Reliability Assessment of Deep Neural Networks and Accelerators Across Design Stages

Mohammad Hasan Ahmadilivani*, Natalia Cherezova*, Michael Glaß§, Juan-David Guerrero-Balaguera†, Maksim Jenihhin*, Angeliki Kritikakou‖, Robert Limas Sierra†, Leticia Bolzani Poehls¶, Jaan Raik*, Lucas Roquet‖, Fernando Fernandes dos Santos‖, Felipe Augusto da Silva‡, Matteo Sonza Reorda†, Alessandro Veronesi¶, Ernesto Cristopher Villegas Castillo§‡, Josie E. Rodriguez Condia†

**Tallinn University of Technology - Department of Computer Systems, Tallinn, Estonia

§Ulm University - Institute of Embedded Systems/Real-Time Systems, Ulm, Germany

†Politecnico di Torino - Department of Control and Computer Engineering (DAUIN), Turin, Italy

‖University of Rennes - INRIA, Rennes, France

¶Leibniz Institute for High Performance Microelectronics (IHP), Frankfurt Oder, Germany

‡Cadence Design Systems, Munich, Germany

***Abstract*—Artificial Intelligence (AI) is driving the introduction of advanced functionality and autonomy across several domains, including automotive, robotics, health care, and aerospace. Such advances are supported by the availability of algorithms and hardware accelerator platforms to efficiently implement them. However, transistor miniaturization raises reliability concerns during in-field operation and poses challenges in terms of computational intensity and performance for which conventional design and fault-characterization strategies are not well-suited. This work reviews three approaches for determining fault vulnerabilities and reliability challenges in hardware accelerators, devices, and software for Deep Neural Networks. The first strategy describes a cost-effective semi-analytical reliability assessment method for the design space exploration of reliability-performance trade-offs using reconfigurable systolic array engines. The second strategy analyzes and evaluates four abstraction levels for dot-product-unit-based accelerators and assesses the implications of selecting an affordable strategy based on the analysis needs. The third approach analyzes the critical role of correctly selecting representative inputs in large Vision Transformers (ViTs) for their effective reliability evaluation.**

***Index Terms*—Colored Petri Nets, Fault Injection, Hardware Accelerators, Reliability, Special Function Units, Tensor Cores.**

I. Introduction

The disruptive adoption of AI is supported by advanced and effective algorithms, as well as the availability of sophisticated platforms for its efficient deployment, including AI hardware based on *Systolic Arrays* (SAs), *Graphic Processing Units* (GPUs), and *Dot Product Unit* (DPU) architectures [1]. In fact, AI plays a crucial role in several application fields, such as robotics, autonomous machines, self-driving cars, and, more recently, in *Internet of Things* as *Artificial Intelligence of Things* or AIoT [2], [3].

In safety-critical environments, the strict reliability requirements make the analysis and assessment of non-functional properties both mandatory and essential throughout the development and deployment of AI-powered applications, such as *Deep Neural Networks* (DNNs), as mandated by industrial standards (e.g., ISO 26262). Unfortunately, the use of conventional fault assessment and evaluation strategies raises several technical challenges. In particular, transistor miniaturization enables the integration of increasingly complex functionalities into hardware while maintaining acceptable energy consumption. However, the high transistor density, combined with the data-intensive nature of AI-powered workloads, demands substantial computational power and large assessment times. Thus, the identification, exploration, and integration of innovative methods for assessing hardware accelerators and DNNs at the design and deployment stages are crucial to improve performance while preserving characterization accuracy.

This work explores 3 cutting-edge approaches to uncover the fault vulnerabilities and reliability challenges of hardware accelerators and software powering today's AI systems.

Section II presents a reliability assessment for the design space exploration of reliability-performance trade-offs in CNNs deployed on Reconfigurable Systolic Array (RSA) accelerators. With a comprehensive reliability and performance evaluation, the section proposes a reconfiguration strategy for a layer-wise reconfiguration of RSAs to balance the reliability and performance of an inference.

Section III analyses the effects on fault characterization accuracy and performance trade-offs across four fault assessment strategies applied on DPU-based AI accelerators for two number formats FP and Posit: *i)* low-level micro-architectural logic simulation, *ii)* emulation, *iii)* structural/architectural simulation, and through *iv)* Colored Generalized Stochastic Petri Nets (CGSPNs). The evaluation of several GEMM tiles on a DPU-based array engine indicates that fault effects vary according to the accelerator's shape and the hardware reuse for computing GEMMs. Furthermore, each fault assessment strategy plays a crucial role and might support different design stages. Our results indicate that structural and CGSPN strategies may be adapted for estimations in early designs, while emulation and logic simulation strategies are more suitable when highly accurate results are required.

Finally, Section IV analyzes the scalability limitations of

979-8-3195-4235-9/26/$31.00 ©2026 IEEE

conventional radiation testing and fault simulation for reliability assessment, and their implications for input selection in large Vision Transformers (ViTs). Existing approaches typically rely on small, randomly sampled input sets. However, due to the dense nature of ViTs, such random sampling often fails to represent the full input space and can substantially underestimate failure rates. The experimental results show that input selection plays a critical role in reliability assessment for image-classification ViTs. In particular, confidence-aware input selection strategies can cause failure rates to vary by up to an order of magnitude in both neutron-beam experiments and software fault injection. Moreover, we show a reliability analysis with software fault simulation, which reveals a strong correlation between input sensitivity and model confidence, underscoring the need for more representative input selection methodologies in ViT reliability evaluation.

II. Reliability-Aware Reconfiguration for Reconfigurable Systolic Arrays

Systolic Arrays (SAs) are widely adopted for accelerating the inference of CNNs [4]. Reconfigurable SAs (RSAs) are presented to optimize the performance and energy consumption of the systems [5]–[7]. However, the reliability of reconfigurable SAs has not been previously studied in the literature [8], [9]. This section studies CNNs deployed on layer-wise reconfigurable SAs. The objective is to evaluate the reliability of CNNs' layers against a single fault in RSA registers, through a design space exploration over different SA sizes, along with their performance and utilization. Furthermore, a resizing strategy is proposed that identifies the best SA size for each layer, considering both reliability and performance.

A. Methodology

1) The Baseline RSA Architecture: The architecture of the RSA considered in this work is presented in Figure 1. The RSA is based on the output-stationary dataflow, in which weights propagate through the SA from top to bottom, activations propagate from left to right, and partial sums (outputs) are kept stationary in the Processing Elements (PEs). Before the execution of a layer, a set of PEs is activated by the controller, and only they perform the operations of that layer of a CNN. The unused PEs are clock-gated to optimize energy consumption. The outputs of the layer are captured from the determined subset of PEs. In this work, the RSA consists of 256×256 PEs, and it can be reconfigured into four sizes: 32×32, 64×64, 128×128, and 256×256. Before an inference, the system's configuration for each layer is determined based on offline profiling, and the RSA is reconfigured at run-time with a negligible performance overhead.

