the ANN output (i.e., the fault is masked, or the corrupted data is not used). (2) **Silent Data Corruption** (SDC) (i.e., an incorrect ANN output). (3) **Detected Unrecoverable Error** (DUE) (i.e., a program crash or device reboot). Note that not all errors in the output of the ANN will impact the classification. Therefore, following previous works SDC classification [5], [10] we can classify the SDCs as **Tolerable SDCs** (i.e., that do not impact classification/detection) and **Critical SDCs** (i.e., that impact classification/detection).

B. HLS design of ANN hardware accelerators

The HLS design flow takes as input a high-level description of the functionality, an RTL component library, and a set of specific design constraints. As illustrated in Figure 1, HLS initially compiles the functional specification (C++ model) to a formal model, such as a Control and Data Flow Graph (CDFG) that describes the data and control dependencies between the operations. Then, it allocates hardware resources, such as functional units, storage components, and buses. HLS schedules the operations to clock cycles, binds the operations to functional units, binds variables to storage elements and transfers to buses, and generates the RTL architecture [27]. Finally, digital implementation takes place, including logic synthesis and place and route.

Note that the HLS steps are usually coupled with each other. For example, if fewer hardware resources are allocated, the occupied area will be smaller, which leads to a longer schedule. Nonetheless, HLS allows the adjustment of the level of hardware parallelism, which results in implementations with

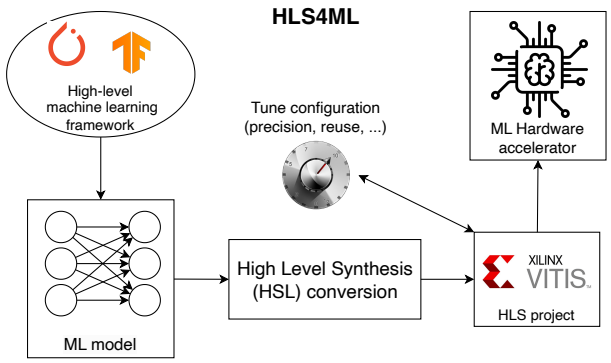


Fig. 2: HLS4ML flow [28], [29]. It allows translating conventional high-level machine learning algorithms to hardware implementation after fine-tuning numeric precision and HW reuse parameters.

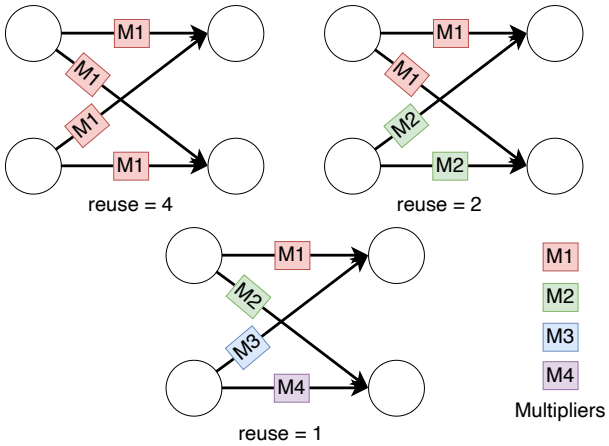


Fig. 3: The reuse parameter expresses the number of times that a given multiplier is reused when computing neuron values.

varying trade-offs between area footprint (such as occupied FPGA logic slices and LUTs) and latency.

C. Proposed analysis and contributions

For our case study, we used HLS4ML framework [28], [29] and Vitis HLS. HLS4ML is an open-source software-hardware co-design workflow to interpret and translate machine learning algorithms for FPGA implementation. Figure 2 sketches HLS4ML flow. It translates machine learning models implemented through well-established open-source high-level frameworks, such as Keras and Pytorch, into HLS-compatible high-level-description parametric models that can be configured, synthesized, and implemented to run on an FPGA [28]. HLS4ML provides configurable parameters allowing the designer to explore different implementations varying in latency, initiation interval, and resource usage. Therefore, with the help of HLS4ML, we could quickly deploy different configurations of ANN accelerators on an FPGA.

For our reliability analysis, we used a Multi-Layer Perceptron (MLP) classifier as case study. Note that, a similar approach can be applied to other ANN accelerators and ANN types. More precisely, we design ANN hardware accelerators for an MLP classifier of 28×28 pixel images composed as follows: (i) 784 input neurons, (ii) 50 hidden neurons, and (iii) 10 output neurons. The ANN is composed of two Fully Connected (FC) layers and two activation layers. The two FC layers perform, for each output neuron O_j, the weighted sum operation $O_j = \sum_{i=0}^{n} I_i \times w_{ij} + b_j$, where I_i is the i-th input of the layer, w_{ij} the i-th weight for j-th output, b_j the j-th bias, and n the number of layer inputs. After the first FC layer, there is a relu activation layer, and after the second FC layer, there is a softmax activation layer. The data type used in the accelerator for ANN weights and activations is 16-bit fixed-point, with 8 bits for the integer part and 8 bits for the fractional part. We trained the NN with the MNIST database of handwritten digits [30], reaching a top-1 accuracy of 95.7%.

Through HLS, we can explore implementations where the designed accelerator reuses multipliers to compute neuron values in a layer with different ANN inference latency. Figure 3 shows the reuse parameter that expresses the number of times a given multiplier is reused during the computation of neuron values. The simple example in the figure depicts a case where only four multiplications are needed: when the reuse parameter is equal to 4, a single multiplier (M1) is re-used four times, and thus, the four multiplications cannot occur in parallel. A reuse parameter equal to 1 means that a given multiplier will be only used once; thus, four different multipliers (M1, M2, M3, and M4) are required for the example at hand and can perform the operations in parallel. As a consequence, the value of the reuse parameter affects the latency, initiation interval, and resource usage of the final design.

