PC-Posits: Enhanced Soft Error Resilience of Posit Arithmetic Through Analytical Modeling

Vishesh Mishra, Marcello Traiola, Angeliki Kritikakou, Fernando Fernandes dos Santos, Urbi Chatterjee

PC-Posits: Enhanced Soft Error Resilience of Posit Arithmetic Through Analytical Modeling

Vishesh Mishra*, Marcello Traiola+, Angeliki Kritikakou+, Fernando Fernandes dos Santos+ and Urbi Chatterjee*
*IIT Kanpur, India - 208016
+Univ Rennes, CNRS, Inria, IRISA - UMR 6074, F-35000 Rennes, France

Abstract

The Posit number system was introduced to overcome the limitations of traditional real-number representations. It provides a representation that achieves higher near-unity precision and a wider dynamic range compared to standard numerical formats. Beyond numerical accuracy, prior studies have shown that certain Posit configurations exhibit notable resilience to Soft Errors (SEs). However, this resilience varies across different Posit configurations, and the underlying causes remain poorly understood. To address these limitations, we propose the first analytical framework for characterizing SE resilience in Posits. Our framework introduces the *Expected Catastrophic Error* and *Exception Probability* metrics to quantify severe deviations and the likelihood of anomalies under SEs, respectively. Together, our metrics explain the varying SE resilience across Posit configurations. We observe that Posit inherently self-compensates against SEs, but this phenomenon weakens when the regime grows too large, leaving insufficient bits to fully encode the exponent. Building on these insights, we propose SE resilient *Proactively-Clipped Posits (PC-Posits)* that enforces "proactive clipping" while preserving a dynamic range and precision adequate for real-world applications. To evaluate *PC-Posits*, we perform memory fault injections across heterogeneous benchmarks. *PC-Posits* deliver up to 29.1 percent point (pp) higher SE resilience than standard Posits. Moreover, the proactive clipping enables simpler hardware and reduces energy consumption by up to 47.2% compared to standard Posits. Finally, *PC-Posits* offer up to 33.9 pp higher SE resilience compared to IEEE Floating-point (FP) formats.

1 Introduction

Traditional numerical representations face trade-offs between precision and dynamic range. Posits address this challenge with a tapered precision design, concentrating accuracy where it is most useful and expanding the representable range [1]. Unlike formats that use fixed-length fields for the sign, exponent, and fraction (e.g., IEEE FP), Posits employ variable-length fields. In Posits, the field length depends upon the number being represented [1]. This structural flexibility allow Posits to offer higher precision near unity and larger dynamic range for a given bit-width. As a result, Posits better capture the statistical characteristics of values encountered in real-world applications. On top, Posits have only a single exception, i.e., *Not-a-Real (NaR)* [1], [2], unlike FP formats with multiple exceptions, i.e., *overflow, underflow*, and *Not-a-Number (NaN)*.

Beyond numerical accuracy, Posits have demonstrated inherent resilience to SEs [3]–[7]. SEs are induced by transient events from environmental factors, such as cosmic radiation, voltage, and temperature fluctuations [8]–[12]. Prior studies have shown that Posits SE resilience strongly depends on the Posit configuration. While some Posits configurations deliver a higher reliability against SEs compared to FP formats [3]–[5], [7], others perform worse [3], [5]. However, existing works have not identified the root causes of this variability, and no systematic methodology exists to enhance the SE resilience of Posit formats. To address these limitations, we pose two fundamental research questions: ***Q1****. What underlying factors govern the soft-error resilience of different Posit configurations, and what mechanisms account for their failure modes?* ***Q2****. How can the impact of soft errors on Posits be systematically reduced?*

In this paper, we address *Q1* and *Q2* using analytical approach. To achieve that, we initially propose a framework to characterize the SE resilience of the Posit format. We identify the reasons for SE resilience of common Posit configurations and the root causes of failures. We show that the variable-length fields of Posits possess a self-compensating mechanism to mitigate the impact of SEs. However, its effectiveness depends on the length variability of the Posit fields and may be compromised in configurations where the fields have high variability. To analyze the efficacy of the self-compensating error mechanism, we propose two probabilistic and statistical metrics appropriate for Posits, namely *Expected Catastrophic Error* and *Exception Probability*. Note that existing conventional reliability metrics capture only error statistics and are thus unsuitable for the variable-length field existing in Posits representation.

Furthermore, building on our analytical framework, we propose the *Proactively-Clipped Posit (PC-Posit)* format that enhances SE resilience. The key insight is that *clipping* the maximum regime length, which represents the large-scale magnitude of the Posit representation, can effectively reduce fault propagation. Note that clipping strategies in signal processing, machine learning, and high-performance computing (HPC), such as gradient, amplitude, or value clipping, help mitigate numerical instability. However, they are inherently reactive, as they rely on profiling and/or online monitoring [13]–[15]. In contrast, our approach leverages Posits' structural properties and, combined with insights from our analytical framework, it proactively bounds regime growth, reducing the sensitivity to single and multi-bit SEs. The proposed *PC-Posits* maintain the precision and flexibility of conventional Posits by limiting, rather than fixing, the regime length. Unlike Fixed-Posits [16], which constrain the regime to a single value, *PC-Posits* impose a cap on the maximum valid regime. As a result, *PC-Posits* ensure resilience, adequate dynamic range, and precision, being able to represent data across a wide range of applications. Furthermore, *PC-Posits* intrinsically simplify the hardware design compared to standard Posits, due to their nature, independently of specific hardware implementation. In summary, our main contributions are:

- An analytical framework to characterize the SE resilience of Posits. To achieve that, we propose two metrics: *Expected Catastrophic Error* to trace catastrophic errors, and *Exception Probability* to denotes the exception triggering induced by bit-flips.

• A novel Posits variant, *Proactively-Clipped Posits (PC-Posits)*, enabled by our analytical framework to improve SE resilience. *PC-Posits* also simplifies hardware implementation, reducing energy consumption by up to 47.2% compared to standard Posits.
• An extensive set of fault injection experiments across five heterogeneous benchmarks. We compare *PC-Posits* with standard Posits and commonly use formats (FP8, FP16, BF16, FP32, INT8). Results demonstrate that *PC-Posits* achieve up to 33.9 pp higher resilience than FP formats and 29.1 pp higher resilience than standard Posits.