2) Reliability and Performance Analysis for the RSA: To analyze the reliability of the RSA, we perform an extensive Fault Injection (FI) experiment and obtain the Architecture Vulnerability Factor (AVF) for each CNN layer on each SA configuration. AVF represents the probability that a fault in the hardware results in an application output error [10]. In the case of CNNs, AVF refers to the probability of altering the golden

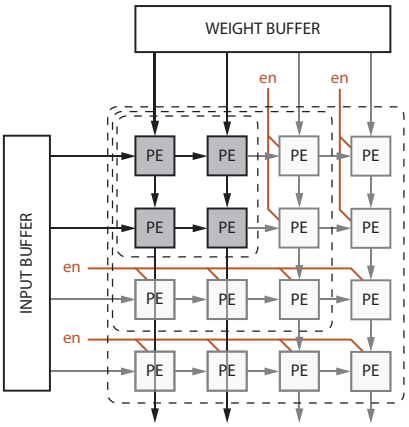


Fig. 1. Architecture of the reconfigurable systolic array

classification for an input by a fault. We consider single bitflips in the input, weight, and output registers of each active PE in each RSA configuration per inference. We perform FI for each register separately, and the AVF is the average of all FI campaigns.

The FI is performed using the reliability assessment method for CNN inference on systolic arrays based on fault propagation analysis presented in [11]. Instead of modeling a systolic array on the microarchitecture level and performing time-consuming cycle-accurate simulations, fault propagation analysis is used to calculate the resulting error in the layer output. This allows to noticeably speed up the reliability assessment. The FI is performed layer-wise based on single bitflips in the PE registers, with over 1000 injected faults per convolutional layer, ensuring 95% confidence and 5% error margin [12]. Faults are injected only in used PEs.

To analyze the performance, we profile the execution time (T) of each layer i on each SA configuration j as well as the utilization (U) of the PEs in each configuration by the CNN layer. To evaluate the performance, we consider the efficiency metric (E), as defined in Eq. (1).

$$E_{size_j}^{layer_i} = \frac{U_{size_j}^{layer_i}}{T_{size_j}^{layer_i}} \tag{1}$$

The efficiency metric combines the utilization and execution time of the SA configurations for one layer. A higher value means that an SA configuration can achieve a faster execution while using PEs as much as possible. To calculate efficiency, we normalize the execution time of all layers of a CNN to the longest execution time across all AS configurations. Also, the obtained E value is normalized to the maximum value of the layers of a CNN.

3) Resizing Strategies: To determine the RSA configuration, we first conduct an extensive layer-wise reliability and performance analysis of all aforementioned RSA configurations for each layer of the target CNNs. Based on the obtained AVF and FoM results, three strategies for layer-wise reconfiguration of the RSA are considered:

- *Constant Configuration*: We consider a constant configuration for the RSA based on the minimum (RSA32) and maximum (RSA256) available configurations, to demonstrate

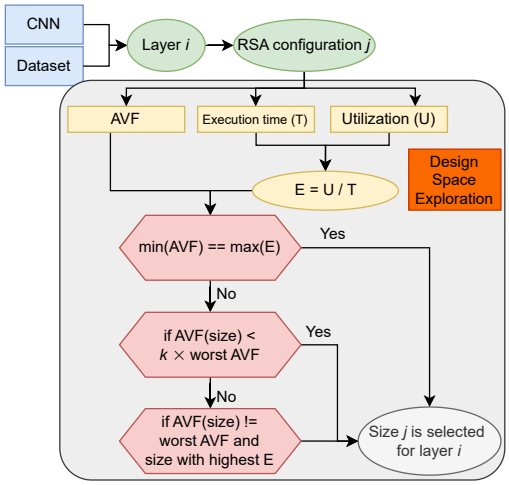


Fig. 2. Proposed algorithm to determine the RSA configuration for each layer by balancing reliability and performance.

TABLE I
SA SIZE SELECTED FOR EACH LAYER BY EACH STRATEGY FOR VGG-19

Layer	RSA32	RSA256	BR	BP	RP
1	32×32	256×256	32×32	64×64	128×128
2	32×32	256×256	64×64	64×64	64×64
3	32×32	256×256	32×32	128×128	128×128
4	32×32	256×256	128×128	128×128	128×128
5	32×32	256×256	256×256	256×256	256×256
6	32×32	256×256	256×256	256×256	256×256
7	32×32	256×256	128×128	256×256	256×256
8	32×32	256×256	32×32	256×256	256×256
9	32×32	256×256	32×32	256×256	256×256
10	32×32	256×256	32×32	256×256	256×256
11	32×32	256×256	32×32	256×256	128×128
12	32×32	256×256	64×64	256×256	256×256
13	32×32	256×256	32×32	256×256	128×128
14	32×32	256×256	32×32	256×256	256×256
15	32×32	256×256	256×256	256×256	256×256
16	32×32	256×256	64×64	256×256	256×256
Average	32×32	256×256	92×92	216×216	204×204

the advantages of the reconfigurability of RSA in terms of reliability and performance.

- *Best Reliability (BR)*: The RSA size for each layer is determined by the configuration, which implies minimum AVF.
- *Best Performance (BP)*: The RSA size for each layer is determined by the configuration, yielding the maximum performance efficiency according to Eq. (1).
- *Reliability-Performance Balance (RP)*: The RSA size for each layer is determined based on the proposed algorithm in Figure 2, to balance the reliability and performance of the SA. In this algorithm, the AVF and efficiency E for all configurations of the RSA for each layer are explored, and the RSA size for a layer is determined, which: 1) offers the best AVF and E, or 2) holds an AVF considerably lower than other sizes, or 3) contains the highest E as long as it does not carry the worst AVF.

In the experiments, we consider the $k = 0.6$ in the *RP* strategy, as a result of an exploration of the obtained AVF values for different layers and RSA configurations.

B. Experimental Results

In this subsection, we present the experimental results. First, we showcase a layer-wise comparison of reliability and performance between different RSA reconfiguration strategies for VGG-19. Afterward, we compare the obtained reliability and performance for three CNNs: VGG-19, ResNet-18, and ResNet-50, all trained on the ImageNet dataset. All networks use a 32-bit floating-point data type.

Figure 3 presents a layer-wise comparison of reliability and performance between different RSA reconfiguration strategies for VGG-19, respectively. Table I shows the RSA size selected for each layer by different strategies. The following observations can be highlighted:

- Reconfiguring the RSA for each layer provides an opportunity to optimize the reliability or performance of the CNNs deployment on systolic arrays. Comparing the results between the constant configurations (*RSA32* and *RSA256*) and the reconfigured configurations (*BR*, *BP*, and *PR*) shows gains in either reliability or performance.
- Based on the reconfiguration strategy, the RSA can provide the best reliability (*BR* strategy) or performance (*BP* strategy) in all layers. As observed, the AVF of all layers in the *BR* strategy is the minimum; however, their efficiency E is very low in many cases. Instead, in the *BP* strategy, efficiency is maximized across all layers; however, the AVF is high in many cases. This observation underscores the need to present the *RP* strategy.
- The proposed *RP* strategy yields configurations in which both reliability and performance are balanced. As observed, the AVF of *RP* in all layers is always equal to or less than that in the *BP* strategy, and the efficiency E in all layers is always equal to or more than that in the *BR* strategy.
- In terms of RSA size, it is observed that the *RP* strategy utilizes an average SA size of 204×204 PEs to balance the inference reliability and performance, which is smaller than that of *BP* and larger than that of *BR*.

