Investigating and Mitigating Critical Faults in Floating-Point and Posit Arithmetic Hardware

Investigating and Mitigating Critical Faults in Floating-Point and Posit Arithmetic Hardware

Josie E. Rodriguez Condia, *Member, IEEE,* Juan-David Guerrero-Balaguera, *Member, IEEE,* Robert Limas Sierra, *Student member, IEEE,* Matteo Sonza Reorda, *Fellow, IEEE*

***Abstract*—Mature computing formats, such as Floating-Point (FP), provide optimal accuracy to process real values and are essential in most scientific domains. However, the massive market adoption of highly parallel systems, with advanced technology nodes, in several domains exacerbates the need for highly reliable systems. Formerly, most reliability evaluations targeted FP hardware. Unfortunately, fine-grain assessments on cores with recent arithmetic format alternatives, such as Posit (particularly suited for Artificial Intelligence), have remained partially unexplored. Similarly, the effects of corruption on operations due to faulty hardware are not well-known, which may prevent the proposal of effective mitigation mechanisms. This work exhaustively evaluates the fine-grain effects of permanent faults in the hardware of arithmetic cores for the three most extensively used operations in modern applications (*Add*, *Multiply*, and *Multiply and Add*), including machine learning, implemented in Posit and FP. Our results indicate that Posit cores are less fault-vulnerable than FP ones. However, Posit cores are more prone to induce significant operational corruption than FP ones (5.2% to 7.5%). We also found that absolute errors in faulty FP cores are higher by up to 2 orders of magnitude than in Posit ones. Finally, we applied and evaluated three mitigation mechanisms (*Self-Check and repair*, *Dual Modular Redundancy*, and *Triple Modular Redundancy*), effectively reducing the most critical errors with moderate area costs (20% to 110%).**

***Index Terms*—Arithmetic circuits, Artificial Intelligence, Floating-point formats, Permanent faults, Posit formats, Reliability evaluation, Selective hardening.**

I. Introduction

ARITHMETIC hardware cores are crucial units in modern computational systems, ranging from general-purpose processors to specialized hardware accelerators, such as Graphics Processing Units or GPUs. These units significantly enhance performance across a broad spectrum of domains, including scientific applications in High-Performance Computing (HPC), low-power computing on Internet of Things (IoT) devices, and operations in safety-critical fields, such as self-driving and autonomous vehicles [1], [2].

Advances in semiconductors and computer architectures encourage the efficient implementation of floating-point (FP) cores for the optimal execution of real-number computations. In fact, FP cores are integral parts of accelerators, such as the *Tensor Cores* (TCUs) in GPUs, for the training and inference of Convolutional Neural Networks (CNNs). Similarly, FP cores are fundamental for performing computations in critical high-precision application domains (e.g., healthcare, banking, and HPC), where the tolerance to errors is restrained, demanding highly accurate and reliable hardware [3].

Unfortunately, several studies [4], [5] demonstrated that advanced technology nodes in modern devices and systems (e.g., 7nm and below) increase the sensitivity to faults during the in-field operation due to two main time-dependent phenomena: *i) temporal* impacts, and *ii) environmental and operational* impact variations. The former impacts are associated with effects arising due to long-term operations, such as premature silicon aging and wear-out [5], [6], while the latter emerge from the system's exposure to harsh operative conditions (e.g., external radiation, electromagnetic fields, or high operational temperatures) [7]. In detail, the semiconductor vulnerabilities promote the appearance of permanent faults in the system's hardware, including arithmetic cores, that once raised might turn into errors (i.e., *corruptions*), possibly leading to application and system failures and eventually catastrophic results [8]. Clearly, these undesired circumstances raise reliability concerns about critical systems (e.g., CNNs in automotive or banking transactions) [9]. Hence, assessing fault vulnerability features in the arithmetic units of a system, which often represent a significant fraction of the total area occupied by a device's logic, is crucial to explore design alternatives, also taking reliability into account, e.g., to support the design of reliable hardware architectures.

Recently, *Posit* emerged as an alternative number format to represent and manipulate operations with real values. The outstanding characteristics of *Posit* (e.g., superior dynamic range and precision w.r.t. the classical FP IEEE-754 standard [10]) and the increasing hardware support for specialized *Posit*-compliant commercial hardware, like IP cores for soft-processors and GPUs[1], make this format an attractive solution for multiple application domains, especially in the design of machine learning accelerators [11]. Moreover, *Posit* offers a promising alternative to increase the reliability of modern computational hardware architectures [12]. Nonetheless, the micro-architecture reliability features of the hardware implementing *Posit* operations (e.g., MAC) have been barely studied, leaving their fault sensitivity and data corruption effects highly unexplored compared to equivalent FP hardware.

To the best of our knowledge, only a few works have focused on the comparative evaluation of the reliability features of Posit and FP formats. Most works used hardware-agnostic approaches (i.e., *Formal* / *Software*) to characterize the

Josie E. Rodriguez Condia, Juan-David Guerrero-Balaguera, Robert Limas Sierra, and Matteo Sonza Reorda are with the Department of Control and Computer Engineering, Politecnico di Torino, Turin, Italy.

Manuscript received Month XX, 202X; revised Month XX, 202X.

[1] *https://vivid-sparks.com/*

format's resilience for several domains, including scientific computing and Machine Learning (ML), through the analysis of transient effects during the application's execution (i.e., bit-flips corrupting values) [12]–[15]. Those works mainly indicate that the *Posit* format is inherently more error resilient than FP. Unfortunately, those evaluations can, at most, represent corruptions stemming from faults in memories or registers inside a system, neglecting the impact of permanent faults in the arithmetic core's logic. Thus, such evaluations lead to incomplete/inaccurate assessments, which are insufficient to support design improvements or develop appropriate fault countermeasures [16].

In arithmetic units, two main factors: *i)* the target operation (e.g., MUL), and *iii)* the format specifications (e.g., FP) determine the overall micro-architecture of a unit, as well as its structural composition and internal interconnections (e.g., need of decoders, binary adders, or shifters), which may determine how internal faults induce errors and how these propagate to the outputs, affecting the overall core reliability. Consequently, it might be possible that arithmetic units adopting a particular format for some operations (e.g., addition in *Posit*) show more fault resilience and, in turn, better reliability than others, or vice-versa.

Motivated by the extensive use of high-precision computations in modern systems and the imperative need to characterize fault effects on the adopted hardware to prevent vulnerabilities [5], this paper extends a preliminary work [17], in which we conducted a first attempt to characterize the impacts of faults in the hardware of arithmetic cores for two operations (*ADD* and *MUL*) on both formats (*FP* and *Posit*).

This work goes beyond the initial reliability characterization performed in [17]: it extends the assessments to other units, it widens the number and type of considered metrics, and it explores the usage of hardware-based selective fault mitigation mechanisms for three essential operations in hardware accelerators for machine learning: *Addition (ADD)*, *Multiply (MUL)*, and *Multiply and Add (MAC)*.

Our evaluation focuses on the identification of fault effects stemming from permanent faults affecting fine-grain circuits (e.g., gate connections) in arithmetic hardware cores for two number formats (*FP* and *Posit*). By estimating the magnitude of the produced error (if any) for every fault, our evaluation identifies faults and their associated structures causing the most critical effects [18]. Then, these associated structures are our main targets for mitigation through selective hardening mechanisms. We adopt hardware-based hardening mechanisms to reduce latency impacts during core computation (e.g., data-intensive workloads like CNN operations processing thousands of operations and involving performance restrictions).

The fault characterization and evaluation resort to experiments on 13 open-source cores for *ADD*, *MUL*, and *MAC* operations from state-of-the-art processors and accelerators. In detail, we evaluated two cores per operation for each format (*FP* and *Posit*). Moreover, we analyze the fault vulnerabilities of one `quire` *Posit* MAC core due to micro-architecture differences. In all experiments, we used matrix multiplication as the reference operation due to its key role in executing ML workloads.

The main contributions of this work are:

- We propose an experimental strategy to extensively evaluate the fine-grain impacts of hardware faults in several FP and Posit arithmetic cores, quantitatively demonstrating that most large-magnitude error corruptions do come from faults affecting specific substructures computing the exponents and regime fields of each core in both number formats (Section VI-A);
- We experimentally demonstrate that fault-sensitive structures in `non-quire` Posit cores are more vulnerable to fault propagation and cause more errors than those in FP cores (from around 5.2% to 7.5%). However, we also found that the absolute error can be up to 2 orders of magnitude larger in FP cores in comparison with errors in Posit units (Sections VI-A and VI-B).
- We found that `quire` Posit MAC cores show better overall structural resilience, i.e., they are less likely (by up to 10.3%) to cause errors than FP and `non-quire` Posit MAC cores. However, the few errors in `quire` Posit MACs are more critical (i.e., have a larger magnitude) than those in FP MAC cores, mainly due to the missing structures masking some effects (Sections VI-A and VI-B).
- We evaluated three hardware-based selective-hardening mechanisms (*Self-checking and Repair*, *Dual Modular Redundancy*, and *Triple Modular Redundancy*) targeting the identified fault-sensitive structures in the arithmetic cores for both formats to effectively reduce large-magnitude errors with limited overhead costs (Section VII).