2 Background and Related Work

Posit Number System: The Posit number system, introduced as an alternative to IEEE-754, employs a variable-length encoding scheme to achieve higher precision near unity and graceful degradation at the extremes of its dynamic range [1]. As Fig. 1 depicts, the exponent field of an N-bit Posit number has a fixed *maximum* length, determined by the parameter exponent size (es), while the regime field is variable and defined through run-length encoding, which can grow up to $(N-1)$ bits, leaving no space for exponent and fraction. This regime length variability allows representing numbers across a wide dynamic range. A number x in the *Posit(N, es)* format (N total bits and es exponent bits) is represented as

$$x = (-1)^s \times (useed)^k \times 2^e \times (1+f) \quad (1)$$

where $s \in \{0,1\}$ is the *sign* bit, useed $= 2^{2^{es}}$ is a *scaling* factor, $k \in \mathbb{Z}$ is the *regime*, e is the unsigned integer *exponent* ($0 \le e \le 2^{es}-1$), and f is the *fractional* part of the significant. The regime k encodes the number's large-scale magnitude and consists of r identical bits (called run-bits) followed by a terminating bit of the opposite value, with a total length of $r+1$. The decoded regime value k is computed as $k = r-1$, if regime has run-bits of 1s followed by a 0, and as $k = -r$, if regime has run-bits of 0s followed by a 1. For example $k = 001 = -2$, $k = 110 = 1$, and $k = 1110 = 2$.

SE impact on Posits arithmetic: Empirical analysis of Posit formats' resilience has shown that they exhibit significantly higher inherent fault tolerance in machine-learning applications [3] and in scientific computing workloads [4] than FP. This inherent average tolerance has also been quantified empirically by assessing the effects of permanent faults in adders and multipliers for Posit arithmetic [5], [6]. Furthermore, a comprehensive study of the SEs' impact on DNNs has reported that a 32-bit Posit-based DNN achieves up to 15% higher classification accuracy than its IEEE-754 counterpart under Single Event Upsets [7]. Moreover, Posit resilience is dependent on its configuration: some configurations effectively mitigate accuracy loss, whereas others perform worse [7]. This is partly explained by recent work findings, showing that the regime and the exponent fields, which regulate the magnitude of Posit numbers, are more vulnerable than the other fields [6]. Despite these valuable observations, existing research lacks a comprehensive analytical framework to characterize Posits' resilience. Using it, we reveal that the general Posit resilience stems from the regime's logarithmic encoding that distributes dynamic range non-uniformly, reducing the likelihood of catastrophic errors.

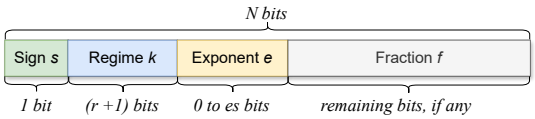

Figure 1. Posit (N, es) Representation with (r+1) regime bits

In Section 3, we observe that Posits have the inherent property of mitigating soft error effects, provided that the regime field leaves enough bits to encode the exponent and the exponent size is not too large (< 5 bits). These insights answer ***Q1***.
Furthermore, existing methodologies for protecting Posit-based computations against soft errors rely on error-detecting and error-correcting hardware structures. Examples include data parity or Error Correcting Codes (ECCs), time redundancy and spatial redundancy as investigated in [6]. However, existing methodologies are intrinsically ***reactive***. Section 4 presents Proactively-Clipped Posits (PC-Posits), a new Posit format that ***proactively*** reduces the impact of potential soft errors, thereby answering ***Q2***. PC-Posits are orthogonal to reactive approaches and are not intended to replace them. Rather, existing techniques can be applied on top of PC-Posits when additional protection is required.

3 Proposed Analytical Framework

We propose an analytical framework that quantifies the impact of SEs on Posits. Our analysis focuses on the resilience of the Posit format itself, which can be assimilated to *studying the resiliency to faults in the data memory*. Posits' regime has a variable length (defined by the represented value), which affects the position and length of the exponent and fraction. This makes the effect of a bit-flip *value-dependent*, and thus, inherently probabilistic. To incorporate this probabilistic and position-dependent behavior, we introduce two metrics that operate on expected values, i.e., (i) *Expected Catastrophic Error* and (ii) *Exception Probability* as discussed in Section 3.1 and Section 3.2, respectively. An important observation, which *enables our framework to also work with multiple bit-flips*, is that the leftmost flipped bit dominates the analysis in Posits. Indeed, if a fault occurs in the regime, it alters the decoding mechanism, so any subsequent flip cannot further corrupt the large-scale magnitude. Consequently, our metrics remain applicable to multi-bit-induced catastrophic errors. For clarity reasons, and without loss of generality, we present the analysis for a 32-bit representation ($N = 32$).

3.1 Expected Catastrophic Error

Our first metric captures catastrophic errors due to bit-flip(s) in the regime or exponent bits. As flipping the fraction bits does not affect the overall magnitude of the number and flipping the sign bit only doubles the error magnitude without causing catastrophic errors, both scenarios are excluded from this analysis. Assuming the bit-flip does not result in NaR or zero, the possible cases of regime bit-flips and/or exponent bit-flips are described below. Each *Case x* has two sub-cases. *Case x.1* accounts for regime run-bits of 1s, and *Case x.2* for regime run-bits of 0s. Fig. 2 illustrates examples for faults occurring on the *Cases x.1. Case x.2* are symmetric.
Case 1 - Single bit-flip in the run-bits of regime: This fault decreases the regime length. In the example of *Case 1.1* of Fig. 2, a bit-flip at the 4^{th} run-bit of the regime reduces the regime length from 8 to 5 bits, modifying the original number $1.78*10^7$ to $3.3*10^4$, i.e., a change of $\approx$ 3 orders of magnitude. Theorem 1 formalizes Cases 1.1 and 1.2 and derives their expected catastrophic errors.
Case 2 - Single bit-flip in the terminating bit of regime (increase by 1): This case leads to an increase in the regime length by one bit. It occurs when the terminating bit of the regime and the bit immediately following have the same value. Hence, when the terminating bit flips, the next bit will terminate the regime. In the example of *Case 2.1* of Fig. 2, a bit-flip occurring in the terminating bit increases the regime length from 8 to 9 bits, causing a change

of magnitude of more than one order. Theorem 2 formalizes Cases 2.1 and 2.2 and derives their expected catastrophic errors.

Case 3 - Single bit-flip in the terminating bit of regime (increase by more than 1): This case increases the regime length by more than 1 bit. It occurs when the terminating bit of the regime and the bit immediately following it have different values. Thus, when the terminating bit flips, the subsequent bit cannot terminate the regime, so the regime increases by at least 2 bits. In the example of *Case 3.1* in Fig. 2, a bit-flip at the terminating bit extends the regime from 8 to 11 bits, leading to a change of almost 3 orders of magnitude. Theorem 3 formalizes Cases 3.1 and 3.2 and derives their expected catastrophic errors.