Two reuse parameters are available for the ANN considered in this study, one per FC layer. Our goal is to explore how varying the value of reuse parameters impacts the reliability of the obtained designs through HLS tools. We refer to the two reuse parameters as R_1 and R_2. R_1 and R_2 values correspond to the number of times a given multiplier is reused when computing neuron values in the first and second layers, respectively. Since the first layer has 784 inputs and 50 outputs, the total number of multiplication is $784 \times 50 = 39,200$; hence, among the possible R_1 values, there are 39,200 (1 multiplier), 19,600 (2 multipliers), 3,920 (10 multipliers), 784 (50 multipliers), and 392 (100 multipliers). Since the second layer has 50 inputs and 10 outputs, the total number of multiplications is $50 \times 10 = 500$, then, among possible values for R_2, we have 500 (1 multiplier), 250 (2 multipliers), 100 (5 multipliers), 25 (20 multipliers), and 10 (50 multipliers). Table I summarizes the reuse configurations we exploit in this paper. More information about the reuse factor parameter is available in HLS4ML documentation [28], [29].

TABLE I: Reuse parameters and the number of multipliers used in each layer. 39200 multiplications are performed in the first layer and 500 in the second.

Reuse Parameters	Multipliers in Layer 1	Multipliers in Layer 2
$R_1 = 39200$, $R_2 = 500$	1	1
$R_1 = 19600$, $R_2 = 250$	2	2
$R_1 = 3920$, $R_2 = 100$	10	5
$R_1 = 784$, $R_2 = 25$	50	20
$R_1 = 392$, $R_2 = 10$	100	50

TABLE II: Hardware characteristics of the five accelerator configurations under study. The modification of HLS constraints leads to final architectures with significant differences.

Config	LUT		LUTRAM		FF		BRAM		DSP		Clock	Longest	Max freq.
(R_1xR_2)	Total	[%]	Total	[%]	Total	[%]	Total	[%]	Total	[%]	Cycles	Path	
39200x500	9811	18.44%	235	1.35%	12378	11.63%	33	23.57%	12	5.45%	39804	13.116 ns	76.241 MHz
19600x250	9469	17.80%	239	1.37%	10911	10.25%	25	17.50%	14	6.36%	19960	9.440 ns	105.927 MHz
3920x100	10443	19.63%	525	3.02%	14709	13.82%	16	11.07%	25	11.36%	4154	9.471 ns	105.590 MHz
784x25	13472	25.32%	810	4.66%	16456	15.47%	54	38.57%	80	36.36%	947	9.214 ns	108.525 MHz
392x10	18824	35.38%	1006	5.78%	25619	24.08%	102	72.50%	160	72.73%	932	8.856 ns	112.910 MHz

III. Experimental Methodology

This section discusses the experimentation methodology, devices, and metrics we employ to assess the neutron-induced error rate of various neural network accelerators generated by HLS tools.

A. ANN Accelerator Configurations

Five ANN accelerators are generated through HLS by combining different values for R_1 and R_2 parameters, named as R_1xR_2. Table II reports the hardware attributes (absolute numbers and percentages) of the tested configurations in terms of total Lookup Tables (LUT), number of LUT used as the memory (LUTRAM), number of Flip Flops (FF), number of Block RAM Tile (BRAM), number of DSPs (DSP), the number of clock cycles needed to perform an ANN inference, longest path timing in the design and corresponding maximum frequency. The maximum frequency we used in the experiments is the default one for the target board, i.e., 100MHz. Figure 4 graphically shows the FPGA occupation for the ANN accelerator implementations with most and least reuse, i.e., 39200x500 and 392x10. As the error rate is linearly dependent on the resource usage, the area differences will directly impact the system error rate, as discussed in Section IV.

From Table II, we can clearly find the area-timing trade-off, where a configuration using many hardware resources needs fewer clock cycles to complete an inference compared with a configuration using less hardware, but taking longer. Note that the last configuration (i.e., 392x10) is an exception. In fact, in the HLS process, Vitis HLS could not satisfy the specified time constraint, hence impacting the loop pipelining. In particular, configuration 392x10 should take roughly half as many clock cycles as configuration 784x25 - since the R_1 is halved and R_2 is 2.5 times lower. However, Vitis HLS could not meet the imposed timing constraints and did not manage to correctly parallelize the loop of the first fully connected layer, ending up requiring two clock cycles per iteration. As a result, configuration 392x10 requires a similar number of cycles as configuration 784x25, despite the difference in hardware resources. Even if larger configurations are possible, further decreasing the reuse parameters would accentuate the performance gain problem and lead to a higher or equal number of cycles compared to the 784x25 configuration. Therefore, we do not explore other configurations with smaller values of reuse parameters. We also confirmed this issue by testing the configuration 196x100, for which Vitis HLS created a design taking four clock cycles per iteration under the same constraints.