The manuscript is organized as follows. Section II examines existing literature and outlines the key contributions of this work. Section III provides background on faults and errors and their impacts on modern systems. It also introduces the FP and Posit number formats, along with a brief overview of their hardware architecture. Section IV describes the method for reliability assessment and selective hardening of arithmetic hardware. Section V details the experimental setup. Section VI reports and analyzes the architectural and critical corruption effects from faults in the hardware cores. Then, Section VII describes the adopted selective hardening mechanisms for both FP and Posit cores, targeting the vulnerable structures causing critical errors. Section VIII provides a discussion, and Section VIII concludes the paper and outlines directions for future work.

II. Related works

In the literature, most works addressed the resilience of real number formats by resorting to two methods: *i)* formal analyses and *ii)* experimental evaluations. The formal (hardware-agnostic) approaches use information theory foundations to determine the resiliency features of a format [19], their coding benefits, and highlight their critical constraints, such as numerical analysis support [20].

In contrast, experimental approaches mostly resort to software evaluations on applications to identify the format's coding behaviors and determine its accuracy and error resiliency [21]. In [22], the authors evaluated (in software) the *Posit* and

Fixed-Point formats on CNNs, suggesting that *Posit* provides accuracy advantages. Authors in [23] indicated that *Posit* reduced coding errors by up to 3 orders of magnitude when compared with *FP*. In [12], the authors highlighted the utility of *Posit* on CNNs due to their improved accuracy for the range 10^{-6} to 10^{6}. However, the immature numerical analysis of *Posit* to represent physical and mathematical constants might compromise their resilience, so suggesting *Posit* as a storage format only, while it might be used as an arithmetic complement to FP. Authors in [15] evaluated (in software) the resilience of one CNN in *Posit* and *FP* formats. Experiments corrupting (by bit-flips) the format bit-fields of the CNN weights on individual layers indicated that a CNN in *Posit* has equivalent reliability as the one in *FP*. Other works evaluated the robustness and sensitivity of *FP* and *Posit* formats through software bit-flips on CNNs and scientific workloads, indicating that *Posit* is inherently more error resilient than *FP* and might be used in the safety-critical, approximate computing, and HPC domains [13] [14], as well as in CNN's training tasks [24]. Nevertheless, the evaluation's scope of the previous works targeted the data format properties at the software level only, neglecting the impacts of hardware faults/defects inside the logic of an arithmetic core, which could lead to inaccurate/incomplete assessments of the structural reliability of the hardware implementing the operations.

Regarding hardware characterization, authors in [12], [24] showed that hardware overhead is higher in *Posit* than on *FP*, mainly due to encoding structures in *Posit* cores [12]. Other works explored efficient design strategies to reduce costs, including area, performance, power, and format accuracy (e.g., '*Fixed-Posit*' referring to the fixed regime and exponent field sizes [20], *Half-Unit Biased* (HUB) strategy on *Posit* cores [25], simple and sequential hardware architectures for division and square root in Posit [26]). Unfortunately, all works neglected the hardware-aware fault characterization of the cores and their impacts on the resilience of a running operation/application, as well as the development of fault countermeasures. In fact, the resilience of *Posit* hardware has been barely studied and remains mostly unexplored [27].

To the best of our knowledge, only one work [28] provided the first attempts to evaluate the resilience of one hardware accelerator (TCUs) under *Posit* and *FP* formats. Their results demonstrated that *Posit* cores are more resilient than FP ones from about 1 to 20 orders of magnitude. However, this work evaluated coarse-grain structures (e.g., connections between circuits) and ignored the fine-grain micro-architecture in the cores (i.e., combinational logic inside the circuits). Instead, this work goes beyond the characterization and evaluation of number formats for some hardware structures. It aims to provide valuable quantitative insights about the impact (on operations) of hardware faults affecting the fine-grain combinational logic in the circuits of *Posit* and *FP* arithmetic cores. In addition, we determined critical effects on results (large-magnitude errors) and correlated them with their corresponding source structure in the cores. Then, we used the primary analysis outcomes to evaluate selective-hardening mechanisms targeting the most vulnerable structures inside *Posit* and *FP* arithmetic cores.

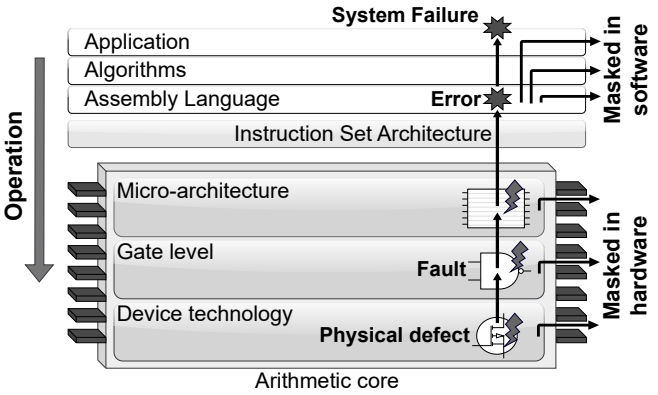

Fig. 1. A general scheme of the relation among physical defects, hardware faults, and software errors in a system with a faulty arithmetic core.

III. Background

A. Impact of Faults, Errors, and Failures in modern systems

Currently, high-performance systems resort to cutting-edge transistor technologies (i.e., 7nm or beyond) to improve performance, area, and energy efficiency in the execution of demanding and complex applications, such as Deep Neural Networks (DNNs). The high transistor density of new systems combined with reliability challenges from *temporal* and *environmental/operational* variation effects (e.g., caused by early aging, voltage dropping, electromigration, or production test scapes [5], [29], [30]) promote the premature rise of physical defects in hardware, which can be propagated as errors in software and potentially affect the system's operation by corrupting a running application/system with failures [31], as depicted in Figure 1. The physical defects in transistors of a structure, i.e., gates of arithmetic cores, might propagate a fault in software as 'errors' by corrupting an output value from an operation (i.e., *Silent Data Corruptions* or **SDCs**) or produce failures causing the collapsing of an application/system (i.e., *Detected Unrecoverable Errors* or **DUEs**). In optimistic scenarios, faults and errors might be neglected when *Masked* by the system's structure or the application's characteristics and interaction with the underlying hardware.

Unfortunately, regular and repetitive data-intensive workloads (e.g., CNNs) execute impressive amounts of arithmetic operations. Hence, the underlying hardware (e.g., processors or hardware accelerators) uses their available arithmetic cores to compute each software operation (e.g., *ADD*, *MUL*, or *MAC*), commonly producing high operational over-stress on the cores, which promotes the arising of faults and the corruption of multiple results. In addition, the intricate structures in modern arithmetic cores increase the complexity of associating faults with error effects since these commonly use specialized hardware topologies to implement operations for a given format [32], which can also affect the overall resilience of a hardware core.

Furthermore, a critical limitation arises due to the combination of the previous factors by posing performance challenges, which impede the use of conventional low-level micro-architecture techniques for the reliability evaluation of complete applications and systems (e.g., small CNN evaluations in RT-level might involve more than 10,000 days) Thus, fine-

grain reliability assessment, focusing on the arithmetic hardware, can support the determination ofoperational advantages and limitations, as well as identify the most fault-sensitive structures to further develop fault countermeasures and prevent application-level failures.

B. Hardware architectures in Floating-Point and Posit formats

Floating-point (FP) and Posit are specialized number formats to represent real numbers and enable the execution of operations with affordable and high precision levels in modern computer systems. Notably, the format's effectiveness depends on the hardware implementation support and its adoption.

FP encodes real numbers using three fields (*sign, exponent, and mantissa/fraction*) and is widely supported by industry standards (IEEE-754 [33]), mature software stacks, and efficient hardware operations (e.g., *ADD*, *MUL*, *MAC* [32]). In contrast, the emerging Posit format uses four fields (*sign*, *regime*, *exponent*, and *fraction*) with regime enabling uniform accuracy across magnitudes and extended dynamic range near ±1.0. Posit offers higher tapered precision and is viewed as a potential replacement or complement to FP [10], [12]. Posit also supports fused MAC operations via `quire` accumulators with extended precision. However, Posit struggles with representing reciprocals and physical constants.

In general, low-latency architectures of arithmetic units for both formats divide the binary code fields of the input operands. Then, specialized circuits (structures) process the fields mostly independently [10], [32], [34].

The execution of FP cores comprises four main steps: *i)* '**Sign and Exponent processing**' or 'S&E' that involves the sign calculation and the exponent normalization, alignment, or comparison, using XOR-based sign comparators, binary adders/subtractors, binary multipliers, multiplexers, and other logic. *ii)* '**Significant processing**' or 'SP' that aligns the input mantissas (when required) and computes the output significant through registers, binary shifters, binary adders, and multipliers. *iii)* '**Normalization**' or 'N' that normalizes, and composes the result using shifting mechanisms, comparators, such as count leading zeros, and other logic, and *iv)* '**Rounding**' or 'RnD' that adjust the result in case of imprecision through comparators, and binary adders [32].