Furthermore, we present the average results of AVF and normalized FoM for each CNN model to compare the impact of strategies. Figure 4 presents the results for average AVF and efficiency E, respectively, for three CNN models deployed on RSA with the five reconfiguration strategies. As observed, layer-wise reconfiguration strategies can effectively lead to a more reliable or performant deployment. On the other hand, in all cases, the *RP* strategy achieves a lower AVF (5.5% to 8.3% more reliable) than the *BP* strategy and higher performance efficiency than the *BR* strategy ($1.6\times$ up to $14.7\times$ more performant). These results highlight that *RP* can successfully balance reliability-performance in the CNN deployment on RSAs.

III. Fault Assessment of DPU-based AI Hardware: Accuracy and Performance Trade-offs in FP and Posit Formats

This section analyzes and evaluates the advantages and constraints of four strategies for characterizing fault-rate accuracy and evaluating the performance of DPU-based array accelerators for *Floating Point* and *Posit* number formats. In particular,

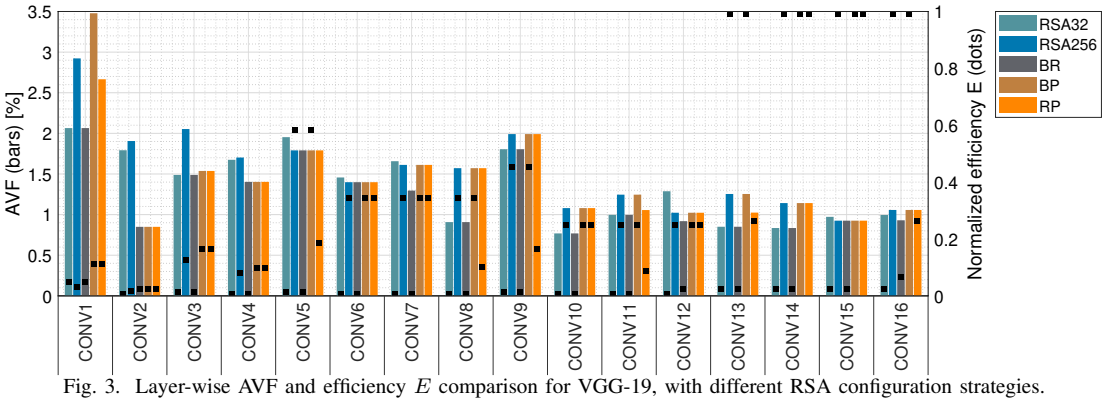


Fig. 3. Layer-wise AVF and efficiency E comparison for VGG-19, with different RSA configuration strategies.

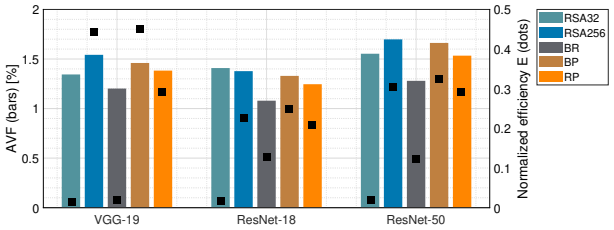


Fig. 4. Model-wise AVF and efficiency E comparison with different RSA configuration strategies.

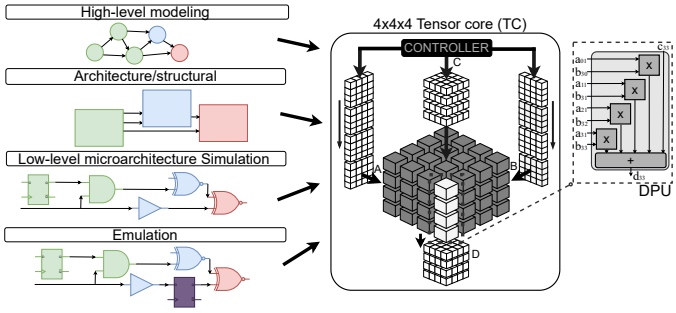


Fig. 5. A general scheme of the multi-abstraction analysis of DPU-based accelerators.

DPU-based-array cores are crucial elements commonly found in several platforms to train and deploy AI-based workloads, including *Tensor cores* (TCs) in GPUs [13], engines in NPUs [14], matrix core extensions in CPUs [15], [16], and on-chip accelerators in automotive devices [17]. Each implementation varies according to program flexibility targets, performance, and power budgets. In practice, some TCs can be configured for size and number format via software interfaces. Instead, other versions are fixed in hardware to improve performance.

In this work, we analyze the effects on fault characterization accuracy and performance trade-offs across four fault assessment strategies applied on DPU-based AI accelerators for two number formats, FP and Posit: *i)* low-level micro-architectural logic simulation, *ii)* emulation, *iii)* structural/architectural simulation, and through *iv)* Colored Generalized Stochastic Petri Nets (CGSPNs), as depicted in Figure 5.

Each abstraction level offers advantages and constraints for experimentally-based reliability analysis of soft-error effects in hardware accelerators. According to the main targets, several studies indicate that some strategies are more effective for guiding the development of fault countermeasures and hardening strategies [8], [18]–[21]. However, evaluating masking and propagation effects, as well as identifying vulnerable structures, might involve multiple evaluation strategies.

A. Main idea

We use four analysis and evaluation strategies to characterize fault-propagation impacts and, if possible, identify the most vulnerable structures in DPU-based accelerators (TCs) for two formats. In particular, we analyze the trade-offs among effectiveness, efficiency, and accuracy, as well as the challenges in identifying fault-vulnerability sources. For the evaluations, we employ equivalent DPU-based array accelerators (TCs) descriptions to assess the impact of soft errors on their computing part. In more detail, our experiments target the most susceptible structures to soft errors (*Single Event Upset* or SEUs) within the computing components of DPUs in TCs (i.e., pipeline registers and memory buffers).

B. Abstraction strategies

1) ***Simulation-based evaluation****:* relies on logic simulators and instrumentation tools that systematically modify a circuit's timing functionality to represent SEU effects. The crucial advantage of this abstraction is the high controllability for injecting faults into any low-level microarchitectural structure of the targeted circuit. Moreover, the extended observability enables focused fault-propagation analyses (i.e., internal-operation observability) and the direct identification of fault-susceptible structures. Our evaluation uses industrial-grade frameworks (Z01X by *Synopsis*) to analyze and determine the vulnerable structures in the TCs.

2) ***Emulation-based evaluation****:* uses SHADOWFI [22] to automatically integrate and instrument fault-injection structures into the gate-level descriptions of the TCs. SHADOWFI is an open-source framework that conceptualizes fault evaluation as a distributed computing task on hyperscale systems. In particular, SHADOWFI decomposes a fault-injection task into sub-tasks, which are distributed and executed across multiple computational nodes for simulation or emulation, as depicted in Figure 6. The simulation workflow distributes fault-injection tasks as multiple logic simulations on several nodes/clusters

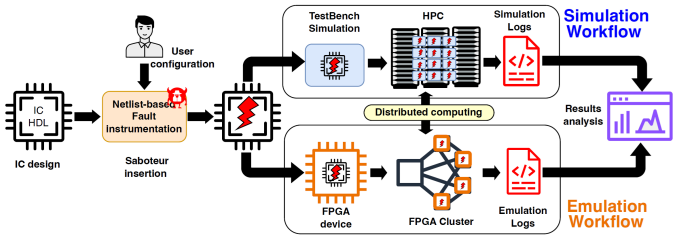


Fig. 6. General fault-injection workflow in SHADOWFI. [22]