Case 4 - Single bit-flip in the exponent: This case does not alter the length of the Posit fields but alters the exponent value, causing changes of approximately an order of magnitude. Theorem 4 formalizes this case, for general errors in the exponent bits.

Multiple bit-flips: When multiple bit-flips occur in the same Posit number, it always falls in one of the above cases. For faults causing catastrophic errors, three sub-cases are possible:

1. *All bit-flips in the regime*: The leftmost bit-flip would reduce the length of the regime (case 1, Theorem 1), and the others would be either in the exponent (Theorem 4) or in the fraction, which does not contribute to catastrophic errors.
2. *A part of the bit-flips in the regime and the rest outside the regime*: If the leftmost bit-flip is not the termination bit of the regime, we fall back in the case 5a; if the leftmost bit-flip is the termination bit, we fall either in case 2 (Theorem 2) or 3 (Theorem 3), possibly with some bit-flips in the (new) exponent (case 4, Theorem 4), or in the fraction, which does not contribute to catastrophic errors.
3. *All bit-flips in exponent or elsewhere*: It corresponds to case 4, possibly with bit-flips in bits not causing catastrophic errors.

Expected Catastrophic Error metric. To quantify catastrophic errors for the aforementioned cases, we introduce the *Expected Catastrophic Error* (η) metric defined as:

$$\eta = \mathbb{E}\Big[\Big|\log_2|x_o| - \log_2|x_f|\Big|\Big], \tag{2}$$

where x_o is the original value and x_f is the faulty value after bit-flip(s). The original *Posit(N, es)* number x_o has an absolute value of $|x_o| = (useed)^{k_o} \times 2^{e_o} \times (1 + f_o) = 2^{(2^{es})k_o + e_o}(1 + f_o)$ (see Eq. 1). Bit-flip(s) impacting the regime and/or exponent bits transform this number into $|x_f| = 2^{((2^{es})k_f + e_f)}(1 + f_f)$, where k_f, e_f and f_f are the potentially modified posit fields. Accordingly, the absolute

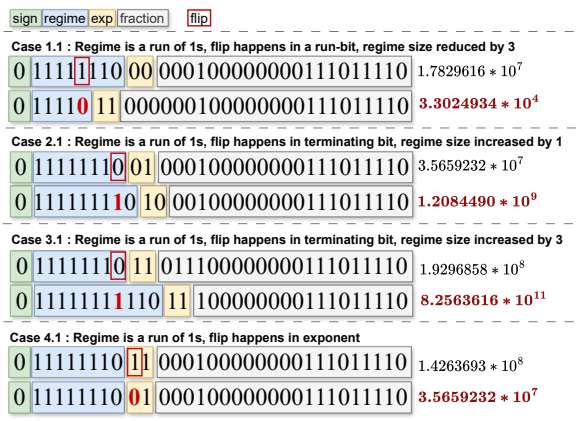

Figure 2. Example of bit-flip(s) in Posit regime or exponent

logarithmic error due to bit-flip(s) for regime and/or exponent bits is: $\Big|\log_2|x_o| - \log_2|x_f|\Big| \approx \Big|2^{es}\left(k_o - k_f\right) + \left(e_o - e_f\right)\Big|$. Thus, the *Expected Catastrophic Error* (η) can be redefined as follows:

$$\eta \approx 2^{es}\mathbb{E}[|k_o - k_f|] + \mathbb{E}[|e_o - e_f|]. \tag{3}$$

To compute η, $\mathbb{E}[|k_o - k_f|]$ is obtained from the contributions of each case formalized through Theorems 1–3, and $\mathbb{E}[|e_o - e_f|]$ from Theorem 4. Definition 1 provides the groundwork for these proofs.

Definition 1. *The bit-flip(s) in the regime is equally likely to occur among the $r_o + 1$ regime bits with probability $\frac{1}{r_o+1}$, where the length $r_o \in [1, 31]$ and it is equally likely to be 1s or 0s with probability $\frac{1}{2}$.*

Theorem 1. *The total expected contribution of Case 1.1 and Case 1.2 is $\mathbb{E}[Case\ 1] = 3.77$.*

Proof. Generalizing the example for *Case 1.1*, the original regime $k_o = (r_o - 1)$ suffers from a bit-flip at position $i \in [1, r_o]$, changing the bit value from 1 to 0. This change produces a new regime $k_f = (i-1) - 1 = (i-2)$. This results in a difference in the regime equal to $|k_o - k_f| = (r_o - 1) - (i-2) = r_o - i + 1$. The probability of such a change is given by $\frac{1}{2} \cdot \frac{r_o}{r_o+1} \cdot \frac{1}{r_o} = \frac{1}{2(r_o+1)}$, where $\frac{1}{2}$ corresponds to the probability of having 1s as regime run-bits, $\frac{r_o}{r_o+1}$ is the probability of flipping one of the r_o regime run-bits, and $\frac{1}{r_o}$ specifies the exact bit position i being flipped. Consequently, the contribution for *Case 1.1* is $\mathbb{E}[Case\ 1.1 \mid\ r_o] = \sum_{i=1}^{r_o} \frac{1}{2(r_o+1)}(r_o - i + 1) = \frac{r_o}{4}$. The *Case 1.2* of Fig. 2, with the run-bits equal to 0s, is symmetric, and thus, has the same contribution as *Case 1.1*. Therefore, the total contribution of both cases for a given r_0 is:

$$\mathbb{E}[Case\ 1 \mid r_o] = \frac{1}{2} \cdot \frac{r_o}{r_o+1} \cdot \frac{r_o}{4} + \frac{1}{2} \cdot \frac{r_o}{r_o+1} \cdot \frac{r_o}{4} = \frac{r_o^2}{4(r_o+1)}$$

Now, averaging over $r_o \in [1, 31]$ gives

$\mathbb{E}[Case\ 1] = \frac{1}{31}\sum_{r_o=1}^{31} \frac{r_o^2}{4(r_o+1)} \approx 3.77.$ □

Theorem 2. *The total expected contribution for Case 2.1 and Case 2.2 is $\mathbb{E}[Case\ 2] = 0.05$.*

Proof. Generalizing the example for *Case 2.1*, a bit-flip from 0 to 1 in the original regime $k_o = r_o - 1$ at position $r_o + 1$, given that the $(r_o + 2)$-th bit is 0, produces a new regime of $r_o + 1$ consecutive 1s. Thus, the changed value becomes $k_f = r_o$, leading to a difference of $|k_o - k_f| = |(r_o - 1) - r_o| = 1$. The probability of this change is given by: (i) the run-bits of 1s occurs with probability $\frac{1}{2}$, (ii) the terminating bit is uniquely located, contributing a factor $\frac{1}{r_o+1}$, and (iii) the $(r_o + 2)^{\text{th}}$ bit being 0 occurs with probability $\frac{1}{2}$. Multiplying these three factors yields $\frac{1}{2} \times \frac{1}{r_o+1} \times \frac{1}{2} = \frac{1}{4(r_o+1)}$. As only one flipping position satisfies these conditions for a given r_o, the expected contribution for *Case 2.1* is $\frac{1}{4(r_o+1)}$. By symmetry, *Case 2.2* has the same contribution. Hence, the total contribution for a given r_o is:

$$\mathbb{E}[Case\ 2 \mid r_o] = \frac{1}{4(r_o+1)} + \frac{1}{4(r_o+1)} = \frac{1}{2(r_o+1)}$$