Consequently, the organization of low-latency Posit cores, follows a similar approach as FP ones, e.g., *Posit(32,2)* representing a 32 bit-width and 2 bit-exponent format with fixed regime bits to reduce logic [20]. These cores operate in four main steps: *i)* '**Decoding and checking**' or 'D&C' that processes the input operands by decoding and translating the exponent and regime fields into extended binaries, and adjusts the fraction fields through two's complement decoders and comparison logic based on *"Counter Leading Zeros"* (CLZ) structures [26]) *ii)* '**Fraction operation**' or 'FO' that processes and adjusts the fraction fields (e.g., addition, multiplication, or factor scaling) through binary adders, shifting logic, and fast binary multipliers. *iii)* '**Rounding**' or 'R' that adjusts, rounds, and normalizes the results by using normalization logic [34], and *iv)* '**Encoding**' or 'EN' that encodes, and assembles the result in Posit format from the regime, exponent, and fraction bit-fields using encoder structures through registers and shifting logic. Furthermore, the Posit MAC core may include an additional step known as '**Quire Accumulation**' '(QA)' when using the *Posit* `quire` format [35]. Specifically, efficient fused *Posit* MAC cores leverage QA to accumulate intermediate results without encoding or rounding, thereby preserving numerical accuracy.

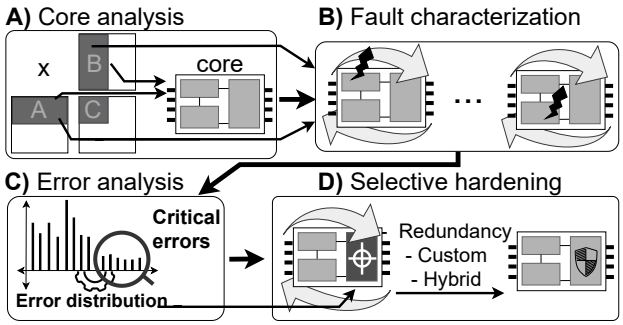

Fig. 2. A scheme of the method to analyze hardware faults in arithmetic cores and apply selective hardening on their most vulnerable structures.

IV. A Fine-grain Method to Assess and Mitigate Fault Vulnerabilities in FP and Posit Hardware

This section describes a method to assess and enhance the reliability of the hardware in *FP* and *Posit* arithmetic cores, comprising four steps: *A) Core analysis and evaluation data*, *B) Fault characterization*, *C) Error analysis and path tracing*, and *D) Selective hardening*, as depicted in Figure 2. First, a core is analyzed and intensively evaluated through fault simulation. Then, the method uses fine-grain error identification to correlate vulnerable circuits with the observed critical errors and determines feasible targets for hardening. In our approach, critical errors are those able to produce a large-magnitude impact on the output (e.g., a difference > 1.0 between an expected value and a corrupted one). Those vulnerable circuits in the cores, prone to critical errors, can corrupt complete CNN workloads [21] and are the main hardening targets.

A. Core analysis and evaluation data

This step characterizes the architecture and the functional operation of every arithmetic core *(a.k.a. functional unit)* whose micro-architectural description (i.e., RT- and gate-level) is available. This analysis determines a clear hierarchical organization of the core's structures for the reliability assessment, which allows the identification of fault-sensitive structures to support the hardening step. Moreover, we identify and select synthetic or representative data workloads for a targeted domain, e.g., *Convolution* as *Matrix Multiplication*, or MxM [34] for CNN applications. These workloads offer realistic input stimuli across appropriate operational ranges, effectively exciting core structures during fault characterization.

B. Fault characterization

This step assesses the impact of faults (e.g., permanent) on the fine-grain structures of the cores (i.e., gate interconnections) by targeting all possible fault sources/sites in the

cores through exhaustive fault injection campaigns. First, we determine a reference *(fault-free)* execution of a targeted core under the complete input workload (later used for comparison and fault classification). Then, a fault injection campaign starts by forcing one hardware fault in the core, and each stimuli vector in the workload is applied and evaluated independently (e.g., 10 ADD vectors require 10 fault simulations to assess each under the same hardware fault). Each operation's output is then collected and stored for later analysis.

We use an automatic framework to handle the fault injection campaigns on the cores. Moreover, the same stimuli/vectors from the benchmarks, encoded in their respective number format, serve as input for the fault characterization experiments.

C. Error analysis and path tracing

This step analyses the fault characterization results and identifies large-magnitude (*critical*) errors (e.g., > 1.0) associated with the most fault-vulnerable structures in the cores likely to cause them. These critical errors are our main analysis targets since low-magnitude errors are usually masked by the application resilience, e.g., in CNNs [18].

For each core, we analyze the effects at two levels: *i)* the structural impacts from the activation and propagation of faults, and *ii)* the corruption impacts after error generation. First, we evaluate the overall fault vulnerability of a core (i.e., fault activation and propagation rate as the accumulation of *SDCs*), which serves to identify the individual contribution of the core's structures to the error generation. Then, we adapt and compute the Absolute Error (AE) and its cumulative distribution per core to identify faults causing critical corruptions in the outputs (i.e., large-magnitude errors) [36]. Finally, we find and correlate the error's magnitude with the faulty structures to identify susceptible hardware highly prone to critical errors and determine the main candidates for mitigation.

D. Selective hardening

This step targets the adoption and development of hardware-based selective hardening mechanisms for the most fault-sensitive structures in the cores (i.e., those causing large-magnitude errors). In literature [18], it has been demonstrated that small errors have limited corruption effects for some applications, such as DNNs. Nonetheless, large error magnitudes are critical when propagating silently throughout the workload execution. In particular, our approach only targets the highly fault-sensitive structures, while others remain unchanged.

We use a bottom-up approach to adapt, develop, and validate the mitigation mechanisms' effectiveness after identifying vulnerable structures in a core. In particular, we first search for regular structures inside and among the cores and then use one or more hardware-based hardening strategies (e.g., redundancy, diversity, custom, hybrid, or a smart combination of them). Then, the validation of a hardening mechanism resorts to complementary focused fault injection campaigns and error analyses.

V. Experimental Setup

The evaluation aims at studying the impact of permanent faults (*Stuck-at faults* $0/1$ *or SAFs*) in the core structures. Thus, we considered 13 open-source arithmetic cores at 32-bit size executing one of three operations (***ADD***, ***MUL***, or ***MAC***) in both formats (FP and Posit). In detail, we evaluated two cores per operation and one Posit `quire` MAC core.

The FP cores (FP_X, where X indicates the operation, e.g., ***ADD***, ***MUL***, or ***MAC***), along with the `quire` (PQ_X) and `non-quire` (P_X) Posit cores, are derived from four distinct sources: one GPU (*FP_ADD*, *FP_MUL*, and *FP_MAC*), one soft-core generator, namely *FloPoCo*, (FP_ADD_F, FP_MUL_F, and FP_MAC_F), one hardware accelerator (*P_ADD*, *P_MUL*, *P_MAC*, and *PQ_MAC*), and one state-of-the-art processor (*P_ADD_F*, *P_MUL_F*, and *P_MAC_F*) [11], [34], [37], [38]. It is worth noting that *FloPoCo* cores lack hardware support for handling exceptions. However, since our analyses primarily focus on identifying errors likely to cause SDCs, which mostly propagate by bypassing system exceptions or interruptions, the absence of exception-handling hardware does not undermine the validity of our analyses.

Our evaluations use gate-level netlists of the cores after logic synthesis using a 15nm technology library [39] with an operational frequency of 500Mhz per core through the *Design Compiler* (*Synopsys*) tool. It is worth noting that the core's hierarchy is preserved to identify vulnerable sub-structures. Table I summarizes the main features per core. An examination of the cores' architectures indicates that identical low-latency algorithms (Booth-based multipliers and Carry-lookahead adders) are used across all cores to implement their fundamental structures.

A custom framework (based on *Soft-Posit* and *Numpy* libraries) automatically generates the input stimuli vectors by extracting the arithmetic operations (*ADD*, *MUL*, and *MAC*) in 3 ranges ($\pm 1.0, \pm 10.0$, and ± 100.0) from matrix multiplications as representative CNN workloads, as suggested in [34]. For each range, we characterize fault propagation effects and their output corruption using the same input vector set (encoded in their respective number format).

We developed a custom experimental framework that uses *ModelSim* (by *Siemens EDA*) and the input vectors to determine the reference results (*fault-free*) per core. Then, an industrial-grade tool (*Z01X* by *Synopsis*) is adapted for the core's fault characterization through fault campaigns on each vector from the stimuli. Each fault campaign exhaustively evaluates all available fault sites per core, considering one fault per simulation. We use smart multi-threading schemes combined with the implicit parallelism of *Z01X* to speed up the core's characterization. In the end, 39 fault injection campaigns targeted the 13 arithmetic cores, considering workloads of 4,096 stimuli vectors per MxM range and evaluating an overall of 4.91×10^9 fault effects. The experiments cumulatively required around 320 hours on two servers with 12 Intel Xeon CPUs at 2.5 GHz and 256 GB of RAM.

VI. Evaluation of arithmetic cores

This section reports and analyses the micro-architecture fault characterization and sensitivity of the cores for both

TABLE I
FEATURES OF THE ARITHMETIC CORES IN BOTH FORMATS.