Table II shows a fluctuating trend of BRAM consumption. We believe this trend is related to the heuristic synthesis choices made by the tool, using either LUTRAM or BRAM as memory elements to store the weights in different percentages. Finally, for the DSP, we notice that the HLS process generates implementations using as many DSPs as the number of parallel multiplications in the fully connected layers plus 10 DSPs for the softmax layer. For example, configuration 3920x100 performs i) 10 multiplications in parallel for the first fully connected layer, each one reusing 3920 (i.e., R_1) times the same DSP, ii) 5 multiplications in parallel for the second fully connected layer, each one reusing 100 (i.e., R_2) times the same DSP, and iii) 10 DSPs for the softmax layer, adding up to 25 total DSPs.

B. Beam Experiments

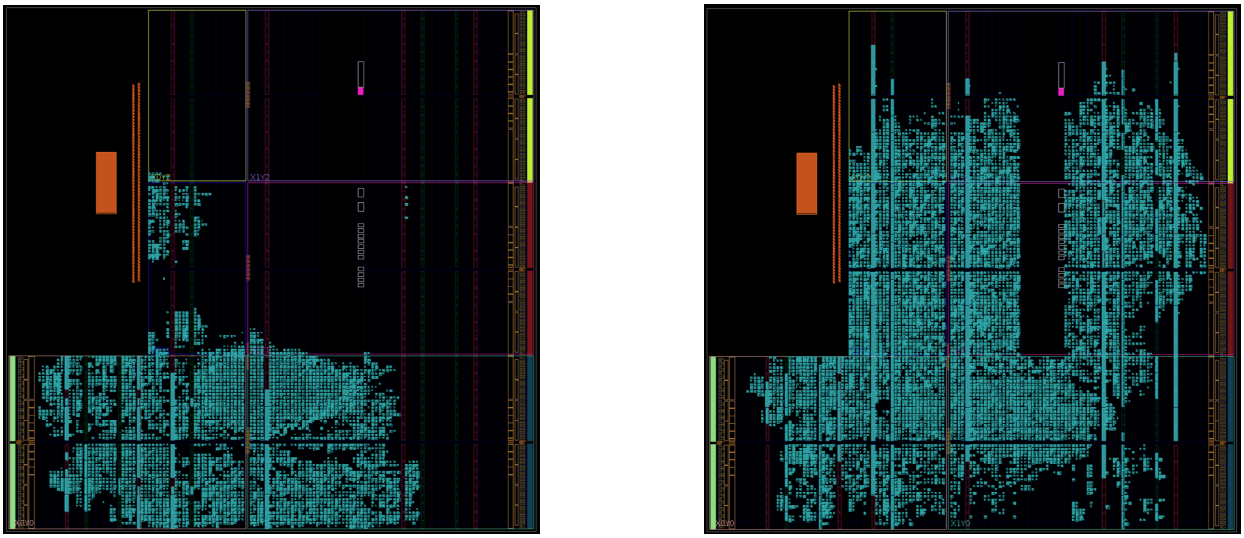
We perform our experiments at the ChipIR facility of the Rutherford Appleton Laboratory, UK. Figure 5 shows the setup mounted in the ChipIR facility. ChipIR facility delivers a beam of neutrons with a spectrum of energies similar to the atmospheric neutron one [31]. The available neutron flux is about $3.5 \times 10^6 n/(cm^2/s)$, $\sim$8 orders of magnitude higher than the terrestrial flux (13 $neutrons/(cm^2 \cdot h)$ at sea level [32]).

Since the terrestrial neutron flux is low, it is improbable to see more than a single corruption during program execution in a realistic application. We carefully design the experiments to maintain this property (observed error rates were lower than 1 error per 2,000 executions). Each ANN accelerator is tested for at least 14 effective hours, not including the setup, result check, initialization, and recovery from the DUE time.

C. System Under Test

As Devices Under Test, we selected two TUL PYNQ-Z2 boards based on the 28nm Xilinx Zynq-7000 SoC (part number xc7z020clg400-1). It features a 650MHz dual-core ARM Cortex-A9 processor, 512MB of DDR3 memory, and the FPGA fabric. The FPGA encompasses 13,300 logic slices (each with four 6-input Look-Up Tables (LUTs) and eight flip-flops), 630 KB of fast BRAM (organized in 140 tiles), and 220 DSPs. It is worth noting that the Zynq SoC has native protections that the user can activate. However, we perform all the experiments without any protection in the ANNs accelerators. That is, all the faults in the configuration memories may propagate until a failure is generated.

We consider a system composed of a software program running on the host processor (ARM processor) of the SoC



(a) 39200x500 configuration (b) 392x10 configuration

Fig. 4: FPGA occupation for the smallest (39200x500) and the largest (392x10) ANN accelerators we generate using HLS4ML.

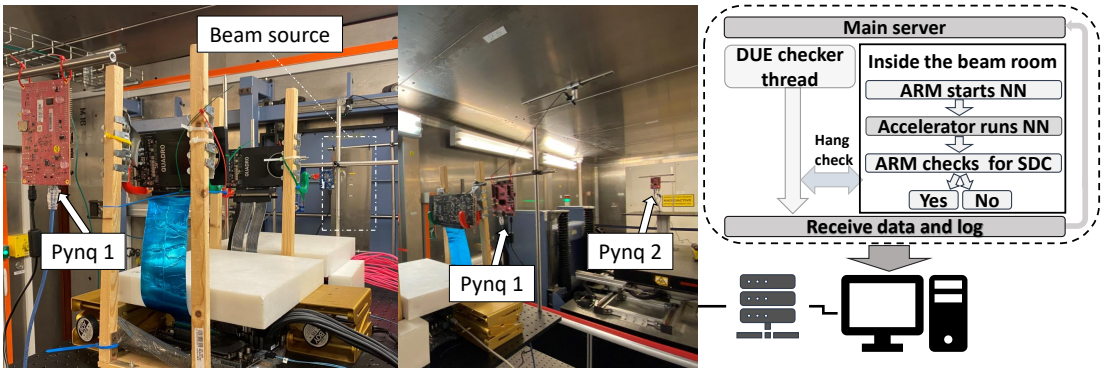


Fig. 5: Beam setup mounted at ChipIR beam line and the software setup. Even with the two boards being positioned in different positions, the error rates for the evaluated configurations show differences lower than 11.2% on average.