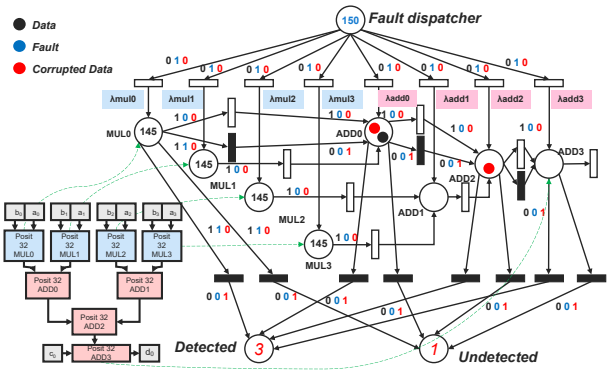


Fig. 7. CGSPN of the TC Tree Topology

within a high-performance computing system. Instead, the emulation workflow leverages FPGA reconfigurability to execute multiple fault injections across FPGA clusters. SHADOWFI employs a fault-emulation strategy via netlist instrumentation using sabotuer circuits to make the fault characterization scalable across simulation and emulation workflows. The fault instrumentation first converts an input RTL into a netlist using generic cells via Yosys, and then inserts the saboteur circuits into the target components within the design. Finally, fault impacts are analyzed after evaluating the outputs from the FIs.

*3) **Structural evaluation**:* employs the EMBER framework [23], a C++-based, cycle-accurate RTL simulator that enables reliability assessment from the early design stages. EMBER combines a cycle-driven simulation kernel with RT-level accuracy, allowing efficient fault characterization. Its runtime parameter initialization supports rapid hardware configuration exploration without requiring recompilation. Additionally, EMBER provides built-in, extensible saboteur instrumentation for modeling single-event upsets (SEUs) at the circuit level.

In this work, the TCs are modeled at cycle-accurate RTL using a modular abstraction of their fundamental components. These include DPU arrays (composed of adders and multipliers), memory elements (buffers and accumulators), and support for two numerical formats: Posit32 and FP32. Each TC also generates execution traces that record cycle counts, memory traffic, and operand reuse patterns to facilitate reliability analysis. The transient faults are modeled as single random bit flips injected into operand buffers or accumulators during a *General Matrix Multiplication* (GEMM) tile execution. The Injection times and locations are uniformly sampled, and each fault-injection campaign introduces one upset per tile execution.

*4) **High-level modeling**:* relies on the CGSPN modeling strategy [24]–[27], which has shown promising results for the early analysis and evaluation of functional safety and reliability features of System-on-Chip (SoC) architectures. This strategy abstracts hardware-level granularity into a high-level graph structure and enables system characterization and fault-propagation analysis via compacted fault-injection events. The graph parameters in the model are extracted from the individual micro-architecture characterization of the fundamental structures in a SoC (e.g., Adders and multipliers for DPUs in TCs). In this work, the analysis of transient faults used timing features from the fundamental components of a SoC (e.g., ADD or MUL cores) to build the model [28].

TABLE II
TC FEATURES ACCORDING TO THEIR ABSTRACTION

Abstraction	Format	Size	Sequential cells	Execution time (s)
Emulation	FP32	388,800 cells	24,256	0.9
	Posit32	639,680 cells	20,352	0.9
Simulation	FP32	263,984 cells	45,248	1.7
	Posit32	358,309 cells	20,352	1.7
Structural	FP32	53,248 bytes	-	0.05
	Posit32	53,248 bytes	-	0.05
High-level	FP32	128 nodes	-	$< 50 \times 10^{-6}$
	Posit32	128 nodes	-	$< 50 \times 10^{-6}$

The methodology consists of the following steps: 1) Component selection: Choose components to abstract as CGSPN nodes (see Figure 7). 2) Interconnection mapping: Model links between components as edges. 3) Structural characterization: Count gates, flip-flops, and determine each component's FIT rate. 4) SET characterization: Run a single-time fault-injection campaign per component. 5) CGSPN configuration: Set deterministic transitions using structural data and SET results. 6) AVF computation: Compute TC-level AVF via CGSPN simulation. 7) Validation: Compare estimated AVF with exhaustive TC fault-injection results.

C. Experimental Results

For the experiments, we evaluated the fault vulnerabilities of 4×4×4 DPU-based TCs in two formats (*FP32* and *Posit32*) across the four abstraction levels. In detail, the description of the DPUs at RTL comprises 12-stage pipelines for the software- and emulation-based evaluations. For simulation and emulation experiments, the TCs were synthesized using the 15nm Nangate Open Cell Library. Instead, the architectural and high-level models are extracted from characterizing the DPU's fundamental components (3-stage pipeline ADD and MUL cores). The structural evaluation on EMBER [23] is based on the main operative data-path structures extracted from a typical TC core implementation on PyOpenTCU [29]. Furthermore, the CGSPN models used the functional structural organization of the TCs and extracted information from focused software-based evaluations to tune the representation parameters [26]. As inputs for each characterization, we used representative GEMM operations in ranges: ± 1.0 and ± 10.0.

In general, each TC description preserves the intended functionality of the accelerator. However, the overall size and execution performance differ among them, as depicted

TABLE III
AVF AND ESTIMATED AVF FOR THE FOUR EVALUATION ABSTRACTIONS ON 4X4X4 TCS

Evaluation abstraction	Number format	TC structures	Injected faults	AVF (%)
Simulation	Posit32	Overall	4,160	75.77 ± 2.3
		ADD	1,869	89.08
		MUL	2,227	66.77
	FP32	Overall	4,161	64.36 ± 2.3
		ADD	2,412	55.30
		MUL	1,748	76.31
Emulation (ShadowFI)	Posit32	Overall	162,816	26.5 ± 1.5
		ADD	73,728	29.2
		MUL	89,088	24.2
	FP32	Overall	194,048	5.3 ± 1.5
		ADD	100,864	5.2
		MUL	85,184	5.4
Structural (Ember)	Posit32	Overall	9,393	88.0 ± 3.0
	FP32	Overall	9,393	87.0 ± 8.0
High-level (CGSPN)	Posit32	Overall	150	34.0 ± 5.0
	FP32	Overall	150	32.7 ± 5.0

Fig. 8. Error distribution occurrences for the FP (*red*) and Posit (*blue*) Tensor Cores for the ± 1 input range.

in Table II. To evaluate the potential impacts on vulnerability analysis, we conducted several fault injection campaigns at each abstraction level to characterize overall impacts using the *Architectural Vulnerability Factor* (AVF). In more detail, Table III reports the AVF and their estimates for each abstraction, along with the AVF per TC's fundamental component, if possible. As observed, the preliminary AVF analysis indicates that Posit32 TC cores are more vulnerable than TC FP32 cores in both simulation and emulation. Clearly, differences in magnitude for both results are associated with the number of sequential structures on each abstraction. Moreover, an inverse fault-susceptibility trend is observed across the TCs: in Posit32 TCs, ADD cores are more susceptible to faults than MUL cores, while in FP32 TCs, MUL cores exhibit higher fault vulnerability than ADD cores.

In summary, simulation and emulation-based analysis allow the identification of vulnerable and critical structures within the cores. However, inconsistencies among strategies may arise, as reported in Table III, suggesting the need for complementary analyses. On the other hand, the AVF estimation on the structural TCs yielded results almost equivalent (around 87%), which can be explained by the description's focus on the cores. At these levels, memory structures, such as register files and buffers, are well identified. In contrast, the descriptions of internal structures (e.g., ADD and MUL cores) are limited to high-level operations, with minimal architectural detail.