Format	Operation	IP core	Cells	Area (μm^2)	SAFs
Posit	ADD	P_ADD	1,816	634.9	12,900
		P_ADD_F	1,592	581.7	11,846
	MUL	P_MUL	3,921	1,721.1	26,510
		P_MUL_F	2,991	1,503.3	22,038
	MAC	P_MAC	14,024	5,040.8	94,083
		P_MAC_F	9,712	3,745.2	61,830
		PQ_MAC	13,896	6,260.8	81,796
FP	ADD	FP_ADD	1,275	345.3	9,366
		FP_ADD_F	1,043	387.1	6,960
	MUL	FP_MUL	1,531	611.1	11,518
		FP_MUL_F	2,004	984.8	13,860
	MAC	FP_MAC	2,815	956.6	20,904
		FP_MAC_F	3,951	1,651.7	26,586

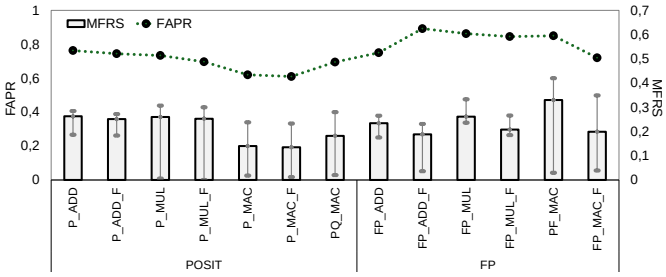

Fig. 3. *Probability of Fault Activation and Propagation* (FAPR) and *Mean Fault Rate per Stimuli* (MFRS) on the analyzed arithmetic cores.

formats. Then, we analyzed the operation's accuracy impacts due to corruption effects caused by the propagation of faults.

A. Structural Evaluation of FP and Posit Cores

The overall structural vulnerability of a core's architecture is identified through the *Fault Activation and Propagation Rate* (FAPR) [40] that describes the probability of propagation for a core's fault (causing any output corruption), as the rate between the number of faults corrupting at least one stimuli (*PF*) over the total number of faults per core (*SAFs*), see equation 1.

Moreover, we determine the average fault sensitivity of the core's architecture using the *Mean Fault Rate per Stimuli*, or MFRS, which calculates the mean rate distribution of faults corrupting every stimuli vector (*i*) over the total number of faults per core and the amount of evaluated stimuli vectors (*N*), as described in equation 2. MFRS considers the scenario of faults affecting every available fault site in a core to determine the average vulnerability of the structures per core.

$$FAPR = \frac{PF}{SAFs} \tag{1}$$

$$MFRS = \frac{\sum_{i=1}^{N} PF_i}{SAFs * N} \tag{2}$$

A preliminary analysis of the FAPR results, per MxM range, shows minimal deviations in the distribution of susceptible fault sources for all cores ($< 5\%$), indicating that typical CNN operand ranges mostly excite similar core structures and can be affected by similar amounts of fault sources.

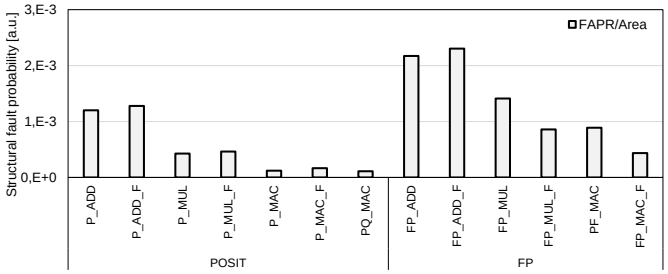

Fig. 4. *Structural Fault Probability*, corresponding to the ratio between the FAPR and the vulnerability parameters of the analyzed arithmetic cores.

Figure 3 depicts, for each core, the FAPR and MFRS obtained as the average from the evaluated ranges. The results indicate that cores affected by permanent faults have a moderate probability of activating and propagating effects on the outputs (FAPR from 0.69 to 0.89). Specifically, 72% to 89% of all exhaustively evaluated faults in FP cores are prone to activation and propagation, compared to slightly lower rates (69%–76%) observed in Posit cores.

The MFRS results show that a moderate percentage of core structures (associated with 15%-26% of faults in all cores for both formats) are highly vulnerable and can impact every core operation once any of these structures is faulty. Our findings indicate that core organization, influenced by number format and implementation complexity, impacts fault propagation sensitivity relative to input stimuli. Indeed, some cores inherently require more sophisticated structures for computation than others, e.g., the control logic for exponent alignment in FP ADD cores is absent in FP MUL ones.

Our experiments indicate that the sensitivity of fault sources (e.g., gate interconnections) varies among cores. For instance, the **MUL** Posit cores showed strong range deviations from 4 (*MFRS*=0.002) to 6,594 (*MFRS*=0.299), as depicted by the error bars. Instead, **ADD** Posit cores exhibited slight deviations, from 2,170 (*MFRS*=0.185) to 3,223 (*MFRS*=0.284). Overall, the results indicate that the number of faults possibly affecting executions in Posit **MUL/MAC**, and FP **MAC** is highly variable, while the number of faults corrupting executions in Posit **ADD** and FP **MUL** remains slightly uniform for all evaluated stimuli. Regarding the posit MAC cores, the experiments showed that `quire` posit core (*PQ_MAC*) has a higher overall fault sensitivity in up to 7% than its `non-quire` version. Still, the variation of fault sources corrupting the stimuli varies proportionally in all MAC cores.

Both, FAPR and MFRS, support the core analysis concerning their structural fault vulnerability. However, they can hardly be used to compare cores with different formats without considering other factors, such as the area and gate technology. In this regard, we calculate a relative *Structural Fault Probability* per core, or *SFP*, as the relation between the FAPR and a vulnerability parameter (associating technology and physical features). Since all experiments used the same technology library, we assume constant technology parameters per core. Moreover, we exhaustively evaluated all faults per core and used the same input stimuli vectors. However, the core's area differs, so we divide the vulnerability parameter by the core's area to preserve the relation between area and faults per core

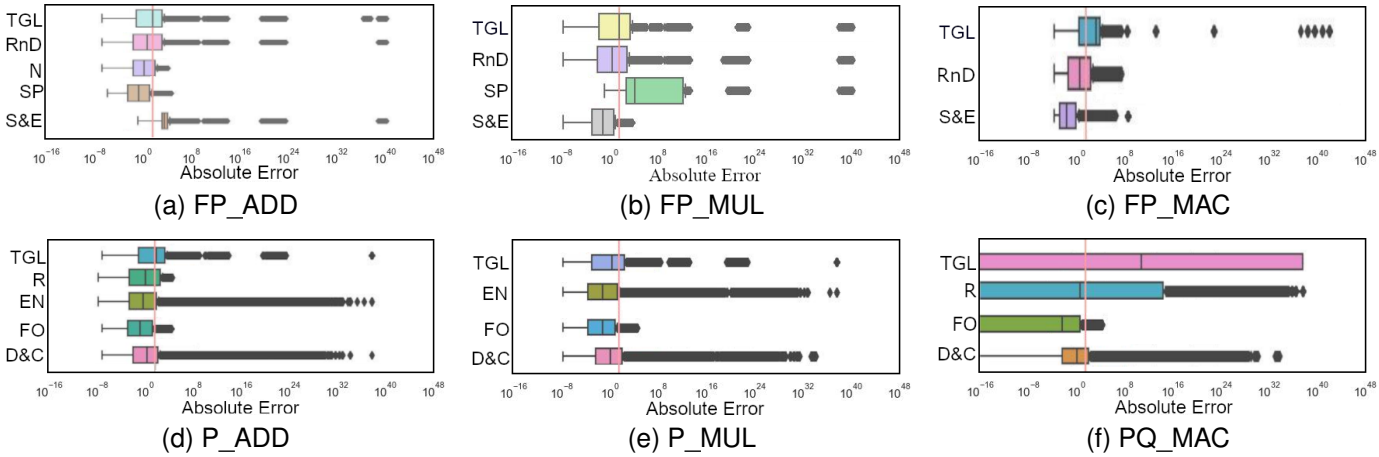

Fig. 5. Absolute error distribution in the structures of the **ADD** (*FP_ADD (a)* and *P_ADD (d)*), **MUL** (*FP_MUL (b)* and *P_MUL (e)*, and **MAC** cores (*FP_MAC (c)* and *PQ_MAC (f)*) for operations within the $\pm$10.0 range.

(i.e., large area cores comprise as many gates as fault sources).

Figure 4 depicts the *SFP* per core on both formats (lower magnitude means better relative resilience). The analysis indicates that Posit cores have lower sensitivity to fault propagation per area than FP cores (0.6X to 7.7X). Interestingly, Posit cores are larger in area than FP ones; see Table I. Thus, our findings suggest that Posit cores are more structurally resilient than FP ones for the analyzed ranges, with fewer fault sources prone to corrupt operations. Moreover, from the experiments, a bit-wise analysis of the corrupted outputs shows that a substantial percentage of single faults can be modeled as single bit-flips on corrupted results (55.2% in FP and 53.8% in Posit cores).