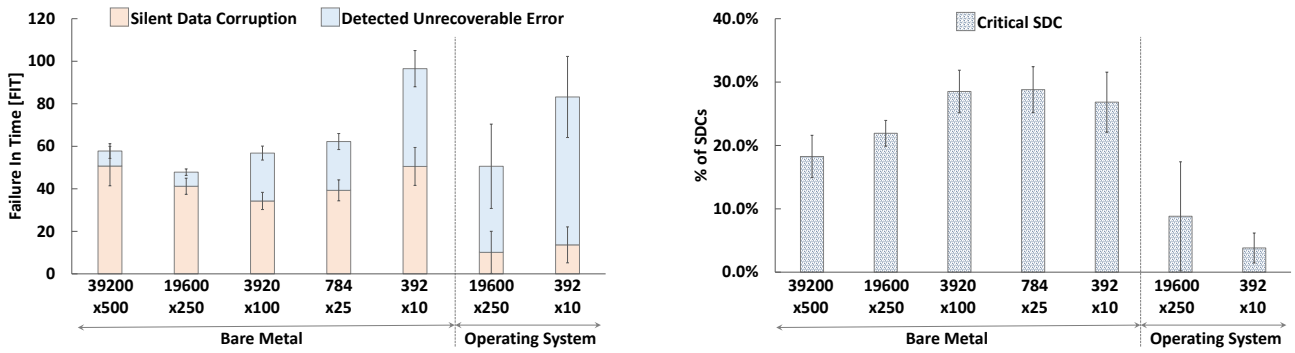
and the HLS-generated custom dataflow ANN accelerator deployed on the FPGA fabric. The experiments we perform can be divided into two categories: experiments where the host processor is running the software program on top of a Linux-based Operating System (OS), i.e., Petalinux, referred to as *OS-based*, and experiments where the software program is running without OS, to which we refer as *bare metal*.

As for the *OS-based*, input images are initially stored in the external SD memory as a binary file. After the OS boot, images are loaded into the main memory by the host processor and fed to the accelerator on the FPGA. For the *bare-metal*, input images are hard-coded directly in the program as constants. Then, for both *OS-based* and *bare-metal* configurations, the accelerator performs the inferences and returns the classification results to the host processor, as shown in Figure 5. Then, the host processor compares the results with the golden reference. The correctness of the golden reference is also checked with a signature to verify that a detected fault comes from the accelerator and not from a faulty golden reference. The host processor sends a periodic *heartbeat* message to the server to notify the correct status of the system. While in the *OS-based* configuration, the OS is responsible for managing the Ethernet communication, in the *bare-metal*, the software program includes a lightweight UDP protocol to enable communication with the server outside of the beam room. Whenever an error is detected, the host processor sends the faulty output to the server, asking for a hard reboot.

D. Error Detection and Error Rate Analysis

During the experiment, the host processor monitors the ANN output and compares it with the pre-computed fault-free golden copy. Whenever there is a mismatch, data is logged and the device re-programmed. That is, we do not use scrubbing but let faults accumulate until the error occurs. This strategy allows us to focus on the raw sensitivity of the different implementations, possibly providing also interesting information on how to tune the scrubbing frequency in a realistic application.

The experiments we perform at ChipIR allow the estimation of the *Failure In Time (FIT)* rates, which represent the error rate of a given accelerator. The FIT rate is directly derived from the application cross-section (σ), which is obtained



(a) FIT rate

(b) Critical SDCs

Fig. 6: Experimentally measured Silent Data Corruption (SDC) and Detected Unrecoverable Error (DUE) Failure In Time (FIT) rates for the Bare Metal and Operating System-based configurations.

by dividing the number of observed errors by the received particle fluence ($neutrons/cm^2$). The cross-section, then, is multiplied by the $flux$ under which the system will operate, yielding the FIT rate. Although the FIT rate is an interesting metric to estimate the failure rates of a system, it does not account for the execution time that different accelerator implementations can have due to the HLS configuration parameters. In order to also consider how many correct executions are going to be performed before experiencing a failure, we can compute the *Mean Executions Between Failures (MEBF)*, which is given by equation 1.

$$MEBF = \frac{MTBF}{ExecutionTime} \tag{1}$$

Where the $MTBF$ is the inverse of the cross-section multiplied by the $flux$ under which the device will operate (i.e., $\frac{1}{\sigma * flux}$), and the $ExecutionTime$ is the application execution time.

It is essential to recall that, when it comes to ANNs, not all SDCs are critical. Therefore, relying solely on the global $MEBF_{SDC+DUE}$ by considering all the observed failures may underestimate the number of correct classifications made by the ANN. To account for this, we also calculate the $MEBF_{CriticalSDC+DUE}$ that only considers the failures that disturb the system execution somehow, i.e., Critical SDCs and DUEs (details at Section II-A).