Averaging over r_o, $\mathbb{E}[Case\ 2] = \frac{1}{31}\sum_{r_o=1}^{31} \frac{1}{2(r_o+1)} \approx 0.05.$ □

Theorem 3. *The total expected contribution for Case 3.1 and Case 3.2 is $\mathbb{E}[Case\ 3] = 0.01$.*

Proof. Generalizing the example for *Case 3.1*, a bit-flip can lead to a regime value $k > r_o$. The probability for regime value being k is

$\frac{1}{2} \times \frac{1}{r_o+1} \times \left(\frac{1}{2}\right)^{k-r_o}$, where $\left(\frac{1}{2}\right)^{k-r_o}$ accounts for the bit-pattern required in order to extend the regime. Note that, this is valid for $k \in [r_o, 30]$. Also, $\frac{1}{2}$ corresponds to run-bits of 1s or 0s, and $\frac{1}{r_o+1}$ ensures the validity of previous regime length. Hence, for *Case 3.1*, an original $k_o = r_o - 1$ and a bit-flip from 0 to 1 results in a new regime value k_f of 1s, so: $|k_o - k_f| = |(r_o - 1) - (k)| = k - r_o + 1$. Based on the above, the contribution of *Case 3.1* is:

$$\mathbb{E}[Case\ 3.1 \mid r_o] = \frac{1}{2(r_o+1)} \times \sum_{k=r_o}^{30} \left(\frac{1}{2}\right)^{k-r_o} \times (k - r_o + 1).$$

Similarly, for *Case 3.2*, the original $k_o = -r_o$ with a bit-flip from 1 to 0 results in a new regime $-k$, and thus, the difference is: $|k_o - k_f| = |-r_o - (-k)| = k - r_o$. Thus, the contribution of *Case 3.2* is:

$$\mathbb{E}[Case\ 3.2 \mid r_o] = \frac{1}{2(r_o+1)} \sum_{k=r_o}^{30} \left(\frac{1}{2}\right)^{k-r_o} \times (k - r_o)$$

By adding the case 3.1 and 3.2 contributions, for a given r_o:

$$\mathbb{E}[Case\ 3 \mid r_0] = \frac{1}{4(r_o+1)^2} \sum_{k=r_o}^{30} \left(\frac{1}{2}\right)^{k-r_o+1} \times [2(k - r_o) + 1]$$

Now, averaging over r_o for $r_o \in [1, 31]$, we get $\mathbb{E}[Case\ 3]$ as

$$\mathbb{E}[Case\ 3] = \frac{1}{31} \sum_{r_o=1}^{31} \left[\frac{1}{4(r_o+1)^2} \sum_{k=r_o}^{30} \left(\frac{1}{2}\right)^{k-r_o+1} [2(k-r_o)+1] \right] \approx 0.01.$$

□

Theorem 4. *The expected* $\mathbb{E}[|e_o - e_f|]$ *is* $\frac{2^{es}-1}{2}$.

Proof. The e_o and e_f are uniformly distributed over $[0, 2^{es} - 1]$, with probability $\frac{1}{2^{es}}$. Correspondingly, their difference $d = |e_o - e_f|$ also ranges from 0 to $2^{es} - 1$ with probability $\frac{1}{2^{es}}$. Based on this,

$$\mathbb{E}[|e_o - e_f|] = \sum_{d=0}^{2^{es}-1} \frac{d}{2^{es}} = \frac{2^{es} - 1}{2} \tag{4}$$

□

Finally, according to Eq. 3, to compute η for a given *Posit(32,es)* configuration, i.e., $2^{es}\mathbb{E}[|k_o - k_f|] + \mathbb{E}[|e_o - e_f|]$, we can use Theorem 4 to derive the second term as shown in Equation 4 and Theorems 1, 2, and 3 to derive the first term as:

$$2^{es}\mathbb{E}[|k_o - k_f|] = 2^{es}\Big(\sum_{i=1}^{3} \mathbb{E}[Case\ i]\Big) = 2^{es} \times 3.83. \tag{5}$$

Therefore, the final Expected Catastrophic Error (η) for *Posit(32,es)* is $\eta = 2^{es} \times 3.83 + \frac{2^{es}-1}{2} = 4.33 \times 2^{es} - 0.5$.

3.2 Exception Triggering

A bit-flip can corrupt a value, making it invalid and triggering an exception. This is quantified by the metric *Exception Probability* (p), which measures the likelihood that a random bit-flip(s) renders the value invalid. As reported in [17], single bit-flips occur with 10× higher rate than multiple bit-flips, while ~85% of multiple bit-flips are usually adjacent double bit-flips [18]. Thus, our analysis is illustrated for single bit-flips and adjacent double bit-flips.

Lemma 1. *For a valid Posit(32,es) value, uniformly distributed over* $2^{32} - 1$ *bit patterns, the exception triggering probability under single or adjacent double bit-flips equals* 2.3×10^{-10}.

Proof. Valid *Posit(32,es)* values are uniformly distributed over $2^{32}-1$ bit patterns. An error occurs by either a single bit-flip or an adjacent double bit-flip. For a single bit-flip, we have 32 cases, occurring at 10× higher rate than an adjacent double bit-flip, i.e., $\frac{9}{10}$ of the time. For an adjacent double bit-flip, we have 31 cases occurring $\frac{1}{10}$ of the time. Hence, $p = \frac{9}{10}P(E \mid \text{single}) + \frac{1}{10}P(E \mid \text{double})$. A single flip at bit $k \in [0, 31]$ ($\frac{1}{32}$ probability each) can cause an exception only if the number differs in one bit from the exception bit pattern. There are 32 such numbers, each with probability $\frac{1}{2^{32}-1}$, so $P(E \mid \text{single}) = \frac{1}{32} \cdot \frac{32}{2^{32}-1} = \frac{1}{2^{32}-1}$. For a double flip at $\{i, i+1\}$, $i \in [0, 30]$ (1/31 each), an exception arises for 31 numbers where each number occurs with probability $\frac{1}{2^{32}-1}$, so $P(E \mid \text{double}) = \frac{1}{31} \cdot \frac{32}{2^{32}-1} = \frac{1}{2^{32}-1}$.