Our analyses aimed to characterize the fault propagation features of the core's architecture for both formats. However, further analyses are needed to identify critical impacts, e.g., large-magnitude errors, from those vulnerable structures.

The next subsection analyses critical error impacts on the outputs from faults in the core structures.

B. Analysis of Critical Error Impacts

1) Error impacts from fault-sensitive structures: We determine the *Error Rate* (ER) as the ratio between the number of observed output corruptions (*SDCs*) per fault (*j*) over the total number of evaluated stimuli (*N*) and faults per core (*SAFs*) for all vector ranges ($\pm$1.0, $\pm$10.0, and $\pm$100.0), see Equation 3.

$$ER = \frac{\sum_{j=1}^{SAFs}(SDCs)_j}{N * SAFs} \quad (3)$$

In particular, the ER results for the ADD, MUL, and MAC cores suffered minor deviations in all analyzed ranges. Then, we analyzed the vulnerability of the cores by considering the magnitude and distribution of the *Absolute Errors* (AE) in the entire operating range among the core's structures for the evaluated units. The AE evaluates, on a per-core basis, the absolute deviation between the fault-free result and the results obtained in the presence of hardware faults. Figure 5 depicts the AE distribution for the observed effects among the structures for two ADD, two MUL, and two MAC cores in both formats (one per format) within the $\pm$10.0 range. It must be noted that the full range of error effects was intentionally depicted to assess boundary corruption and outlier cases within the analyzed cores. According to the core's architecture, our findings indicate that several structures inside some processing steps in PF and Posit cores are highly sensitive and produce large-magnitude *(critical)* errors (up to $3.2\text{x}10^{38}$ and $1.3\text{x}10^{36}$ for FP and Posit cores, respectively), which are associated with corruption effects (out of the operative range) due to critical impacts on the binary representation of a result.

Our analyses indicate four main findings:

- Our results in Figure 5 indicate that faults in FP cores might cause large-magnitude errors up to two orders of magnitude larger than those in Posit cores for the evaluated operational ranges. This discrepancy can be attributed to the coding format of FP exponents and their associated vulnerable core structures.
- The core's organization plays a crucial role in its vulnerability to generate critical (large-magnitude) errors. The vulnerable structures on the cores for both formats differ in size and are placed across the processing steps, i.e., mostly near the input operands, while processing **S&E**, and **D&C**, and near the outputs, during **N**, **RnD**, **R**, **EN**, or **QA** processing. Furthermore, we identified logic substructures (**TGL**) in the critical data path that remain highly sensitive to affect computations massively. Notably, all highly fault-sensitive circuits compute the result's exponent (**S&E**) and the normalization (**N**) in FP cores and the regime and exponent fields for Posit cores (**D&C**), illustrated as the largest AEs in Figure 5.
- The ER results for all cores are moderate (from 17.6% to 27.9%), whether in FP or Posit formats. In particular, the ER in posit cores shows consistent levels for ADD (25.1%-26.3%), MUL (from 24.0% to 5.9%), and MAC (17.6% to 18.6%) cores, while in FP cores, the ER showed wide ranges for the evaluated ADD (18.8%-23.4%), MUL (20.7% to 26.1%), and MAC (23.1% to 27.9%) cores. Section VI-A showed that Posit cores have a lower structural fault probability per area than FP cores. However, ER results indicate that those few fault-sensitive circuits in Posit units are more prone to

generate errors (SDCs) than those in FP ones due to permanent faults (i.e., from 5.4% to around 7.5%) for the evaluated operational ranges.

- The `quire` Posit MAC core showed better overall error resilience than FP and `non-quire` Posit MAC cores (ER up to 10.3%). The `quire` *Posit* core is larger in area than FP ones, with mostly masked error effects from those large structures. Moreover, the absence of some highly fault-sensitive structures (e.g., **R**, and **EN**) seems to positively contribute to improving the unit's resilience. Nevertheless, the results for the `quire` *Posit* MAC reveal that few structures are fully resilient to propagate errors, see Figure 5.(f). Our findings indicate that errors on the *Quire accumulator* for extended accuracy (i.e., a 512-bit `quire` format uses 157 bits for accurate decimals) are highly prone to cause large-magnitude error corruptions.

2) Error distribution on the outputs: We analyze the error distributions from the evaluated stimuli vectors. For this purpose, we determine the frequency and magnitude of errors across the entire operational range for all cores. Figure 6 depicts the full range of observed AE and the accumulated error distribution for all corrupted operations for FP MUL (*in red*) and Posit MUL cores *(in blue)*.

While the AE and accumulated distributions are only depicted for MUL cores (i.e., ADD and MAC are not illustrated), the *Cumulative Distribution* curves for all cores identified a high concentration of errors below a magnitude of 1.0 for FP cores (58.5%, 87.5%, and 72.7% for ADD, MUL, and MAC, respectively), and Posit cores (60.4%, 83.9%, and 61.3% for ADD, MUL and MAC, respectively). These distributions suggest that the core's operation slightly influences the magnitude of error effects (i.e., low-magnitude errors in MUL cores are more frequent than in ADD and MAC units). Surprisingly, the progression of accumulated errors on all cores is noticeably equivalent and slightly differs due to the error dominance and their occurrence between small (mainly Posit, *"in blue"*) and large ranges (mainly FP, *"in red"*). Curiously, the cumulative error profile in `quire` Posit MAC core differs due to the extended 512-bit accumulator accuracy and removes the lower and higher errors by grouping them near 1.0. The findings also indicate that a considerable percentage of errors (58.5-87.5%) can be neglected since low-magnitude errors are usually masked by the implicit resilience in some applications, e.g., CNNs [18], [41]. On the other hand, large-magnitude errors can be considered as *critical* error impacts and are the main focus of the analysis.

A detailed examination of the errors and the corrupted bit-fields shows that the most frequently corrupted fields are the mantissa in FP and the fraction parts in Posit with limited error impacts ($<$1.0). However, the occurrence of critical errors is linked to the encoding formats of FP and Posit. Specifically, in FP cores, the error distribution is concentrated in distinct ranges (around 10^6, 10^{11}/10^{12}, 10^{20}, and 10^{38} in ADD, MUL, and MAC cores) due to corruptions observed in the exponent (bits 30-24). On the other hand, larger errors in Posit cores are widely distributed (ranging from 10^0 to 10^{30}), primarily stemming from corruptions observed in the exponent and dynamic regime fields (bits 30-26).

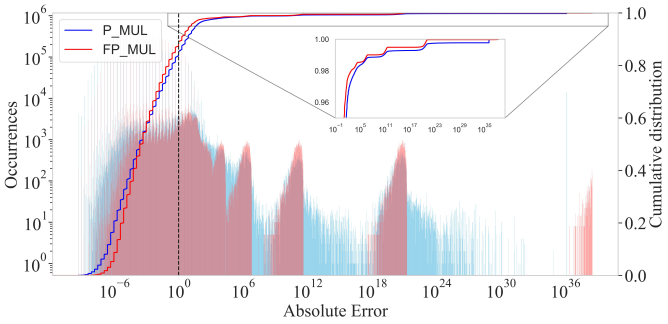

Fig. 6. Absolute error occurrences and cumulative distribution of errors for the MUL cores. The dotted black line depicts an absolute error of 1.0.

Therefore, our experimental results on the core structures allow us to affirm that a moderate percentage of single faults can produce large magnitude errors, representing critical effects, with those mostly associated with structures processing the exponent and regime fields. Instead, a significant percentage (about 57%) of the single faults produced low-magnitude error corruption effects.

In addition, we evaluate a software-only fault characterization approach to compare and emphasize the main differences and benefits of our structure-aware evaluation. In practice, the software evaluation uses the same input stimuli for the arithmetic operations. At the end, a total of $5.24x10^6$ (single bit-flip) errors are analyzed by exhaustively evaluating both formats by addressing all bits on the input operands and on the result. From the obtained AE results, we identified two main differences between the software and our approach: *i)* around 80% of the evaluated error in software produced similar AE's distribution effects for large magnitudes, including error concentrations on some ranges, but in smaller proportions than our approach. *ii)* Software-based evaluation yielded a significant portion of low-magnitude deviations (around 17% in FP and 19% in Posit cores) within the range of $1x10^{-36}$ to $1x10^{-7}$, which were not present under our proposed fault characterization method. Such effects indicate that single-effect analysis through software can lead to inaccuracies in the overall quantitative evaluation due to unrealistic errors, exacerbating the need for structure-aware analyses.

Our previous analyses, in subsections (VI-B1 and VI-B2), highlight the significant influence of the hardware structures and data formats on the distribution and magnitude of errors within faulty arithmetic cores. In particular, the evaluation of the architecture in arithmetic FP cores allows the identification of those few structures related to large-magnitude corruptions (associated with the exponent's computation) and allows the focused deployment of hardening mechanisms. In contrast, the wide range of vulnerable units in Posit cores interferes with efficiently deploying hardening mechanisms, indicating that more elaborated hardening mechanisms are required. We anticipate that clever mitigation solutions are required for Posit cores.