IV. ANN Accelerators Error Rate

In this section, we discuss the outcomes of radiation experiments we conducted on the HLS-generated ANN hardware accelerators. It is worth noting that both boards produce almost identical results, with a mere 11.2% difference on average across most configurations. However, for tests on the 392x10 design, one of the boards stopped responding and was removed from the beam line. Therefore, we only report and analyze the findings of the PYNQ board that remained functional during all configurations.

A. FIT rate

Figure 6a shows the accelerators' SDC and DUE FIT rates while running Bare Metal and with the Operating System. The neutron beam significantly impacts the OS-based experiments, resulting in neutron-induced faults that prevent the OS from booting. As a result, the accelerators cannot pass the initialization phase, causing the DUE to increase by more than an order of magnitude, making it unfeasible to gather as many data as for the same accelerators tested with Bare Metal. Consequently, due to beam time limitations, we choose two configurations, 19600x250 and 392x10, to demonstrate the OS's impact on the FIT rates of the accelerators. The data in Figure 6a is presented with a 95% confidence interval.

1) DUE analysis: for the bare metal configurations, the SDC rate is always higher than the DUE rate (3.55× higher on average). The DUE FIT rates are 7.05, 6.64, 22.53, 22.95, and 45.99 for the 39200x500, 19600x250, 3920x100, 784x25, and 392x10 accelerators, respectively. We note that the DUE FIT rate increases with the accelerator area. This is justified since the larger the accelerator area (i.e., lower values of R_1 and R_2 reuse parameters), the higher the chance that a bit flip corrupts a critical circuit resource, which leads to hang or crash of the application. In particular, the special case of configuration 392x10 leads to a higher DUE rate. This is probably due to the accelerator having a large area exposed longer to the radiation; in fact, it uses more hardware resources than 784x25 configuration for a similar number of cycles. When evaluating the accelerators using the OS, the DUEs are much more probable than SDCs. The DUE FIT rate for the 19600x250 configuration is 40.47, and for the 392x10 is 69.56. The observed DUEs predominantly stem from exceptions occurring in the operating system running on the ARM processor. Since the processor performs background OS tasks that are not directly related to the ANN execution, a large part of SoC resources are consumed by the OS. Moreover, the entire SoC, including the ARM processor and its caches, is exposed to the beam. Hence, a substantial portion of DUEs occurs prior to any NN inference — precisely, 58% of the DUEs are attributed to OS-related factors. That is, most DUE events occurring during the boot or the configuration of the FPGA accelerator prevent the occurrence of SDCs. This circumstance does not occur when employing the bare-metal configuration. Additionally, prior works have demonstrated that the operating system can increase the DUE rate by one order of magnitude compared

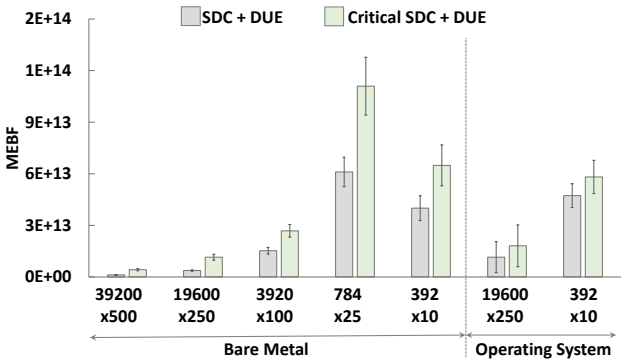


Fig. 7: Mean Executions Between Failures (MEBF) for both bare metal and operating systems configurations. The data includes the sum of DUEs with all the SDCs (Tolerable and Critical) or just the Critical SDCs.

with bare-metal configurations on SoCs that have ARM and FPGAs devices [33], [34].

2) SDC analysis: on bare metal configurations, the reuse parameters and the FPGA resource usage directly influence the SDC rate. The accelerators' SDC FIT rates are 50.70, 41.20, 34.28, 39.28, and 50.51 for the 39200x500, 19600x250, 3920x100, 784x25, and 392x10 accelerators, respectively. It is worth noting that the smaller accelerators (39200x500 and 19600x250) have higher or equal SDC rates compared to the larger ones (3920x100, 784x25, and 392x10). It is worth highlighting that no built-in fault detection functionalities, such as scrubbing and/or the activation ECC, were utilized in any of the experiments. As a result, smaller accelerators running for longer might accumulate errors in the extensively reused hardware and - since they experience less DUE stopping the execution - faults can propagate to outputs leading to more SDCs. As for OS-based configurations, they have much higher DUE rates, which prevents faults in the accelerator from propagating to the output. This leads to a much lower SDC rate for the OS-based configurations, with SDF FIT rates of 10.12 and 13.64 for the 39200x500 and 19600x250 configurations, respectively.

Moreover, as shown in Figure 6b, when comparing *Tolerable* and *Critical* SDCs, it is noticeable that the Critical SDCs are more frequent with bare metal configurations than with OS-based configurations. On average, the Critical SDC percentage is 24.87% for bare metal configurations and 6.32% for OS configurations. As discussed, this is due to the very high DUE rate for OS-based configurations. For bare metal configurations, data shows that the criticality of SDCs is related to the usage of hardware resources. Indeed, large accelerators experience a higher percentage of critical SDCs compared to small ones. Our insight is that, since the amount of work done is constant and the larger area is instantiated to improve performances (i.e., there is not a waste of resources), in the case of larger accelerators, there is indeed a higher probability that the accelerator is performing a critical computation at any given time. To illustrate this concept, let us use a hypothetical scenario where 10% of computation of a layer is critical. The criticality is the operation that, once corrupted, leads to a failure and, at this level of abstraction can be considered as a characteristic of the software. Note that, as a large accelerator uses more resources, it can execute more operations in parallel compared to a small accelerator. Assuming that the large accelerator is able to process 10% of the entire layer during a clock cycle, it will execute on average one critical computation per cycle. On the contrary, the small accelerator executes less operations within a cycle, due to fewer resources. Assuming that the small accelerator processes 1% of the layer within a clock cycle, it will have several clock cycles where no critical computation takes place.