Thus, $p = \left(\frac{9}{10} + \frac{1}{10}\right)\frac{1}{2^{32}-1} = \frac{1}{2^{32}-1} \approx 2.3 \times 10^{-10}$.

□

3.3 Posit error analysis

We report the Posit resilience analysis achieved by using our analytical framework. By analyzing the η computation cases (Theorems 1–4), we observe that errors in the regime contribute the most to η. This is also confirmed by empirical studies [6]. In particular, single bit-flips in the run-bits of the regime (Theorem 1) have the greatest impact, as Case 1 of Fig. 2 is more likely to happen than Cases 2 and 3. In these cases, we observe that the regime corruption is often compensated for by the subsequent exponent bits. As sketched in Fig. 2, in *Case 1.1*, a bit-flip impacting a regime with run-bits of 1s *reduces* the magnitude of the Posit number. This also tends to increase the exponent, thereby *increasing* the magnitude of the number and counteracting the bit-flip effect in the regime. The symmetric case occurs in *Case 1.2*, where a bit-flip impacting a regime with run-bits of 0s *increases* the magnitude of the Posit number but also tends to *reduce* the exponent, thereby attenuating the fault impact. However, this self-compensating behavior breaks down when: ① the regime becomes very large, and no bits remain to fully encode the exponent, and ② the exponent becomes too large and overcompensates the error. For the effect ②, the maximum offset introduced by the exponent is 2^{es-1}, which may lead to overcompensation, particularly for large *es* (i.e., $es \geq 5$). Increasing *es* to large values inherently degrades precision, thus using too large *es* values is not recommended [1]. For the suggested range ($0 \leq es < 5$), effect ① remains dominant. These insights collectively answer ***Q1***. As Table 1 shows, η increases with increasing *es*, thus highlighting, consistently with the trends reported in prior studies [5], [7], that *Posit configurations with high es values have reduced resilience, compared to those having lower es values.*

4 Proposed resilient PC-Posit representation

To systematically enhance Posit resilience (***Q2***), we propose a novel Posit representation, *Proactively-Clipped Posits (PC-Posits)*. To avoid the reduction in the self-compensation behavior of Posits, due to the effect ①, PC-Posits proactively limits the maximum regime length to ensure that exponent bits are always present. This means that for a given configuration *Posit(N,es)*, the corresponding *PC-Posit(N,es)* version will impose $r \in [1, N - es - 2]$, as shown in Figure 3. In this way, the regime always leaves space for its terminating bit and to encode the exponent. The drawback of such an approach is the reduced range compared to the original Posit configuration. Thus, to have a comparable range to a given Posit configuration

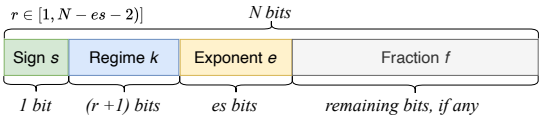

Figure 3. Proposed PC-posit format

Table 1. η, p and range for Posit, PC-Posit and FP32. For FP32, F1 corresponds to a single bit-flip and F2 to adjacent double bit-flips.

Format	*es*	*p*	η	Range
PC-Posit(32,0)	0	$\approx 2.3 \times 10^{-10}$	1.26	$(0.1 \times 10^{-4}, 3.2 \times 10^{4})$
Posit(32,0)	0	$\approx 2.3 \times 10^{-10}$	3.83	$(0.2 \times 10^{-8}, 1.0 \times 10^{9})$
PC-Posit(32,1)	1	$\approx 2.3 \times 10^{-10}$	3.02	$(0.4 \times 10^{-8}, 5.3 \times 10^{8})$
Posit(32,1)	1	$\approx 2.3 \times 10^{-10}$	8.16	$(0.7 \times 10^{-17}, 1.4 \times 10^{17})$
PC-Posit(32,2)	2	$\approx 2.3 \times 10^{-10}$	6.54	$(0.7 \times 10^{-17}, 1.4 \times 10^{17})$
Posit(32,2)	2	$\approx 2.3 \times 10^{-10}$	16.82	$(0.2 \times 10^{-32}, 6.4 \times 10^{32})$
PC-Posit(32,3)	3	$\approx 2.3 \times 10^{-10}$	13.58	$(0.1 \times 10^{-34}, 1.0 \times 10^{34})$
Posit(32,3)	3	$\approx 2.3 \times 10^{-10}$	34.15	$(1.2 \times 10^{-63}, 8.1 \times 10^{62})$
PC-Posit(32,4)	4	$\approx 2.3 \times 10^{-10}$	27.66	$(1.8 \times 10^{-68}, 5.3 \times 10^{67})$
Posit(32,4)	4	$\approx 2.3 \times 10^{-10}$	68.78	$(3.8 \times 10^{-121}, 2.5 \times 10^{120})$
PC-Posit(32,5)	5	$\approx 2.3 \times 10^{-10}$	56.12	$(3.4 \times 10^{-125}, 1.9 \times 10^{124})$
FP32 (F1)	8	$\approx 9.8 \times 10^{-4}$	31.87	$(0.1 \times 10^{-37}, 3.4 \times 10^{38})$
FP32 (F2)	8	$\approx 1.01 \times 10^{-3}$	34.13	$(0.1 \times 10^{-37}, 3.4 \times 10^{38})$

Posit(N,es), the corresponding *PC-Posit* version to use is the one with 1 more exponent bit, i.e., *PC-Posit(N,es+1)*. For example, *Posit(32,2)* and *PC-Posit(32,3)* achieve a similar representable range, but *PC-Posit(32,3)* provides significantly higher SE resilience, as confirmed by their Expected Catastrophic Error (η) values. To obtain the η and p values for *PC-Posit(32,es)*, we applied the same approach used for Posit (shown in Sec. 3). *The η values for PC-Posit(32,es) can be calculated as* $1.76 \times 2^{es} - 0.5$ *and p values as reported in section 3.2.* Table 1 reports η and p for different Posit and PC-Posit configurations. It shows that PC-Posits consistently provide better resilience than their Posit counterparts over comparable ranges. Hence, this discussion answers ***Q2***. For values of $es \geq 5$, the effectiveness of PC-posit will be reduced. Indeed, the effect ②, which diminishes the self-compensating behavior of the exponent, will start to dominate over the effect ① leveraged by PC-posit to enhance resilience. As mentioned in Sec. 2, large *es* values (≥ 5) are not recommended, from the arithmetic standpoint [1].