TABLE IV
AVF AND ESTIMATED AVF (eAVF) FOR CGSPN ON FP & POSIT32

Architecture	FP	Posit	FP	Posit	FP	Posit	FP	Posit	FP	Posit
SET (ns)	200	300	500	600	2,405	900	10,225	1,200	24,050	1,500
Detected	1,768	6,879	2,850	5,550	3,927	18,619	4,492	15,059	4,590	18,492
Undetected	12,094	13,334	11,012	14,673	9,935	1,601	9,370	1,136	9,272	1,281
AVF (%)	12.8	34.0	20.6	27.4	28.3	92.1	32.4	93.0	33.1	93.7
eAVF (%)	11.92	32.74	20.1	27.04	26.82	90.1	31.72	92.16	31.51	92.9
AEE (%)	3.8	3.7	2.4	1.3	5.2	1.2	2.1	0.9	4.8	0.85

Despite the limitations of the structural analyses, these provide acceptable performance when evaluating the distributional impacts of faults, as depicted in Section III-C, which illustrates the error distribution of the observed propagated faults across operations in the TCs for both formats (FP32 and Posit32). We found that errors on FP32 and Posit32 TCs mostly occur within the range 10^{-7} to 10^{18}. However, large-magnitude error effects, which are more prone to critically corrupt applications, occurred in FP32 cores, indicating higher fault susceptibility than on Posit32 ones. In particular, large error effects are mostly represented as corruptions associated with the number format (i.e., corruption in the exponent fields of the FP format). In fact, the results indicate that FP's exponent field corruptions directly produce error magnitudes that are a power of two of the corrupted field in the exponent (e.g., 10^{24}, 10^{32}, and 10^{38}). Instead, the field's organization in the Posit format contributed to decreasing the magnitude of the observed errors.

For the high-level modeling, the CGSPN analysis enabled us to estimate the AVF of both the FP and Posit TC implementations using the methodology described earlier. The main challenge was preserving the TC's functional behavior while accurately capturing SET-induced fault effects and propagation. As shown in Table IV, modeling the TC with time-specific SET faults in a CGSPN yields an Absolute Estimation Error (AEE) below 5%, confirming sufficient accuracy for AVF estimation. Each CGSPN simulation required about five minutes per faulty component. Because of architectural symmetry, we only needed to inject faults into one Multiplication Unit and one of the two adders, resulting in four total simulations. This results in a total runtime of roughly 20 minutes—far faster than a full, exhaustive fault-injection campaign, which typically requires several hours. These results demonstrate that the proposed CGSPN methodology effectively supports SET-fault analysis.

D. Analysis of Inconsistencies

A complementary evaluation of *Multiple Event Upsets* (MEUs) was performed on the TCs to remove inconsistency sources and determine the most susceptible structures in FP32 and Posit32 TCs. In particular, we used SHADOWFI to evaluate both TCs across 8,000 experiments in which the targeted sequential cells and injection times were randomly selected. In detail, we evaluated the impact of MEUs injecting 2, 3, 4, 5, and 6 bit-flips per injection.

Table V reports the fault evaluations for MEU. As expected, the results show that the greater the number of corrupted bits, the greater the impact on TC operation in both FP32 and Posit32. However, Posit32 TC exhibits higher AVF sensitivity than FP32. In detail, when injecting 2 bit-flips, the TC FP32's AVF reached 10%, while the AVF for Posit32 reached 46%.

TABLE V
MEU AVF USING EMULATION ON BOTH TCS

TC format	Structures	MEU-2	MEU-3	MEU-4	MEU-5	MEU-6
FP32	overall	0.105	0.140	0.149	0.198	0.228
	MUL	0.109	0.129	0.153	0.170	0.188
	ADD	0.100	0.151	0.146	0.226	0.269
Posit32	overall	0.460	0.590	0.685	0.781	0.823
	MUL	0.437	0.551	0.652	0.736	0.789
	ADD	0.482	0.629	0.719	0.826	0.857

Nonetheless, when injecting 6 bit-flips, the AVF for Posit TC increases significantly (82%), whereas the AVF for PF TC barely reaches 22%. Regarding the fault-susceptibility of the internal cores, the MEU results show that, in most cases, for both TCs, ADD cores are slightly more vulnerable to fault corruption than MUL cores.

The structural placement of the ADD cores near the outputs and within the cone of influence of the primary outputs of a TC increases the likelihood that faults propagate directly to the outputs and manifest as observable errors. In contrast, faults in MUL cores may be masked by subsequent operations in the ADD cores within the TC, thereby reducing overall error propagation.

IV. INPUT SELECTION FOR DNN RADIATION TESTING

This section discusses the challenges of input selection for the reliability assessment of large DNNs. As reliability evaluation methods such as physical and software FI are inherently constrained by time and resources, careful input selection is necessary to obtain representative results.

A. The Challenge of Input Selection

Assessing the reliability of large DNNs through beam or fault simulation is becoming increasingly complex and often unfeasible due to the many combinations of dataset samples, model variants, and hardware accelerator optimizations. For example, testing the entire ImageNet validation set (50,000 samples divided into 1,000 classes) under realistic radiation conditions, and assuming we would test each sample for 1 hour of beam time, *it would take* ≈ 5.7 *years to gather enough statistical data for the ImageNet validation set*. Notably, this is a very optimistic assumption [30], [31], as testing complex devices might yield zero events even after a long period of beam testing [32]. A common strategy to manage the complexity of DNN reliability assessment is to evaluate only a small subset of random inputs [31], [33]. However, if this selection is not done carefully, the evaluation may be biased [34]. Different input subsets can yield drastically different failure rates. Some subsets may yield very few errors (optimistic scenario), while others may lead to significantly higher failure rates (pessimistic scenario). Developing improved, customized methods for beam testing DNNs is necessary to reduce the bias introduced by naive random selection.

Some works have explored input selection strategies to evaluate the robustness of DNNs, particularly in software testing and adversarial robustness [34]. These methods typically rely on criteria such as neuron coverage (i.e., which neurons are activated during input processing), surprise adequacy (distance between inputs), and uncertainty estimation from output probabilities [34]. Coverage- and distance-based techniques often have high computational costs and do not scale well to large models and datasets (e.g., transformers). In contrast, uncertainty metrics such as prediction variance or maximum probability are lightweight and effective as selection criteria [35]. We therefore use a confidence-based input selection methodology to bound DNN reliability between optimistic and pessimistic scenarios. We use the prediction probabilities that the models output in the classification task as a measure of confidence (i.e., the higher the $TOP-1$ probability, the higher the model's confidence in its prediction). Using the distribution of prediction confidences over the dataset, we partition inputs into *High-confidence* and *Low-confidence*. We hypothesize that *High-confidence* inputs are less likely to change their prediction under faults, whereas *Low-confidence* inputs are more prone to fault-induced misclassification.

Not all faults affect DNN outputs equally. A fault can perturb DNN execution without altering the correct classification. We refer to faults that change the classification as *Critical Silent Data Corruptions (SDCs)* (misclassifications).