B. Mean Executions Between Failures

When comparing the ANN accelerators, relying on the FIT rate alone is insufficient, as it does not provide information on the application's execution time [35], [36]. Therefore, we need to correlate error rates with performance to make a more holistic comparison. To achieve that, we evaluate the Mean Executions Between Failures (MEBF) rate for each configuration of the tested accelerator (as explained in Section III).

Figure 7 shows on the y-axis the MEBF for the different configurations we evaluate, which are given on the x-axis. For each ANN accelerator, the MEBF values are provided, taking into account both SDC and DUE failures and taking into account only Critical SDCs and DUEs. Note that, analyzing these two MEBF values separately can help identify failures that will actually affect negatively the ANN accelerator utilization (i.e., crash or wrong classification) if no fault mitigation is adopted. On average, the MEBF for Critical SDCs + DUEs is 2.36× and 1.40× higher than the MEBF for SDCs + DUEs for bare metal and OS configurations, respectively. As it is easy to observe, this result can be attributed to the infrequency of Critical SDCs when compared to non-critical SDCs. Specifically, Critical SDCs only occur when a fault impacts the classification capability of the NN, i.e., the top-1 accuracy is impacted.

The MEBF shows a linear correlation with the reuse parameters. The SDC + DUE MEBF increases for bare metal configurations from 1.19×10^{12} (39200x500) to 6.11×10^{13} (784x25). When we compare the Critical SDC + DUE for the 784x25 accelerator, it has a MEBF that is 4.23×10^{12}, which is 15.35× higher than the 39200x500. For OS-based configurations, the SDC + DUE MEBF are 1.15×10^{13} and 4.73×10^{13} for 19600x250 and 392x10, respectively. Like bare metal configurations, the Critical SDC + DUE MEBF is higher than the MEBF that considers all SDCs. The Critical SDC + DUE MEBF are 1.81×10^{13} and 5.82×10^{13} for the 19600x250 and 392x10 configurations, respectively.

For both MEBF, SDC+DUE and Critical SDC+DUE, the bare metal 392x10 configuration shows a lower MEBF than the 784x25. This is not surprising as the hardware generated by 392x10 is bigger than that of 784x25 and provides similar performance. As discussed in Section II, reducing the reuse parameters for the evaluated ANN beyond 784x25 did not improve performance since the Vitis HLS could not meet the time constraints that affect the loop pipeline of the 392x10 accelerator.

All in all, data shows that fast and large accelerators (e.g., 784x25) manage to perform more inferences between two failures w.r.t. slow and small ones (e.g., 39200x500). However, when they fail, it is more probable that they experience a critical SDC.

C. Mitigation Strategies and Future Directions

The FIT rate results indicate that adjusting the HLS reuse parameters can significantly enhance the Critical SDC + DUE MEBF more than an order of magnitude (15× higher MEBF) through a primarily software-based approach. We assess various configurations without utilizing the built-in fault detection features provided by the board vendor, such as scrubbing and Error Correction Codes (ECC), which are known to considerably reduce SDC and DUE rates on FPGAs, as demonstrated in previous studies [37], [38]. Consequently, in future evaluations, it is essential to conduct a hardware-oriented assessment that considers FPGA detection and correction techniques and how much they will affect the HLS reliability and performance.

The HLS tools simplify the implementation phases and provide an almost hardware-agnostic implementation of the ML models, making it easier for ML engineers to benefit from highly efficient and low-power FPGAs. Evidently, our work focuses on using ANN accelerators for image classification. Still, as ML models become more complex, it's essential to consider the impact of HLS parameters on reliability. For resource-demanding models such as Transformers and Deformable Convolutions, a naive configuration of a fault tolerance technique like scrubbing may lead to unacceptable performance losses. For instance, a high-rate scrubbing on a large transformer accelerator may add too much overhead to the system. However, our data shows that - by acting on HLS parameters - the MEBF can be increased, thus making it possible to decrease the scrubbing frequency, ultimately reducing both downtime and power consumption.

V. Conclusions

We examined how HLS tools affect the reliability of NN accelerators generated for FPGAs under high-energy neutrons running on Xilinx PYNQ boards. We test five versions of the same NN algorithm using HLS and measure their ability to correctly classify under a neutron beam. Our findings indicate that SDCs that are tolerable occur more frequently than critical ones for all the accelerators. Additionally, the results show that larger hardware configurations have a higher percentage of Critical SDCs but also higher MEBF values for bare metal setups. Furthermore, in OS-based setups, the high DUE rate masks the fault propagation until the output of the NN accelerator, leading to lower SDC and critical SDC FIT rates.