Soft-Error Resilience of PC-Posit, Posit vs IEEE FP: Following the same reasoning as described in Section 3, we can extract the soft error resilience of IEEE FP32. Consider an IEEE FP32 number y_o having absolute value $|y_o| = 2^{e_o - 127} \cdot (1 + f_o)$. After a bit-flip(s) in the exponent and/or fraction bits, the number becomes $|y_f| = 2^{e_f - 127} \cdot (1 + f_f)$, where e_f and f_f are the faulty FP32 fields. The absolute logarithmic error in exponent bits after a bit-flip is: $|\log_2 |y_o| - \log_2 |y_f|| \approx |(e_o - e_f)|$, with the expected value $\eta = \mathbb{E}[|\log_2 |y_o| - \log_2 |y_f||] \approx \mathbb{E}[|e_o - e_f|]$.

The η value for IEEE FP32 can be estimated by using the expectation $\mathbb{E}[|e_o - e_f|]$ in the exponent field of an IEEE 754 FP32 number. For single-bit flips in the 8-bit exponent, flipping with uniform probability 1/8, $\mathbb{E}[|e_0 - e_1| \mid \text{single flip}] = \frac{255}{8} = 31.875$. For double adjacent bit flips, consider pairs $(i, i+1)$ for $i = 0$ to 6, giving 7 pairs. With probability 1/7, $\mathbb{E}[|e_0 - e_1| \mid \text{double flip}] = \frac{3}{7} \sum_{i=0}^{6} 2^i = \frac{381}{7} \approx 54.43$. Combining with a 9:1 probability ratio, $\mathbb{E}[|e_0 - e_1|] = \frac{9}{10} \cdot 31.875 + \frac{1}{10} \cdot \frac{381}{7} \approx 34.13$.

Using similar reasoning, we can also compute the exception probability p. For an IEEE-754 FP32 value uniformly sampled from the $2^{32} - 2^{24}$ valid bit patterns, the p value is $\approx 9.8 \times 10^{-4}$ for a single-bit flip and $\approx 1.01 \times 10^{-3}$ for two adjacent bit flips.

Table 1 summarizes all the soft-error resilience for all the number representations. It shows that Posits have lower p than FP32, suggesting better numerical validity of the stored numbers. Formats, such as *Posit(32,0)*, *Posit(32,1)*, and *Posit(32,2)* exhibit lower η than FP32 (F1/F2), indicating stronger SE resilience. Contrarily, *Posit(32,3)* shows higher η than FP32 (F1), suggesting reduced resilience. Notably, *PC-Posit(32,0)*, *PC-Posit(32,1)*, and *PC-Posit(32,3)* achieve a lower η, surpassing FP32 (F1) in resilience. By comparison, *PC-Posit(32,3)* balances range and resilience, offering further improvement. The next section empirically validates these insights.

5 Experimental Evaluation

This section presents results for the most commonly utilized configuration, typically benchmarked against FP [3], [6], namely *Posit(N,2)*. We compare this with the proposed *PC-Posit(N,3)*, which offers a similar range. Additionally, we include *Posit(N,3)* as well as several widely used formats: *FP32*, *FP16*, *BF16*, *FP8*, and *INT8*.

Benchmarks: We first evaluate the proposed approach on a GEneral Matrix Multiplication (GEMM) kernel, multiplying two 100×100 square matrices. Each element of the matrices was chosen uniformly randomly from the full numerical range of the corresponding format (listed in Table 1). Moreover, we use `Image Sharpening` (`IS`), `Gaussian Smoothing` (`GS`), `K-Nearest Neighbors` (`KNN`), `Neural Networks` (`DNN`), `Support Vector Machines` (`SVM`). `IS` and `GS` apply the unsharp mask method and Gaussian blur, respectively, on 30 images [19]. `IS` and `GS` results are evaluated using Peak Signal-to-Noise Ratio (PSNR) [20]. `KNN` is tested on CIFAR-100 with $k = 5$, `DNN` inference uses a pre-trained ResNet-50 on CIFAR-100, and `SVM` employs a linear kernel with Fisher vector features. These results are evaluated using classification accuracy [21].

Fault injection: To capture single-bit and multi-bit fault scenarios, we use two memory fault models, as in state-of-the-art work. The Soft Error Rate (SER) of a hardware component x is expressed as $\text{SER}_x = e_x \cdot p_x$, where e_x denotes the error-generation rate (dependent on technology and operating conditions), and p_x represents the conditional probability of a system-visible failure, typically reported using Failures-In-Time (FIT) rates [22]. As reported in [17], single-bit flips occur with e_x ranging from 1–4%, whereas multi-bit faults are rarer, with rates between 0.1–0.4% and ~85% being 2-bit flips. Assuming uniform FIT likelihood ($p_x = 0.5$), the estimated SER for non-adjacent 2-bit flips is $\approx 0.85 \times 0.5 \times (0.1\% \text{ to } 0.4\%) \approx 0.02\%$ to 0.08%. For single-bit flips ($p_x \approx 1$), SER lies between 1–4%. Correspondingly, our first fault model (F1) accounts only for single-bit flips. The second fault model (F2) considers single-bit and adjacent double-bit flips. Each fault model contains four events (E1–E4) based on the SER of single-bit flips, i.e., 1%–4% for single-bit flips and 0.02%–0.08% for adjacent double-bit flips. We performed the fault injections using eight different fault models: four for single-bit flips (F1E1, F1E2, F1E3, and F1E4) and four for double adjacent-bit flips (F2E1, F2E2, F2E3, and F2E4). The required number of injections is computed with 95% confidence and 2% error margin [23], yielding up to 9,500 injections for F1 and 9,300 for F2 per benchmark, depending on *SER*.

Analysis: Table 2 shows the Mean Relative Error Distance (MRED) (in %) over 10,000 iterations for different fault models across different formats for the GEMM kernel. Posit(N,2) consistently achieves better results than FP counterpart and Posit(N,3), in accordance with state-of-the-art findings. PC-Posit(N,3) improves upon Posit(N,2), consistently drawing down the error. Figure 4 shows, through violin plots, the distribution of application output quality degradation, i.e., PSNR for `GS` and `IS`, and Accuracy for `KNN`, `DNN`, and `SVM`), under different fault models (F1E1–F2E4) across 8, 16, and 32-bit formats. For `DNN`, `KNN`, and `SVM`, *PC-Posit(N,3)* ( ) improves accuracy by an average of 2.3 pp over *Posit(N,2)* ( ) and 19.4 pp over *Posit(N,3)* ( ). For `IS` and `GS`, *PC-Posit(8,3)* delivers substantial gains compared to