B. Input Selection Approach

We propose an approach for selecting inputs for DNN radiation testing by classifying inputs based on the model's confidence. This method bounds random-input critical-failure results between a very pessimistic and a very optimistic critical SDC rate. We performed physical FIs under a neutron beam and software fault simulation (SWFS) to cross-validate it. Fig. 9 depicts our methodology divided into 3 steps.

1) Profiling: The first step of our method (step ⓐ) is to profile the dataset to gather useful data. Given a selected model, we run inference on each image of the dataset validation set. In this work, we selected the ImageNet validation dataset, which contains 50,000 images and 1,000 classes. Our approach can be extended to any vision dataset. From the validation set, we can extract whether the DNN correctly predicts the class of each input and the set of probabilities associated with each class for each input, including the $TOP1$ (final classification of the model) and $TOP2$ (second-highest classification probability). All the data is stored in a database for post-processing in the next phase.

2) Data Analysis and Input Classification: Using profiling data, we can extract different metrics to select the inputs (step ⓑ). First, we classify the input images into two classes: correct predictions and incorrect ones (the model's inference matches or differs from the ground truth). In this work, we focus only on correct predictions, as they provide the most valuable insights into the model's Critical SDCs rates. Secondly, we can compute the confidence of the correct prediction subset for each image. The confidence of input is given by the difference between the $TOP1$ probability and the $TOP2$ probability ($TOP1 - TOP2$). If the difference is close to 1, the model is very confident about its result. On the other hand, if the difference is close to 0, the model is not confident about

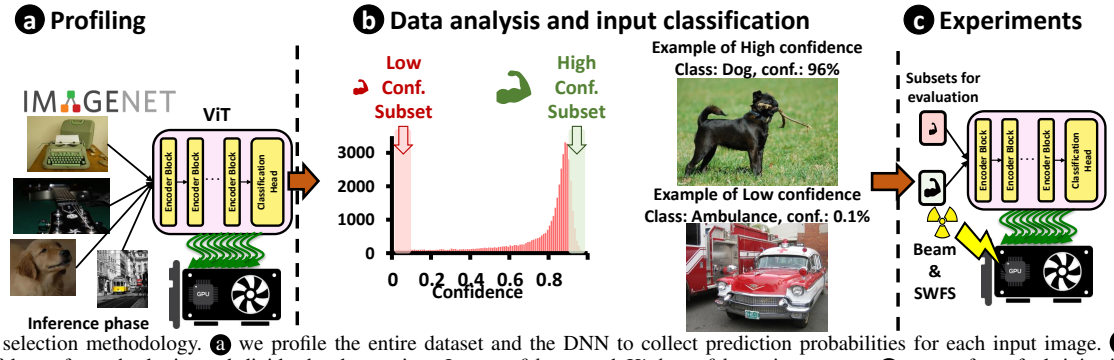


Fig. 9. Input selection methodology. (a) we profile the entire dataset and the DNN to collect prediction probabilities for each input image. (b) we compute the input confidence from the logits and divide the dataset into *Low-confidence* and *High-confidence* image sets. (c) we perform fault-injection experiments under a neutron beam and using SWFS to assess the complete dataset.

the difference between the two classes and might even be a random guess ($TOP1$ and $TOP2$ are very similar).

Figures 10b and 10c show the confidence distribution extracted for ViT and two examples of inputs extracted from the *High-confidence* and *Low-confidence* subsets. The horizontal axis shows the confidence, and the left vertical axis shows the observed frequency on the ImageNet validation dataset. We will discuss the SDC criticality results (right vertical axis) in the next section. The confidence distributions are very similar for both DNNs evaluated in this work on ImageNet. Both models are confident in the majority of inputs. We then extract two subsets from the full ImageNet dataset: the *lowest* and the *highest* confidence levels. We hypothesize that predicted confidence will affect the model's sensitivity to faults. For a fault to cause a Critical SDC (misclassification) on an input that the model is very confident in is much harder than for an image that the model has low confidence (it is easier to change the $TOP1$ probability). To illustrate how the complexity of an input affects the model's confidence, we show two real examples from the ImageNet validation set using the ViT model. 9b shows examples from the *High-confidence* set, where the ViT is 96% sure that a dog is in the image, and from the *Low-confidence* set, where the ViT is not sure whether it is an old car, an ambulance, or a fire truck, resulting in a confidence of 0.1%.

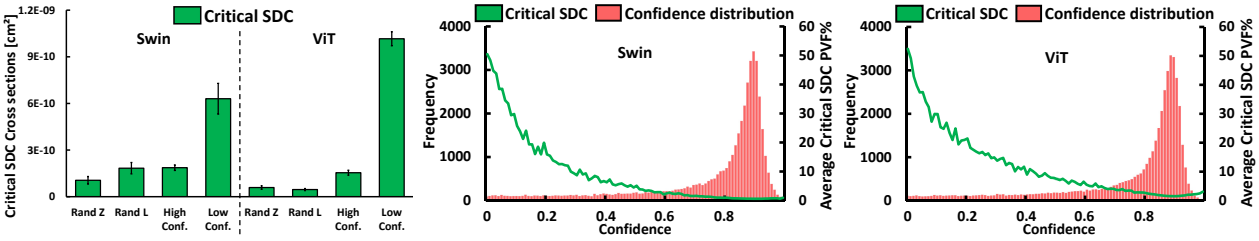
3) FI Experiments: Our analysis's final step (c) is the model reliability evaluation with 1) beam experiments to assess the effectiveness of our input selection method, and 2) SWFS to assess the complete dataset and cross-validate our method. In this step, we perform beam experiments to determine the failure rates of the different subsets (*High-/Low-confidence*) we extracted in step (b). In the last phase, we perform SWFS across the entire dataset to reproduce the impact of neutron-induced faults at the software level. This allows us to obtain a general correlation between the critical SDC rates and the confidence for all inputs. To ensure our fault simulations reflect realistic hardware behavior, we derive our fault model directly from beam experiments using a dedicated microbenchmarking procedure.

4) Evaluated DNNs and Inputs: We evaluated 2 ViT models from the HuggingFace library (v0.8.19) [36]: ViT-B-224, denoted as ViT [37], and Swin-B-W7-224, denoted as Swin [38]. ViT and Swin achieve accuracies of 84.50% and 85.20%, respectively. For the experiments, we used Python and PyTorch v2.0.0 to load the ViT and perform inferences on a batch of images from the ImageNet dataset [39].

We compare the proposed approach with the common random input selection used in previous works [31], [33]. The Random subsets were selected by setting two different random seeds. The first is with `seed = 0`, denoted Rand Z, and the second is with a large seed (`seed = 29052001`), denoted Rand L. For the neutron beam experiments, we tested four configurations for each model. We build batches of 32 images for each subset: 1) *Low-confidence* input sets are built with the 32 lowest confidence predictions; 2) *High-confidence* with the 32 highest confidence predictions; 3) *Inputs of random batches are picked among the whole ImageNet validation set*. It is worth noting that the *High-confidence* and *Low-confidence* sets include only correctly predicted inputs, as our goal is to identify inputs that may be more or less vulnerable under fault conditions. In contrast, the Random subsets include both correctly and incorrectly predicted inputs, following common practice in prior state-of-the-art DNN reliability studies [31], [33]. This distinction does not affect our conclusions, since a Critical SDC is defined as a fault-induced change in the model's $TOP1$ prediction relative to the fault-free execution, rather than relative to the ground-truth label.