Acknowledgment

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 899546 with the support of the Brittany Region and under the Rad-Next grant agreement No 101008126 [39]. This project is partially funded by ANR FASY (ANR-21-CE25-0008-01) and ANR RE-TRUSTING (ANR-21-CE24-0015-02). ChipIR and RADNEXT provided and supported neutron beam time experiments (DOI https://doi.org/10.5286/ISIS.E.RB2310472). We acknowledge the researchers Dr. Christopher Frost and Dr. Maria Kastriotou, who helped with neutron experiments, and Joseph Paturel for the fruitful technical discussions.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 22, no. 3, pp. 436–44, May 2015.

[2] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "14.5 Envision: A 0.26-to-10TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable Convolutional Neural Network processor in 28nm FDSOI," in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, February 2017, pp. 246–247.

[3] Y.-H. Chen, J. Emer, and V. Sze, "Using Dataflow to Optimize Energy Efficiency of Deep Neural Network Accelerators," *IEEE Micro*, vol. 37, no. 3, pp. 12–21, June 2017.

[4] S. Rokicki, D. Pala, J. Paturel, and O. Sentieys, "What You Simulate Is What You Synthesize: Designing a Processor Core from C++ Specifications," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, November 2019, pp. 112–120.

[5] G. Li, S. K. S. Hari, M. Sullivan, T. Tsai, K. Pattabiraman, J. Emer, and S. W. Keckler, "Understanding Error Propagation in Deep Learning Neural Network (DNN) Accelerators and Applications," in *SC17: International Conference for High Performance Computing, Networking, Storage and Analysis*, November 2017, pp. 84–96.

[6] A. Ruospo, A. Balaara, A. Bosio, and E. Sanchez, "A pipelined multi-level fault injector for deep neural networks," in *2020 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, October 2020, pp. 13–19.

[7] A. Ruospo, E. Sanchez, M. Traiola, I. O'Connor, and A. Bosio, "Investigating data representation for efficient and reliable Convolutional Neural Networks," *Microprocessors and Microsystems*, vol. 86, p. 104318, October 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0141933121004786

[8] M. Traiola, A. Kritikakou, and O. Sentieys, "A machine-learning-guided framework for fault-tolerant DNNs," in *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. Antwerp, Belgium: IEEE, April 2023.

[9] ——, "harDNNing: a machine-learning-based framework for fault tolerance assessment and protection of DNNs," in *2023 IEEE European Test Symposium (ETS)*. Venezia, Italy: IEEE, May 2023.

[10] F. F. d. Santos, P. F. Pimenta, C. Lunardi, L. Draghetti, L. Carro, D. Kaeli, and P. Rech, "Analyzing and Increasing the Reliability of Convolutional Neural Networks on GPUs," *IEEE Transactions on Reliability*, vol. 68, no. 2, pp. 663–677, November 2019.

[11] J. Gava, A. Hanneman, G. Abich, R. Garibotti, S. Cuenca-Asensi, R. P. Bastos, R. Reis, and L. Ost, "A lightweight mitigation technique for resource- constrained devices executing dnn inference models under neutron radiation," *IEEE Transactions on Nuclear Science*, vol. 70, no. 8, pp. 1625–1633, March 2023.

[12] R. M. Brewer, S. L. Moran, J. Cox, B. D. Sierawski, M. W. McCurdy, E. X. Zhang, S. S. Iyer, R. D. Schrimpf, M. L. Alles, and R. A. Reed, "The Impact of Proton-Induced Single Events on Image Classification in a Neuromorphic Computing Architecture," *IEEE Transactions on Nuclear Science*, vol. 67, no. 1, pp. 108–115, December 2020.

[13] S. Blower, P. Rech, C. Cazzaniga, M. Kastriotou, and C. D. Frost, "Evaluating and Mitigating Neutrons Effects on COTS EdgeAI Accelerators," *IEEE Transactions on Nuclear Science*, vol. 68, no. 8, pp. 1719–1726, June 2021.

[14] R. L. Rech Junior, S. Malde, C. Cazzaniga, M. Kastriotou, M. Letiche, C. Frost, and P. Rech, "High Energy and Thermal Neutron Sensitivity of Google Tensor Processing Units," *IEEE Transactions on Nuclear Science*, vol. 69, no. 3, pp. 567–575, January 2022.

[15] A. Ruospo, E. Sanchez, L. M. Luza, L. Dilillo, M. Traiola, and A. Bosio, "A Survey on Deep Learning Resilience Assessment Methodologies," *Computer*, vol. 56, no. 2, pp. 57–66, February 2023.

[16] F. Libano, P. Rech, B. Neuman, J. Leavitt, M. Wirthlin, and J. Brunhaver, "How Reduced Data Precision and Degree of Parallelism Impact the Reliability of Convolutional Neural Networks on FPGAs," *IEEE Transactions on Nuclear Science*, vol. 68, no. 5, pp. 865–872, January 2021.

[17] F. Libano, B. Wilson, J. 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, November 2019.

[18] M. G. Trindade, A. Coelho, C. Valadares, R. A. C. Viera, S. Rey, B. Cheymol, M. Baylac, R. Velazco, and R. P. Bastos, "Assessment of a Hardware-Implemented Machine Learning Technique Under Neutron Irradiation," *IEEE Transactions on Nuclear Science*, vol. 66, no. 7, pp. 1441–1448, June 2019.

[19] L. M. Luza, D. Söderström, G. Tsiligiannis, H. Puchner, C. Cazzaniga, E. Sanchez, A. Bosio, and L. Dilillo, "Investigating the impact of radiation-induced soft errors on the reliability of approximate computing systems," in *2020 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, October 2020, pp. 49–55.