The following section explores the development of solutions to mitigate structural error susceptibility by resorting to selective hardening mechanisms on the cores of both formats.

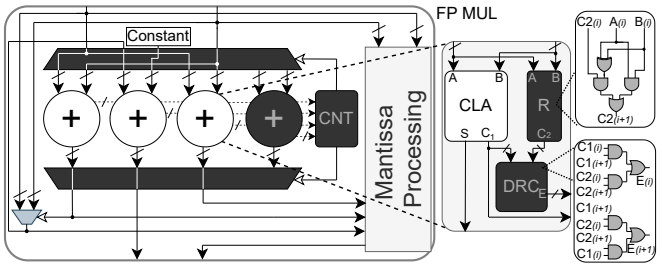

Fig. 7. A general scheme of the SC-R mechanism on the FP MUL core.

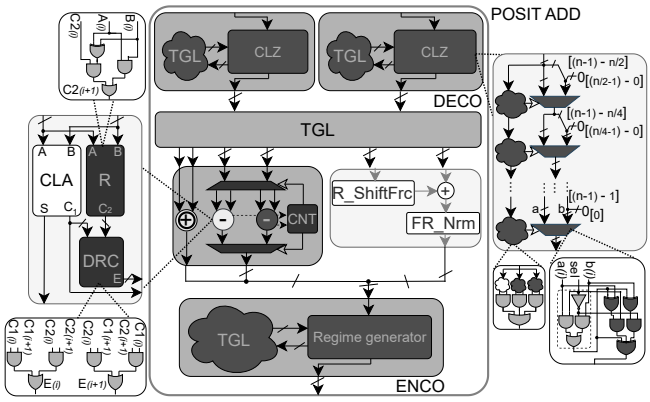

Fig. 8. A general scheme of the SC-R mechanism for an ADD Posit core.

VII. Selective hardening mechanisms for FP and Posit cores

This section describes the adoption of three coarse-grain selective hardening mechanisms to mitigate large-magnitude errors, which are critical to produce SDCs in complex applications, such as machine learning [42]. In detail, the mechanisms target the protection of those highly fault-vulnerable structures (prone to large-magnitude errors) in the cores of both formats. Furthermore, we assess, report, and analyze the overhead costs and the mechanism's effectiveness through focused evaluations (i.e., by checking the removal of large-magnitude errors).

Each hardening strategy uses hardware-based mechanisms to limit latency while effectively providing affordable computational performance on data-intensive operations, such as CNNs. In particular, the analysis of error distribution per core on both formats, see Figure 5, indicates that structures calculating the exponent and regime fields exhibited a strong fault sensitivity to cause large-magnitude errors and are the principal targets for selective hardening. The analyses revealed a high regularity in the core's vulnerable structures, mainly comprising binary adders, registers, *LZC*'s, and multiplexers, allowing the replication of one or more hardening mechanisms. In detail, we use a bottom-up strategy to develop and adopt custom mechanisms by firstly targeting structures locally and then extending them to the core. It is worth noting that the hardening mechanisms avoid structures causing low-magnitude errors since these are mostly masked at software or application levels [18].

A. Architecture of the selective hardening mechanisms

The first proposed mechanism (*Self-Check and Repair* or *'S-CR'*) combines parity-based fault detection [43] and self-repair circuits [44] to provide fault tolerance attributes on the targeted structures. First, we analyzed the vulnerable core structures, e.g., those in the critical path computing the result's exponent (for FP and Posit), pre-encoding the operands, or post-encoding the result (Posit), to identify regular units (*Basic Blocks* [44]). Then, we strategically instrument the core with self-checking units to trigger the mitigation mechanism through spare cores, multiplexers, and controllers.

To illustrate the mechanism adoption in the cores, Figure 7 depicts the *S-CR* hardening mechanism (in grey) for a *MUL* FP core. The *Basic Blocks* computing the result's exponent (binary adders) are grouped and adapted as *Carry-Look-ahead Adder* (CLA) to access the bit-wise carries. One or more parity circuits (R) independently compute the internal and the result parities, serving as the reference for fault detection. A *Dual Rail Checker* (DRC) circuit serves as a fault detection mechanism by finding mismatches between the CLA and R circuits. Indeed, DRC's encoding allows classification to discern between faults from CLAs or R circuits. Then, the DRC outputs and the CLA's (results) parity trigger one general controller (CNT), which, through one cold spare CLA and some multiplexers –placed on the CLA's inputs and outputs– handles the core's execution flow for selective self-repair/mitigation purposes (e.g., by replacing a faulty CLA with the spare one). Successively, the core is restarted to compute partially (e.g., *missing pipeline stage*) or completely a new operation. In addition, *S-CR* is complemented with *Triple Modular Redundancy* (TMR) schemes to protect multiplexers and other logic in the core's vulnerable data path. Despite that *S-CR* requires two additional clock cycles (one for detection and spare activation, and one for correction), the R and DRC circuits are the only ones permanently used for fault detection, while others are only enabled after being triggered.

In Posit cores, according to the vulnerable structure types, the *S-CR* mechanism combines three strategies: *1)* fine-grain repairing mechanisms on pre- and post-encoding structures (*Deco* and *Enco*), *2)* use of the *S-CR* mechanism to target regime and exponent logic, as described for FP cores, and *3)* TMR schemes to harden control and other data path logic.

In particular, the pre- and post-encoding structures comprise CLZs, binary counters, and multiplexers in cascade, which are hardened through bit-wise TMR structures. In addition, we adapted a bit-wise redundant structure [45] to cover faults in the multiplexers inside CLZs, as depicted in Figure 8. Furthermore, the regularity of the structures computing the exponent in FP and Posit cores allows the adoption of equivalent mechanisms through R, DRC, and CNT circuits. Regarding the regime computation structure (based on shifting circuits), we reuse the same redundant structure for multiplexers, while other logic in the data-path is protected through TMR schemes.

In addition, we implemented two redundancy-based hardening mechanisms: the first mechanism (*Dual Modular Redundancy* or DMR) instruments the core with copies of the vulnerable units. Then, one or more XOR-three arrays detect errors by comparing the units' outputs. When an error is detected, the redundant outputs are used to continue the core

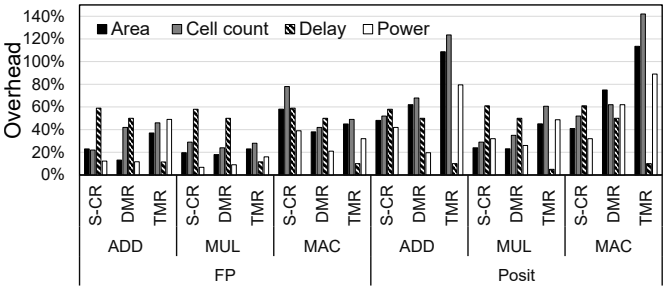

Fig. 9. Overhead costs for the *S-CR*, *DMR*, and *TMR* mechanisms on the *ADD*, *MUL* and *FMA* cores in both formats (FP and `non-quire` Posit).

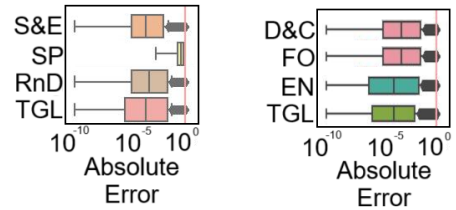

Fig. 10. AE distribution for the FP *(a)* and Posit *(b)* cores executing the MUL operation after hardening through the *S-CR* mechanism.

operation. The second mechanism uses TMR schemes and bit-wise voters to harden all vulnerable core structures.

B. Evaluation and validation of the hardening mechanisms

Figure 9 depicts the relative overhead costs (area, cells, performance delay, and power) of the three strategies (*S-CR*, *DMR*, and *TMR*) on the hardened cores at the gate level for both formats (FP and `non-quire` Posit). For the area analysis, we preserved the integrity of the hardening mechanism structures by avoiding enhanced optimization during synthesis. Moreover, power analyses considered 50% of switching activity.

Furthermore, a complementary analysis evaluates the overall resilience effectiveness and the reduction of large-magnitude errors of the hardening mechanisms in the cores through fault simulation campaigns, injecting one fault per simulation. Figure 10 illustrates the error distribution (AE) for hardened MUL cores in both formats through the *S-CR* mechanism.