5) Physical FI at ChipIr: We performed physical FI at the ChipIr facility of the Rutherford Appleton Laboratory, UK [40]. This facility has been designed to perform neutron-beam testing of electronic devices, providing an adequate test bench needed for the experiments. It notably allows multiple devices to be exposed simultaneously. ChipIR provides energy ranges similar to those in the atmosphere, with a neutron flux $10^9\times$ higher than terrestrial. On average, each configuration received a fluence of 3.33×10^{10} n/cm².

We performed neutron experiments on 3 NVIDIA RTX A2000 GPUs with the Ampere architecture. RTX A2000 has SECDED ECC to protect registers, caches, and GDDRs. We conducted all tests with ECC enabled. Our beam experiments focused only on GPU core errors (beam spot set to an $3.4cm \times 3.0cm$ area to avoid affecting other components).



a) Critical SDC cross-sections b) Swin PVF vs confidence c) ViT PVF vs confidence

Fig. 10. Critical SDC cross-sections and correlation between the confidence distribution and the average Critical SDC PVF%.

C. Software Fault Simulation

Our SWFS framework injects faults into the activations of a selected ViT layer. Because simple models such as single-bit flips are often unrealistic [20], we use fault models shown to represent how low-level hardware faults propagate through activation computations in large accelerators [41], [42].For each injection, we modify the targeted layer's activations by applying a selected relative error to the entire batch, and we repeat this over the full dataset to enable per-input correlations. For a given tensor, we corrupt a row/line or a block by multiplying each value by the chosen relative error.

We implement FI during model execution with a Python script to measure fault impact over the full dataset for each model. Injecting a single fault across all correctly predicted ImageNet validation inputs takes about 2 hours on average, which limits the campaign size. We target the Encoder Blocks of the ViT and Swin architectures, sampling 100 random faults and injecting them at three blocks of each ViT model (first, middle, last). From SWFS, we compute the Program Vulnerability Factor (PVF), i.e., the probability that a fault propagates to the application output as an SDC [43]. Overall, the campaign exceeded 1200 hours of computation.

D. Experimental Results

Beam Experiments: Fig.10a depicts the Critical SDC cross-sections (y-axis) for the assessed configurations (model and input subset) shown on the x-axis. For each configuration, the cross-section is computed as the number of observed Critical SDCs divided by the neutron fluence. This cross-section shows the probability of a Critical SDC per circuit area (the likelihood of a Critical SDC per area). Values are reported with 95% confidence intervals [44].

The *Low-confidence* subsets are the most vulnerable configurations, and generate the highest Critical SDC cross-sections. The average Critical SDC cross-section of *Low-confidence* sets is 8.23×10^{-10}. Contrarily, random and *High-confidence* configurations show an average Critical SDC cross-section of 1.7×10^{-10}. These results support our hypothesis, even a small error that propagates through the DNN can change the probability of the $TOP1$ class for inputs the model is not confident in, leading to a Critical SDC. Note that *High-confidence* set exhibits a similar Critical SDC rate to random sets. For the Swin model, the Critical SDC cross-section for the Random subsets is on average 1.45×10^{-10} and 1.87×10^{-10} for *High-confidence* subset. A similar result is found for ViT, the Random subsets have an average cross-section of 5.25×10^{-11} and the *High-confidence* has a cross-section of 1.54×10^{-10}. This further confirms the idea that selecting random inputs leads to cross-sections closer to an optimistic result. Finally, in our beam experiments, including initially misclassified inputs in the Random subsets did not affect the measured Critical SDC cross-sections, as no fault-induced changes in the $TOP1$ prediction were observed for these inputs relative to their fault-free execution.

Software Fault Simulation: To complement the beam experiments, we performed SWFS and focused on the occurrence of Critical SDCs across different input confidence levels. Figures 10b and 10c show the correlation between the input confidence and Critical SDC rates. The x-axis represents the input confidence values, from 0 to 1. We partition the full confidence range into 100 equal-sized bins. The left y-axis shows the frequency for each confidence bin across the dataset. The right y-axis corresponds to the average critical SDC PVF per confidence bin (i.e., the number of critical SDCs for a given bin divided by the number of injected faults).

Similarly to our beam experiment results, on average, Swin has shown to be more robust to critical faults, with lower Critical SDC rates, 2.70% on average, compared to 4.47% for ViT. The maximum Critical SDC rates are observed for the lowest-confidence bin (0–0.01), reaching 52.41% for ViT and 50.43% for Swin. At the other extreme, Swin also achieves the lowest Critical SDC rate, 0.60% for the 0.89–0.90 confidence bin, compared to 1.53% for ViT in the 0.90–0.91 bin. Overall, both DNNs show a strong inverse correlation between input confidence and Critical SDC PVF%. As confidence rises, the likelihood that a fault causes a misclassification drops sharply.

Discussion: In neutron beam experiments, *Low-confidence* subsets result in 7.79× more Critical SDCs than Random subsets on average, and can increase to 22.13× in the case of ViT with Rand L configuration. Conversely, the *High-confidence* subset only results in 2.19× more Critical SDCs than Random subsets. These results demonstrate that random input selection can severely underestimate DNN vulnerability. SWFS corroborate these findings. The *Low-confidence* subsets consistently show the highest Critical SDC rates, averaging 51.30% across models, compared to just 1.49% for *High-confidence*. Swin shows the largest disparity, 0.52% for *High-confidence* versus 51.75% for *Low-confidence*. Again, random configurations resemble the *High-confidence* case, further confirming their optimistic bias.

V. Conclusions

This work describes three approaches to understand the fault vulnerabilities and reliability challenges of hardware accelerators and software powering today's AI systems.

First, it presents a comprehensive design space exploration of reliability-performance trade-offs for Reconfigurable Systolic Array (RSA) CNN accelerators. In this regard, it presents a reconfiguration strategy (*RP*) for layer-wise reconfiguration of RSAs to balance reliability and performance during inference. It demonstrates that *RP* achieves 5.5% to 8.3% higher reliability than the best-performing reconfiguration strategy and $1.6\times$ to $14.7\times$ higher efficiency than the most reliable reconfiguration strategy.

Furthermore, the experiments across several abstraction levels indicate that each analysis level provides information that can guide design refinements or the development of fault-mitigation strategies. Although they examine the same circuit, the different abstractions capture distinct aspects of its structure and behavior, thereby offering complementary insights.

Finally, the third analysis, based on neutron beam and SWFS experiments, shows that model confidence strongly influences failure rates. Low-confidence inputs are significantly more vulnerable to hardware faults than higher-confidence inputs. This input-dependent sensitivity highlights the need for systematic input analysis prior to reliability assessment.

Acknowledgment

This paper is supported in part by EU Grants #101160182 TAICHIP and #101194287 NexTArc, the Estonian grant PUT PRG1467 CRASHLESS and CoE TK202 Universum.

References

[1] B. Dally, " Hardware for Deep Learning ," in *IEEE Hot Chips 35 Symp. (HCS)*, 2023, pp. 1–58.

[2] A. S. Rajasekaran, F. Al-Turjman *et al.*, *AIoT: Artificial Intelligence of Things*. CRC Press, 2025.

[3] A. Pandharipande *et al.*, "Sensing and machine learning for automotive perception: a review," *IEEE Sensors Journal*, vol. 23, no. 11, pp. 11 097–11 115, 2023.