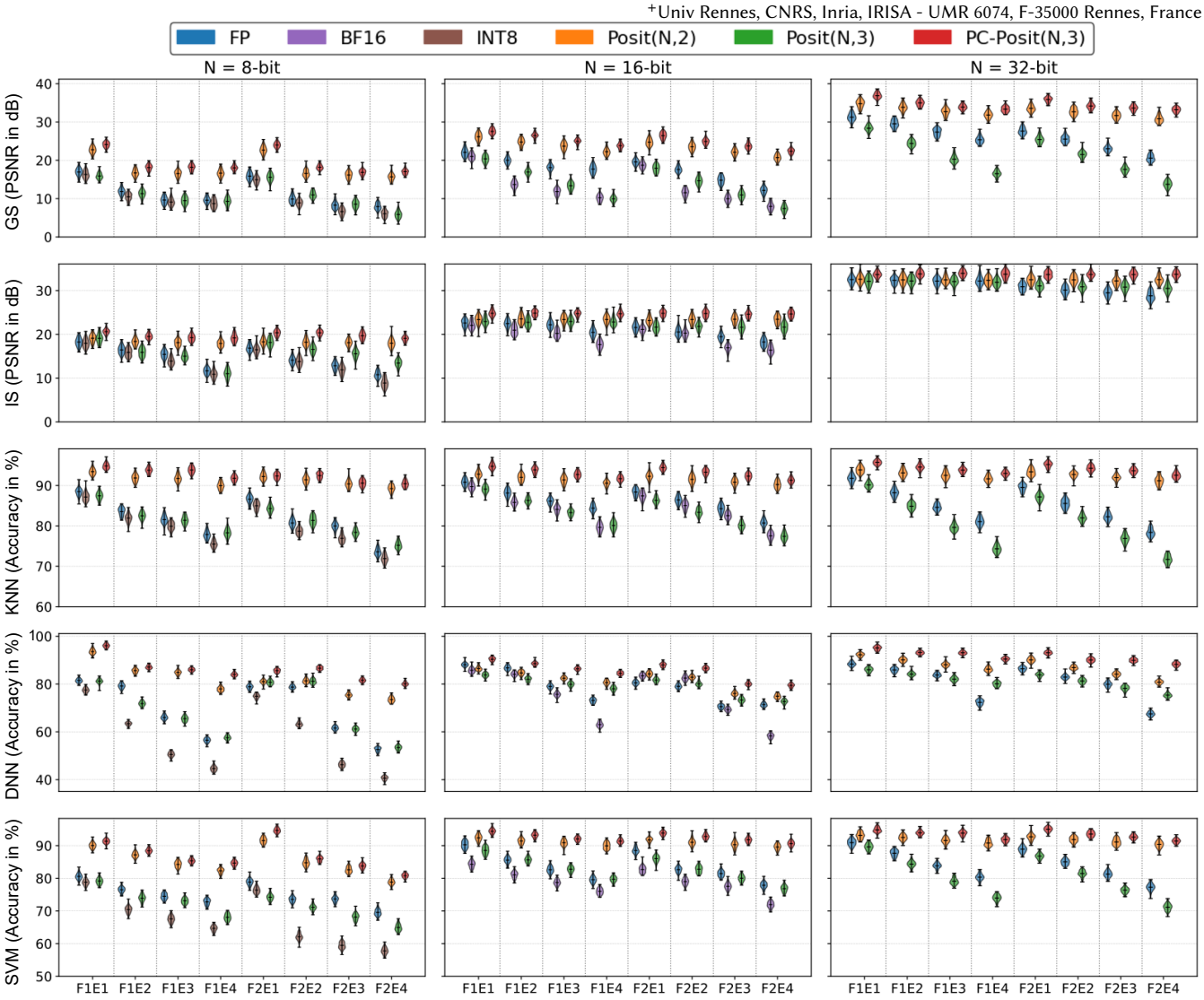

Figure 4. Application evaluation results for single and multi-bit fault injections with varying SER.

Table 2. MRED (in %) for GEMM with Posit, PC-Posit, and IEEE formats under fault models (lower is better)

Event	Posit(32,3)	Posit(32,2)	PC-Posit(32,3)	FP32	Posit(16,3)	Posit(16,2)	PC-Posit(16,3)	FP16	Posit(8,3)	Posit(8,2)	PC-Posit(8,3)	FP8
F1E1	1.95×10^{-3}	6.15×10^{-4}	$\mathbf{5.00\times10^{-4}}$	1.23×10^{-3}	0.049	0.037	**0.035**	0.039	2.34	1.30	**0.98**	1.95
F1E2	3.90×10^{-3}	1.23×10^{-3}	$\mathbf{1.00\times10^{-3}}$	2.47×10^{-3}	0.098	0.073	**0.07**	0.078	4.69	2.60	**1.95**	3.91
F1E3	5.85×10^{-3}	1.85×10^{-3}	$\mathbf{1.50\times10^{-3}}$	3.70×10^{-3}	0.15	0.115	**0.105**	0.12	7.03	3.90	**2.93**	5.86
F1E4	7.80×10^{-3}	2.46×10^{-3}	$\mathbf{2.00\times10^{-3}}$	4.93×10^{-3}	0.20	0.15	**0.14**	0.16	9.38	5.20	**3.91**	7.81
F2E1	1.97×10^{-3}	6.20×10^{-4}	$\mathbf{5.05\times10^{-4}}$	1.25×10^{-3}	0.049	0.038	**0.036**	0.039	2.36	1.31	**0.98**	1.97
F2E2	3.94×10^{-3}	1.25×10^{-3}	$\mathbf{1.01\times10^{-3}}$	2.50×10^{-3}	0.098	0.076	**0.072**	0.079	4.72	2.62	**1.97**	3.94
F2E3	5.91×10^{-3}	1.87×10^{-3}	$\mathbf{1.52\times10^{-3}}$	3.75×10^{-3}	0.15	0.116	**0.108**	0.12	7.08	3.94	**2.96**	5.91
F2E4	7.88×10^{-3}	2.50×10^{-3}	$\mathbf{2.02\times10^{-3}}$	5.00×10^{-3}	0.20	0.154	**0.144**	0.16	9.44	5.24	**3.94**	7.88

Posit(8,2) and *Posit(8,3)*. It improves PSNR by 1.3 pp over *Posit(8,2)* and by 29.1 pp over *Posit(8,3)*. For 16-bit formats, *PC-Posit(16,3)*yields a 1.4 pp gain over *Posit(16,2)* and 25.4 pp over *Posit(16,3)*; similar are the trends for 32-bits. *PC-Posit(N,3)* consistently mitigates the weaknesses of *Posit(N,3)* and strengthens the resilience of *Posit(N,2)*.