[20] F. Benevenuti, M. Gonçalves, E. C. F. P. Junior, R. G. Vaz, O. L. Gonçalez, J. R. Azambuja, and F. L. Kastensmidt, "Neutron-induced Faults on CNN for Aerial Image Classification on SRAM-based FPGA Using Softcore GPU and HLS," in *2021 21th European Conference on Radiation and Its Effects on Components and Systems (RADECS)*, September 2021, pp. 57–61.

[21] H.-B. Wang, Y.-S. Wang, J.-H. Xiao, S.-L. Wang, and T.-J. Liang, "Impact of Single-Event Upsets on Convolutional Neural Networks in Xilinx Zynq FPGAs," *IEEE Transactions on Nuclear Science*, vol. 68, no. 4, pp. 394–401, February 2021.

[22] J. Tonfat, L. Tambara, A. Santos, and F. L. Kastensmidt, "Soft error susceptibility analysis methodology of HLS designs in SRAM-based FPGAs," *Microprocessors and Microsystems*, vol. 51, pp. 209–219, June 2017.

[23] L. A. Tambara, J. Tonfat, A. Santos, F. Lima Kastensmidt, N. H. Medina, N. Added, V. A. P. Aguiar, F. Aguirre, and M. A. G. Silveira, "Analyzing reliability and performance trade-offs of hls-based designs in sram-based fpgas under soft errors," *IEEE Transactions on Nuclear Science*, vol. 64, no. 2, pp. 874–881, January 2017.

[24] A. Dixit and A. Wood, "The impact of new technology on soft error rates," in *2011 International Reliability Physics Symposium*, April 2011, pp. 5B.4.1–5B.4.7.

[25] S. Rehman, M. Shafique, and J. Henkel, *Reliable Software for Unreliable Hardware: A Cross Layer Perspective*. Springer Publishing, April 2016.

[26] A. Lotfi, S. Hukerikar, K. Balasubramanian, P. Racunas, N. Saxena, R. Bramley, and Y. Huang, "Resiliency of automotive object detection networks on gpu architectures," in *2019 IEEE International Test Conference (ITC)*, February 2019, pp. 191–200.

[27] P. Coussy, D. D. Gajski, M. Meredith, and A. Takach, "An Introduction to High-Level Synthesis," *IEEE Design & Test of Computers*, vol. 26, no. 4, pp. 8–17, August 2009.

[28] FastML Team, "fastmachinelearning/hls4ml," 2021. [Online]. Available: https://github.com/fastmachinelearning/hls4ml

[29] J. Duarte, S. Han, P. Harris, S. Jindariani, E. Kreinar, B. Kreis, V. Loncar, J. Ngadiuba, M. Pierini, D. Rankin, R. Rivera, S. Summers, N. Tran, and Z. Wu, "Fast Inference of Deep Neural Networks for Real-Time Particle Physics Applications," *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 305–335, February 2019.

[30] L. Deng, "The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, October 2012.

[31] C. Cazzaniga *et al.*, "Progress of the scientific commissioning of a fast neutron beamline for chip irradiation," in *ICANS*, May 2017, pp. 159–164.

[32] C. Slayman, *JEDEC Standards on Measurement and Reporting of Alpha Particle and Terrestrial Cosmic Ray Induced Soft Errors*. JEDEC, September 2011, pp. 55–76.

[33] P. Bodmann, G. Papadimitriou, D. Gizopoulos, and P. Rech, "The Impact of SoC Integration and OS Deployment on the Reliability of Arm Processors," in *2021 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, April 2021, pp. 223–225.

[34] C. J. Corley, H. M. Quinn, and E. E. Swartzlander, "Accelerated Nuclear Radiation Effects on the Raspberry Pi 3B+," in *2022 IEEE Radiation Effects Data Workshop (REDW) (in conjunction with 2022 NSREC)*, October 2022, pp. 203–211.

[35] G. Reis, J. Chang, N. Vachharajani, and S. Mukherjee, "Design and evaluation of hybrid fault-detection systems," in *Proceedings of the 2005 International Symposium on Computer Architecture, ISCA'05*. IEEE Press, June 2005, pp. 148–159.

[36] P. Rech, L. Pilla, P. Navaux, and L. Carro, "Impact of GPUs Parallelism Management on Safety-Critical and HPC Applications Reliability," in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, September 2014, pp. 455–466.

[37] P. Maillard, Y. P. Chen, J. Arver, V. Merugu, A. Shui, and M. Voogel, "Neutron and 64MeV Proton Characterization of Xilinx 7nm VersalTM Multicore Scalar Processing System (PS)," in *2022 IEEE Radiation Effects Data Workshop (REDW) (in conjunction with 2022 NSREC)*, 2022, pp. 18–23.

[38] P. Maillard, Y. P. Chen, J. Vidmar, N. Fraser, G. Gambardella, M. Sawant, and M. L. Voogel, "Radiation-Tolerant Deep Learning Processor Unit (DPU)-Based Platform Using Xilinx 20-nm Kintex UltraScale FPGA," *IEEE Transactions on Nuclear Science*, vol. 70, no. 4, pp. 714–721, October 2023.

[39] R. G. Alía, A. Coronetti, K. Bilko, M. Cecchetto, G. Datzmann, S. Fiore, and S. Girard, "Heavy ion energy deposition and see intercomparison within the radnext irradiation facility network," *IEEE Transactions on Nuclear Science*, vol. 70, no. 8, pp. 1596–1605, March 2023.