[4] Y. Chen *et al.*, "A survey of accelerator architectures for deep neural networks," *Engineering*, vol. 6, no. 3, pp. 264–274, 2020.

[5] S. Yin *et al.*, "A high energy efficient reconfigurable hybrid neural network processor for deep learning applications," *IEEE Journal of Solid-State Circuits*, vol. 53, pp. 968–982, 4 2018.

[6] C. Peltekis *et al.*, "ArrayFlex: A systolic array architecture with configurable transparent pipelining," in *DATE*, 2023.

[7] C.-J. Lee *et al.*, "ReSA: reconfigurable systolic array for multiple tiny dnn tensors," *ACM T ARCHIT CODE OP*, vol. 37, p. 24, 9 2024.

[8] M. H. Ahmadilivani *et al.*, "A systematic literature review on hardware reliability assessment methods for deep neural networks," *ACM Comput. Surv.*, vol. 56, no. 6, Jan. 2024.

[9] ——, "Enhancing fault resilience of qnns by selective neuron splitting," in *IEEE 5th Int. Conf. on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, 2023, pp. 1–5.

[10] S. S. Mukherjee *et al.*, "A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor," *IEEE MICRO*, pp. 29–40, 2003.

[11] N. Cherezova *et al.*, "FORTALESA: Fault-tolerant reconfigurable systolic array for DNN inference," *Microprocessors and Microsystems*, vol. 119, p. 105222, 12 2025.

[12] R. Leveugle *et al.*, "Statistical fault injection: Quantified error and confidence," in *DATE*, 2009, pp. 502–506.

[13] J. Choquette *et al.*, "Nvidia a100 tensor core gpu: Performance and innovation," *IEEE Micro*, vol. 41, no. 2, pp. 29–35, 2021.

[14] E. Feng *et al.*, "snpu: Trusted execution environments on integrated npus," in *ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, 2024, pp. 708–723.

[15] H. Kim *et al.*, "Exploiting intel advanced matrix extensions (amx) for large language model inference," *IEEE Comput. Archit. Lett*, vol. 23, no. 1, pp. 117–120, 2024.

[16] E. Cui *et al.*, "Risc-v instruction set architecture extensions: A survey," *IEEE Access*, vol. 11, pp. 24 696–24 711, 2023.

[17] E. Talpes and athers, "Compute solution for tesla's full self-driving computer," *IEEE Micro*, vol. 40, no. 2, pp. 25–35, 2020.

[18] D. Gnad *et al.*, "Reliability and security of ai hardware," in *IEEE European Test Symp. (ETS)*, 2024, pp. 1–10.

[19] L. Yang *et al.*, "Gpu reliability assessment: insights across the abstraction layers," in *Int Conf. on Cluster Computing (CLUSTER)*, 2024, pp. 1–13.

[20] G. Papadimitriou and D. Gizopoulos, "Demystifying the system vulnerability stack: transient fault effects across the layers," in *ISCA*, 2021.

[21] O. Chatzopoulos *et al.*, "Measuring the unmeasurable: Demystifying silent data corruptions in ai accelerators through microarchitectural modeling," *IEEE Micro*, pp. 1–11, 2025.

[22] J.-D. Guerrero-Balaguera *et al.*, "Shadowfi: An open-source framework for fault evaluation of complex ic designs using hyperscale computing," *IEEE Access*, vol. 13, pp. 211 382–211 406, 2025.

[23] A. Veronesi *et al.*, "Ember: A cycle-based framework for early-stage reliability assessment in parametric rtl designs," in *34th Asian Test Symp. & The 9th Int. Test Conf. Asia (ATS/ITC-Asia 2025)*, 2025, pp. 1–6.

[24] E. C. V. Castillo *et al.*, "An efficient approach for stls development of automotive socs using colored petri nets," in *DDECS*, 2024, pp. 98–103.

[25] E. C. V. Castillo, F. A. Da Silva *et al.*, "Diagnostic coverage estimation for automotive socs based on colored stochastic petri nets," in *VLSI-SoC*, 2024, pp. 1–6.

[26] E. C. V. Castillo, J.-D. Guerrero-Balaguera *et al.*, "Early reliability estimation in hardware accelerators using improved colored petri nets," in *ITC*, 2025, pp. 498–501.

[27] E. C. V. Castillo *et al.*, "A predictive framework for fault vulnerabilities in edge hardware accelerators," in *LASCAS*, 2026.

[28] M. Cannon *et al.*, "Multiscale system modeling of single-event-induced faults in advanced node processors," *IEEE Trans. on Nuclear Science*, vol. 68, no. 5, pp. 980–990, 2021.

[29] R. L. Sierra *et al.*, "Analyzing the impact of different real number formats on the structural reliability of tcus in gpus," in *VLSI-SoC*, 2023, pp. 1–6.

[30] I. Rodriguez-Ferrandez *et al.*, "Proton evaluation of single event effects in the nvidia gpu orin som: Understanding radiation vulnerabilities beyond the soc," in *IEEE IOLTS*, 2024, pp. 149–155.

[31] P. Rech, "Artificial neural networks for space and safety-critical applications: Reliability issues and potential solutions," *IEEE Trans. Nucl. Sci.*, vol. 71, no. 4, pp. 377–404, 2024.

[32] H. Quinn *et al.*, "Measuring zero: Neutron testing of modern digital electronics," *IEEE Trans. Nucl. Sci.*, vol. 71, no. 4, pp. 670–679, 2024.

[33] J. M. Badia *et al.*, "Reliability of vision transformers and cnns on edge ai systems under neutron radiation," *IEEE Trans. Nucl. Sci.*, vol. 72, no. 8, pp. 2706–2716, 2025.

[34] S. Wang *et al.*, "A survey on test input selection and prioritization for deep neural networks," in *ISSSR*, 2024, pp. 232–243.

[35] W. Ma *et al.*, "Test selection for deep learning systems," *ACM Trans. Softw. Eng. Methodol.*, vol. 30, no. 2, pp. 1–22, 2021.

[36] R. Wightman, "Huggingface," huggingface.co/timm.

[37] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *ICLR*, 2021, pp. 1–21.

[38] Z. Liu *et al.*, "Swin transformer: Hierarchical vision transformer using shifted windows," in *IEEE/CVF ICCV*, 2021, pp. 9992–10 002.

[39] J. Deng *et al.*, "ImageNet: A large-scale hierarchical image database," in *IEEE CVPR*, 2009, pp. 248–255.

[40] C. Cazzaniga *et al.*, "Fast neutron measurements for the characterization of the chipir beamline," *IEEE TNS*, vol. 71, no. 8, pp. 1520–1526, 2024.

[41] Y. Sun *et al.*, "Demystifying the resilience of large language model inference: An end-to-end perspective," in *SC*, 2025, p. 1127–1144.

[42] C. Bolchini *et al.*, "Fast and accurate error simulation for cnns against soft errors," *IEEE TC*, vol. 72, no. 4, pp. 984–997, 2022.

[43] V. Sridharan and D. Kaeli, "Eliminating microarchitectural dependency from architectural vulnerability," in *IEEE HPCA*, 2009, pp. 117–128.

[44] H. Quinn, "Challenges in testing complex systems," *IEEE Trans. Nucl. Sci.*, vol. 61, no. 2, pp. 766–786, 2014.