In summary, we identify the following features for the evaluated hardening mechanisms on the cores for both formats:

- All strategies (*S-CR*, *DMR*, and *TMR*) are more area-costly in Posit cores than on FP ones (i.e., about 8-80% in FP, and around 6-142% in Posit) due to their architecture and the number of vulnerable structures to large-magnitude errors. The structures in the encoding stage (**EN**), and other units processing the exponent and regime fields in Posit cores account for around 23.3-54.9% of the core's area, while sensitive units in FP cores only represent 8-23% of the core's area.
- Our evaluation demonstrates that the core operation type (e.g., ADD or MUL) and the implicit vulnerability of their structures are associated with the area and cell overheads for both formats. Hardened MUL cores involved moderate area costs (18-25% in FP and about 24-45% in Posit), while ADD and MAC cores required additional costs, especially in Posit (15-57% in FP cores and around 40-142% for `non-quire` Posit ones).
- The power costs depend on the hardening mechanism and behave similarly on the cores of both formats. The cold spare units in the S-CR mechanism reduce the overall power (0.34X-4X when compared to TMR), while DMR and TMR schemes constantly use redundancy for each core execution. Interestingly, the power savings of the S-CR mechanism are limited when the number of units to harden is large (see FP MAC core). The several full-adder cells inside MAC cores requires the permanent use of self-checking testing circuits and increase the power to equivalent levels of the TMR strategy (about 35-40%).
- The delay costs are determined by the specific fault-mitigation strategies used in each mechanism. In DWC and S-CR, error handling relies on state-machine control and the execution of additional cycles to activate the mitigation structures once a fault is detected, resulting in moderate overheads of approximately 50–60% across all evaluated cores and formats. Instead, TMR employs combinational voter logic that operates continuously during normal execution, introducing only minimal delay impacts (below 15%), which can often be accommodated within the available timing slack of the core.
- The results indicate that selective-hardening mechanisms are feasible solutions to mitigate large-magnitude errors caused by exponent and regime corruptions, as reported in Figure 10 for MUL cores and all others (not shown) for both formats FP and `non-quire` Posit.

During the mechanism's evaluation, we identified sets of highly sensitive sources causing moderate-magnitude errors (from 10^0 to 10^2). Most sources in FP cores are related to the sign logic and gates/flip-flops between the exponent and mantissa processing units (e.g., 16 fault sources in the MUL core), which are easily protected through TMR schemes. Similarly, sensitive sources in Posit cores are mostly associated with gates/flip-flops between the exponent and mantissa processing units (e.g., 1,319 in the MUL core). However, we also found sources related to fraction computing sub-units (e.g., 233 in the MUL core). For fault mitigation completeness, we harden those sensitive structures using TMR schemes. It is worth noting that the structural complexity in Posit cores might require more sophisticated hardening mechanisms.

Our analyses and the hardening mechanisms are intended to support the development of robust designs. Moreover, these analyses can also provide accurate error models for reliability evaluations at the system level, emphasizing the importance of incorporating reliability as a complementary parameter in system development. The results suggest that selective hardening strategies serve as a solution to increasing the system's reliability when robustness and low error impacts are crucial application factors. While our selective hardening mechanisms focused mainly on mitigating permanent faults arising in the arithmetic cores, these mechanisms can be effectively used to mitigate other fault models, such as transient faults.

Finally, our evaluation indicates that Posit cores are more structurally resilient than FP cores independently of the

error magnitude. However, mitigating vulnerable structures in Posit hardware requires more sophisticated strategies or a combination of multiple approaches. Since FP cores are technologically mature and are more hardware-efficient than Posit ones, this is also observed in the overhead cost of their hardening mechanisms. Similarly, our results demonstrate that improvements in the design of posit hardware might reduce the fault sensitivity of some structures (e.g., **Decoding and checking** or **Encoding**) while limiting mitigation mechanisms' overhead costs.

VIII. Discussion

This work analyzed the impact of hardware faults in the structures of arithmetic cores for three fundamental operations on FP and Posit number formats (ADD, MUL, and MAC).

A fine-grain characterization of the arithmetic cores identified their most fault-sensitive structures, indicating that up to 76% and 89% of single effects in the structures are prone to causing error corruption. Next, this work analyzed and identified the structural sources of large-magnitude error effects, showing that a considerable percentage of faults ranging from 12.5% to 41.5% in FP cores and from about 16.1% to 39.6% in Posit cores might cause effects in the order of $1x10^{38}$ and $1x10^{36}$ on FP and Posit units, respectively. Our findings indicate that faulty FP cores might cause corruption effects up to two orders of magnitude larger than in Posit ones. In detail, hardware structures associated with the exponent and the regime fields for FP and Posit are directly responsible for causing such critical effects. Although our experiments focused on 32-bit size units, clearly similar fault vulnerabilities are present in compacted or custom variations of the formats, since these highly vulnerable structures are also included in the hardware for such compacted formats. The preservation of the same structures allows us to indicate that equivalent hardware susceptibilities can be found on reduced-size hardware units. However, a comprehensive analysis of compacted or custom hardware cores for both formats is still required and planned to be addressed in future work.

Furthermore, this work adopted and evaluated three selective-hardening strategies to mitigate single effects in the arithmetic cores, showing that large-magnitude error effects can be effectively removed with moderate impacts on area overhead (ranging from 20% to 112%). In particular, our mechanisms were effective for `non-quire` Posit cores due to their regular structures. Unfortunately, the wide variety of highly sensitive structures in the `quire` Posit MAC core impeded the effective adaptation of the mechanisms under reasonable costs since these structures require spare units and controllers with null cost savings. In fact, the implementation excessively increased the overhead on `quire` MAC (e.g., TMR required more than 153% additional area, which is more costly than the combined area of the `non-quire` ADD and MUL cores), indicating the need for more elaborate and alternative hardening strategies for `quire` cores.

It is worth noting that this work focused on the analysis and hardening of arithmetic units for FP and Posit formats. The reported results are intended to guide and contribute to the evaluation of complete applications, such as CNNs, through complementary strategies, such as cross-layer methods, but such evaluations are out of the scope of the current work.

IX. Conclusions and Future works

This work introduced a strategy to evaluate the fine-grain structural impact of permanent faults on the reliability of several arithmetic cores supporting the FP and Posit formats. Our findings indicate that Posit cores are more structurally resilient than FP ones (12% to 94% lower structural fault propagation per area unit). However, the most fault-vulnerable structures in Posit cores might cause broader error distributions and are more susceptible to significant error propagation than those in FP cores (from 5.2% up to 7.5%). In contrast, faults in FP cores can produce errors up to two orders larger in size than those observed on Posit cores.

Furthermore, this work explored hardware-based selective hardening mechanisms to remove critical effects from faults in arithmetic cores for both formats. Our analyses indicate that large-magnitude errors (e.g., > 1.0) can be effectively removed with acceptable area overhead costs (from 20% to 110%).

In the future, we plan to extend our evaluations into wider input dynamic ranges to address more recent applications, such as transformer-based workloads, as well as evaluating/developing mitigation solutions for hardware cores using emerging and relevant formats for Artificial Intelligence.

Acknowledgments

This work has been supported by the National Resilience and Recovery Plan (PNRR) through the National Center for HPC, Big Data and Quantum Computing (ICSC), and through the Innovation Grant at Spoke 1 - FUTURE HPC & BIG DATA: project MeDeHa.

References

[1] Z. Ji, H. Chen, and X. Li, "Design for reliability with the advanced integrated circuit (ic) technology: challenges and opportunities," *Science China Information Sciences*, vol. 62, pp. 1–4, 2019.

[2] L. Xing, "Reliability in internet of things: current status and future perspectives," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 6704–6721, 2020.

[3] D. Goldberg, "What every computer scientist should know about floating-point arithmetic," *ACM Comput. Surv.*, vol. 23, no. 1, p. 5–48, 1991.

[4] A. J. Strojwas, K. Doong, and D. Ciplickas, "Yield and reliability challenges at 7nm and below," in *2019 Electron Devices Technology and Manufacturing Conference (EDTM)*, 2019, pp. 179–181.

[5] IEEE, "The international roadmap for devices and systems: 2022," in *Institute of Electrical and Electronics Engineers (IEEE)*, 2022.

[6] J. Rajski, V. Chickermane, J.-F. Côté, S. Eggersglüß, N. Mukherjee, and J. Tyszer, "The future of design for test and silicon lifecycle management," *IEEE Design Test*, pp. 1–1, 2023.

[7] J.-W. Han, M. Meyyappan, and J. Kim, "Single event hard error due to terrestrial radiation," in *Int. Reliability Physics Symp.(IRPS)*, 2021.

[8] H. D. Dixit, S. Pendharkar, M. Beadon, C. Mason, T. Chakravarthy, B. Muthiah, and S. Sankar, "Silent data corruptions at scale," 2021. [Online]. Available: https://arxiv.org/abs/2102.11245

[9] J. Athavale, A. Baldovin, R. Graefe, M. Paulitsch, and R. Rosales, "Ai and reliability trends in safety-critical autonomous systems on ground and air," in *Ann. IEEE/IFIP Int. Conf. on Dependable Systems and Networks Workshops (DSN-W)*, 2020, pp. 74–77.

[10] J. L. Gustafson and I. T. Yonemoto, "Beating floating point at its own game: posit arithmetic," *Supercomput. front. innov.*, vol. 4, no. 2, 2017.

[11] R. Murillo, A. A. Del Barrio, and G. Botella, "Customized posit adders and multipliers using the flopoco core generator," in *IEEE Int. Symp. on Circuits and Systems (ISCAS)*, 2020, pp. 1–5.

[12] F. de Dinechin, L. Forget, J.-M. Muller, and Y. Uguen, "Posits: the good, the bad and the ugly," in *Conf. for Next Generation Arithmetic (CoNGA)*, 2019.

[13] I. Alouani, A. B. Khalifa, F. Merchant, and R. Leupers, "An investigation on inherent robustness of posit data representation," in *34th Int. Conf. VLSI Design & 20th Int. Conf. on Embedded Systems (VLSID)*, 2021.