Compared with IEEE FP formats, *PC-Posit* consistently outperforms in resilience. For IS and GS, *PC-Posit(32,3)* ( ) achieves a 19.6 pp PSNR gain over *FP32* ( ). Similarly, *PC-Posit(16,3)* improves by 22.7 pp over *FP16* and 33.9 pp over *BF16* ( ). *PC-Posit(8,3)* improves by 30.6 pp over *FP8* and 37.6 pp over *INT8* ( ). For DNN, KNN, and SVM, *PC-Posit* achieves higher accuracy in both fault-free and faulty cases. On average, *PC-Posit(32,3)* improves accuracy by 15.2 pp over *FP32*, *PC-Posit(16,3)* by 9.4 pp over *FP16* and 16.1 pp over *BF16*, and *PC-Posit(8,3)* by 18.5 pp over *FP8* and 25.2 pp over *INT8*.

Hardware: We implemented *PC-Posit* in Verilog and synthesized it with Synopsys Design Compiler using the NanGate 45nm Open Cell Library under a 1.0 ns clock, 1.10 V supply, and 25°*C*. Multiplications were performed with Posit(32,2), Posit(32,3), PC-Posit(32,3) and

FP32. As observed in Table 3, *PC-Posit* achieves up to 47.2% energy savings over standard Posit. The savings are achieved through the reduced complexity of the regime encoder and decoder hardware. FP32 has 3.4% less area than *PC-Posit*, it offers reduced resilience.

Table 3. Hardware evaluation results for Posit and PC-Posit.

Format	Area (μm^2)	Power (mW)	Delay (ns)	Energy (pJ)
Posit(32,2)	8346.2	0.23	4.4	1.01
Posit(32,3)	8590.5	0.27	4.8	1.31
PC-Posit(32,3)	7946.7	0.17	4.1	0.69
FP32	7690.0	0.15	3.6	0.54

6 Conclusion

This work introduces the first analytical framework for assessing SE resilience in Posits, addressing Q1. Leveraging this framework, we propose *PC-Posits* to enhance resilience without costly error correction, addressing Q2. We evaluated *PC-Posits* through extensive fault injection and observed up to 29.1 pp higher SE resilience than standard Posits. Our proactive clipping also reduced energy consumption by up to 47.2% relative to standard Posits, Further, *PC-Posits* also achieved up to 33.9 pp higher SE resilience over IEEE FP formats.

References

[1] J. L. Gustafson *et al.*, "Beating floating point at its own game: Posit arithmetic," *Supercomputing frontiers and innovations*, vol. 4, no. 2, pp. 71–86, 2017.

[2] X. Li *et al.*, "Design and evaluation of gpu-fpx: A low-overhead tool for floating-point exception detection in nvidia gpus," in *Proceedings of the 32nd International Symposium on High-Performance Parallel and Distributed Computing*, 2023, pp. 59–71.

[3] I. Alouani *et al.*, "An investigation on inherent robustness of posit data representation," in *2021 34th International Conference on VLSI Design and 2021 20th International Conference on Embedded Systems (VLSID)*, IEEE, 2021, pp. 276–281.

[4] B. Schlueter *et al.*, "Evaluating the resiliency of posits for scientific computing," in *Proceedings of the SC'23 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis*, 2023, pp. 477–487.

[5] J. E. R. Condia *et al.*, "Analyzing the structural and operational impact of faults in floating-point and posit arithmetic cores for cnn operations," in *2024 IEEE European Test Symposium (ETS)*, IEEE, 2024, pp. 1–4.

[6] J. E. R. Condia *et al.*, "Investigating and mitigating critical faults in floating-point and posit arithmetic hardware," *IEEE Transactions on Emerging Topics in Computing*, pp. 1–12, 2025.

[7] M. Yousefloo *et al.*, "Design exploration of fault-tolerant deep neural networks using posit number representation system," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 32, no. 7, pp. 1350–1363, 2024.

[8] H. D. Dixit *et al.*, "Silent data corruptions at scale," *arXiv preprint arXiv:2102.11245*, 2021.

[9] G. Papadimitriou *et al.*, "Silent data corruptions: Microarchitectural perspectives," *IEEE Transactions on Computers*, vol. 72, no. 11, pp. 3072–3085, 2023.

[10] E. Cheng *et al.*, "Cross-layer resilience: Challenges, insights, and the road ahead," in *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019, pp. 1–4.

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

[12] A. Vallero *et al.*, "Syra: Early system reliability analysis for cross-layer soft errors resilience in memory arrays of microprocessor systems," *IEEE Transactions on Computers*, vol. 68, no. 5, pp. 765–783, 2018.

[13] G. Andrew *et al.*, "Differentially private learning with adaptive clipping," *Advances in Neural Information Processing Systems*, vol. 34, pp. 17 455–17 466, 2021.

[14] B. Defraene *et al.*, "Real-time perception-based clipping of audio signals using convex optimization," *IEEE transactions on audio, speech, and language processing*, vol. 20, no. 10, pp. 2657–2671, 2012.

[15] Y. Liu *et al.*, "Hierarchical filter and refinement system over large polygonal datasets on cpu-gpu," in *2019 IEEE 26th International Conference on High Performance Computing, Data, and Analytics (HiPC)*, IEEE, 2019, pp. 141–151.

[16] V. Gohil *et al.*, "Fixed-posit: A floating-point representation for error-resilient applications," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 10, pp. 3341–3345, 2021.

[17] M. Hashimoto *et al.*, "Characterizing sram and ff soft error rates with measurement and simulation," *Integration*, vol. 69, pp. 161–179, 2019.

[18] S. S. Mukherjee *et al.*, "Cache scrubbing in microprocessors: Myth or necessity?" In *10th IEEE Pacific Rim International Symposium on Dependable Computing, 2004. Proceedings.*, IEEE, 2004, pp. 37–42.

[19] J. Burkardt, *Dataset*, https://people.math.sc.edu/Burkardt/data/tif/tif.html, Online Database, collection of standard test images in TIF format, 2023.

[20] A. Hore *et al.*, "Image quality metrics: Psnr vs. ssim," in *2010 20th international conference on pattern recognition*, IEEE, 2010, pp. 2366–2369.

[21] V. Mishra *et al.*, "Vadf: Versatile approximate data formats for energy-efficient computing," *ACM Transactions on Embedded Computing Systems*, vol. 22, no. 5s, pp. 1–21, 2023.

[22] A. Vijayan *et al.*, "Online soft-error vulnerability estimation for memory arrays and logic cores," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 37, no. 2, pp. 499–511, 2017.

[23] R. Leveugle *et al.*, "Statistical fault injection: Quantified error and confidence," in *2009 Design, Automation & Test in Europe Conference & Exhibition*, IEEE, 2009, pp. 502–506.