[14] B. Schlueter, J. Calhoun, and A. Poulos, "Evaluating the resiliency of posits for scientific computing," in *Workshops of The Int. Conf. High Perform. Comput. Netw. Storage Anal. (SC 23)*, 2023, p. 477–487.

[15] G. Gavarini, A. Ruospo, and E. Sanchez, "On the resilience of representative and novel data formats in cnns," in *Int. Symp. on Defect and Fault Tolerance in VLSI and Nanotechnology Systems*, 2023, pp. 1–6.

[16] G. Papadimitriou and D. Gizopoulos, "Demystifying the system vulnerability stack: Transient fault effects across the layers," in *ACM/IEEE 48th Ann. Int. Symp. on Computer Architecture (ISCA)*, 2021, pp. 902–915.

[17] J. E. R. Condia, J.-D. Guerrero-Balaguera, R. Limas Sierra, and M. Sonza Reorda, "Analyzing the structural and operational impact of faults in floating-point and posit arithmetic cores for cnn operations," in *29th IEEE European Test Symp. (ETS)*, 2024, pp. 1–4.

[18] G. Li, S. K. Sastry, 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 *Int. Conf. High Perform. Comput. Netw. Storage Anal. (SC'17)*, 2017.

[19] S. Boldo, C.-P. Jeannerod, G. Melquiond, and J.-M. Muller, "Floating-point arithmetic," *Acta Numerica*, vol. 32, p. 203–290, 2023.

[20] V. Gohil, S. Walia, J. Mekie, and M. Awasthi, "Fixed-posit: A floating-point representation for error-resilient applications," *IEEE Trans. Circuits Syst. II: Express Br.*, vol. 68, no. 10, pp. 3341–3345, 2021.

[21] J. Elliott, F. Mueller, F. Stoyanov, and C. Webster, "Quantifying the impact of single bit flips on floating point arithmetic," North Carolina State University. Dept. of Computer Science, Tech. Rep., 2013.

[22] S. H. Fatemi Langroudi, T. Pandit, and D. Kudithipudi, "Deep learning inference on embedded devices: Fixed-point vs posit," in *1st Workshop on Energy Efficient Machine Learning and Cognitive Computing for Embedded Applications (EMC2)*, 2018, pp. 19–23.

[23] P. Lindstrom, S. Lloyd, and J. Hittinger, "Universal coding of the reals: Alternatives to ieee floating point," in *Proc. of Int. Conf. for Next Generation Arithmetic (CoNGA '18)*, 2018.

[24] H. De Silva, H. Tan, N.-M. Ho, J. L. Gustafson, and W.-F. Wong, "Towards a better 16-bit number representation for training neural networks," in *Int. Conf. for Next Generation Arithmetic (CoNGA)*, 2023, p. 114–133.

[25] R. Murillo, J. Hormigo, A. Del Barrio, and G. Botella, "Hub meets posit: Arithmetic units implementation," *IEEE Trans. Circuits Syst. II: Express Br.*, vol. 71, no. 1, pp. 440–444, 2024.

[26] F. Xiao, F. Liang, B. Wu, J. Liang, S. Cheng, and G. Zhang, "Posit arithmetic hardware implementations with the minimum cost divider and squareroot," *Electronics*, vol. 9, no. 10, 2020.

[27] A. Poulos, S. A. McKee, and J. C. Calhoun, "Posits and the state of numerical representations in the age of exascale and edge computing," *Software: Practice and Experience*, vol. 52, no. 2, pp. 619–635, 2022.

[28] R. Limas Sierra, J. Guerrero-Balaguera, J. E. R. Condia, and M. Sonza Reorda, "Exploring hardware fault impacts on different real number representations of the structural resilience of tcus in gpus," *Electronics*, vol. 13, no. 3, 2024.

[29] J. Sun, J. Huang, and M. Snir, "Pinpointing crash-consistency bugs in the hpc i/o stack: A cross-layer approach," in *Int. Conf. High Perform. Comput. Netw. Storage Anal. (SC 21)*, 2021.

[30] W. Li, C. Nigh, D. Duvalsaint, S. Mitra, and R. Blanton, "Pepr: Pseudo-exhaustive physically-aware region testing," in *IEEE Int. Test Conf. (TC)*, 2022, pp. 314–323.

[31] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, 2004.

[32] I. Koren, *Computer arithmetic algorithms*. CRC Press, 2018.

[33] "Ieee standard for floating-point arithmetic," *IEEE Std 754-2019 (Revision of IEEE 754-2008)*, pp. 1–84, 2019.

[34] D. Mallasen, R. Murillo, A. Barrio, G. Botella, L. Pinuel, and M. Prieto-Matias, "Percival: Open-source posit risc-v core with quire capability," *IEEE Trans. Emerg. Top. Comput.*, vol. 10, no. 03, pp. 1241–1252, 2022.

[35] R. Murillo, D. Mallasén, A. A. Del Barrio, and G. Botella, "Energy-efficient mac units for fused posit arithmetic," in *IEEE Int. Conf. on Computer Design (ICCD)*, 2021, pp. 138–145.

[36] F. J. H. Santiago, H. Jiang, H. Amrouch, A. Gerstlauer, L. Liu, and J. Han, "Characterizing approximate adders and multipliers for mitigating aging and temperature degradations," *IEEE Trans. Circuits Syst. I: Regul. Pap.*, vol. 69, no. 11, pp. 4558–4571, 2022.

[37] J. E. R. Condia, B. Du, M. Sonza Reorda, and L. Sterpone, "Flex-gripplus: An improved gpgpu model to support reliability analysis," *Microelectron. Reliab.*, vol. 109, p. 113660, 2020.

[38] F. de Dinechin and B. Pasca, "Designing custom arithmetic data paths with flopoco," *IEEE Des. Test Comput.*, vol. 28, no. 4, pp. 18–27, 2011.

[39] M. Martins, J. M. Matos, R. P. Ribas, A. Reis, G. Schlinker, L. Rech, and J. Michelsen, "Open cell library in 15nm freepdk technology." *Proc. of Int. Symp. on Physical Design (ISPD '15)*, 2015, pp. 171-178.

[40] J. D. Guerrero Balaguera, J. E. R. Condia, F. F. Dos Santos, M. Sonza Reorda, and P. Rech, "Understanding the effects of permanent faults in gpu's parallelism management and control units," in *Int. Conf. High Perform. Comput. Netw. Storage Anal. (SC'23)*, 2023, pp. 1–10.

[41] L. Ping, J. Tan, and K. Yan, "Sern: Modeling and analyzing the soft error reliability of convolutional neural networks," in *Proc. Great Lakes Symp. on VLSI (GLSVLSI '20)*, 2020, p. 445–450.

[42] 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, 2019.

[43] M. Nicolaidis, "Carry checking/parity prediction adders and alus," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 11, no. 1, pp. 121–128, 2003.

[44] T. Koal, H. T. Vierhaus, and D. Scheit, "A concept for logic self repair," in *12th Euromicro Conf. on Digital System Design, Architectures, Methods and Tools*, 2009, pp. 621–624.

[45] A. K. Neelam and S. Musala, "Real-time self repairable multiplexer for fault tolerant systems," in *Int. Conf. on Communication and Signal Processing (ICCSP)*, 2020, pp. 1124–1127.

Josie E. Rodriguez Condia received the M.Sc. degree in electronics engineering from Universidad Pedagogica y Tecnologica de Colombia (UPTC), Sogamoso, Colombia, in 2017 and the Ph.D. degree in Computer Engineering from Politecnico di Torino, Turin, Italy, in 2021. He is now an Assistant professor at the same institution. His research interests include functional testing, parallel architectures, Graphics Processing Units, and embedded system design. He is an IEEE member.

Juan-David Guerrero-Balaguera received a Ph.D. degree in Computer Engineering from Politecnico di Torino, Turin, Italy, in 2024, and he is now a researcher at the Department of Control and Computer Engineering. His research interests include functional tests, artificial intelligence, and parallel architectures. Guerrero-Balaguera has an M.Sc. in electronics from the Universidad Pedagógica y Tecnológica de Colombia (UPTC), Sogamoso, Colombia. He is an IEEE member.

Robert Limas Sierra received the B.S. and M.Sc. degrees in electronic engineering from the Universidad Pedagógica y Tecnológica de Colombia (UPTC), Colombia, in 2019 and 2022, respectively. He is currently pursuing a Ph.D. degree with the Department of Control and Computer Engineering, Politecnico di Torino, Turin, Italy. His research interests include system functional testing, parallel architectures, GPGPUs, and hardware accelerators. He is an IEEE student member.

Matteo Sonza Reorda received an M.Sc. degree in electronics and a Ph.D. degree in Computer Engineering from Politecnico di Torino, Italy, in 1986 and 1990, respectively, and is currently a full professor at the Department of Control and Computer Engineering. He published more than 400 papers in the area of test and fault-tolerant design of reliable circuits and systems, receiving several Best Paper Awards at major international conferences. He is involved in research projects with companies and research centers worldwide. He is an IEEE Fellow.