Fault Analysis of Microscaling Formats on a RISC-V SoC

Abstract

Microscaling (MX) formats share one exponent across an element block, creating a new fault surface: a single-bit flip in the shared exponent corrupts all block elements simultaneously. Five MX-compatible 8-bit encodings (MXINT8, MXFP8-E4M3, MXFP8-E5M2, LOG8-SUM, and LOG8-MAX) are compared through exhaustive single-bit weight fault injection across four workloads using gate-level simulation on the CAPRI1 RISC-V SoC, with the primary workload additionally validated on fabricated silicon. Format choice alone can substantially change vulnerability. Three MX-specific mechanisms explain this variation: block-shared exponent amplification, attention routing inversion from exponent-field faults, and log-domain fault filtering in the max-reduction variant. No single format dominates all axes: LOG8-MAX leads speed and energy but is limited by accuracy on some attention workloads, MXINT8 provides the strongest fault resilience with full accuracy, and LOG8-SUM offers a balanced compromise across speed, accuracy, and resilience. MX element encoding should be treated not only as an accuracy-efficiency choice, but also as a security-relevant design decision.

Keywords

Microscaling formats, fault injection, RISC-V, hardware security, log-domain arithmetic, attention mechanism

1 Introduction

Edge AI systems increasingly adopt low-precision numeric formats to reduce memory, bandwidth, and energy. Microscaling (MX) formats [1] share a single scale value across an element block, improving storage efficiency while preserving dynamic range. Figure 1 introduces five MX-compatible 8-bit formats, including two log-domain variants we propose (LOG8-SUM and LOG8-MAX), each with distinct trade-offs for coefficient storage and inference speed. All share the E8M0 block scale; they differ in element interpretation and reduction arithmetic. As MX moves from standardization toward deployment, its evaluation must extend beyond accuracy and efficiency to include fault resilience.

Fault resilience matters most in embedded and safety-critical systems, where transient disturbances or targeted fault injection can corrupt stored weights and silently alter model outputs. A bit flip in a conventional per-element representation perturbs one value. A bit flip in an MX shared exponent perturbs an entire block simultaneously. This block-level coupling means MX formats exhibit fault behavior qualitatively distinct from standard INT8 or FP32 representations.

Prior MX hardware work [8–10] addresses throughput, area, and accuracy, while neural-network fault studies [2, 3, 15] target conventional formats. A comparative study examining fault behavior across MX encodings on a common platform remains absent. This gap matters because MX formats differ structurally in ways that affect fault propagation, as we detail in Section 2.

We study five MX-compatible 8-bit formats through exhaustive gate-level single-bit SRAM fault injection on the CAPRI1 netlist, derived from a fabricated 0.18 μm RISC-V SoC with scan-chain observability. This setup provides controlled format-to-format comparison under a single-bit model while preserving the timing and state visibility of the target platform. It supports both misclassification analysis and fault propagation tracking through scan-chain state capture.

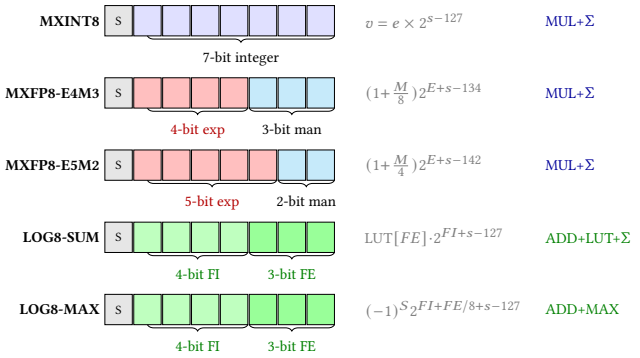

Figure 1: Five MX element encodings: bit-field layouts and decode formulas. MXFP8-E4M3's 4-bit element exponent (red) creates exponential fault sensitivity (2^1–2^8 per bit flip). LOG8-SUM and LOG8-MAX share the same [S:1][FI:4][FE:3] encoding; LOG8-SUM converts to linear via LUT before summation, while LOG8-MAX replaces MUL+Σ with ADD+MAX, bounding fractional-bit error to $\leq 1.84\times$. MXINT8 has purely linear error. The shared exponent s (E8M0) applies to all five formats.

Our results show that format choice materially changes vulnerability. No single format dominates all axes: LOG8-MAX leads speed (2–4.6×) and energy, MXINT8 provides the strongest fault resilience with full accuracy, and LOG8-SUM balances all three without hardware multipliers. MXFP8-E4M3 is the most vulnerable format, while LOG8-MAX achieves zero misclassifications on the primary workload (Exp 1) but is limited to 50% golden accuracy on one multi-token attention workload. These results suggest that MX format selection is a hardware-security design decision, not merely an accuracy–efficiency trade-off.

The contributions are as follows:

(1) A **five-format comparison across four workloads** on the CAPRI1 platform (0.18 μm RISC-V SoC), with the primary workload additionally validated on fabricated silicon, using exhaustive fault analysis and scan-chain observability.

(2) A **mechanistic explanation** of up to 8× vulnerability disparity on the primary workload (0%–8.9%), tracing it to exponent bit-width, reduction path, and attention routing sensitivity.

(3) Evidence that **no single format dominates all axes**: LOG8-MAX leads speed/energy, MXINT8 leads resilience, and LOG8-SUM provides a Pareto-optimal balance.

Section 2 introduces the five MX formats and their hardware cost. Section 3 defines the threat model and formalizes security properties. Section 4 analyzes how each format responds to faults within the attention dataflow. Section 5 describes the CAPRI1 platform and workloads. Section 6 presents results from cross-workload overview to per-format mechanisms. Sections 7–8 discuss implications and related work, and Section 9 concludes.

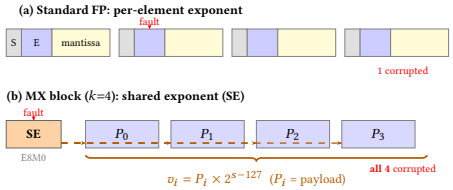

Figure 2: Standard FP vs. MX block. (a) Per-element exponent; a fault corrupts one value. (b) MX shares one exponent (SE) across k=4 elements; a single SE bit flip shifts *all* values by $2^{\pm\Delta}$ (P_i = element payload).

2 Background and Preliminaries

2.1 Microscaling Formats

An MX block [1] packs one shared scale byte (E8M0) with k element bytes (k=4 here), as shown in Figure 2. The decoded value $v_i = P_i \cdot 2^{s-127}$ couples every element to a single scale s. This coupling underlies both the compression benefit and the fault risk: a bit flip in the shared exponent rescales all k values at once, whereas a flip in an element byte affects only one.

2.2 Five MX-Compatible Encodings

Figure 1 shows the five 8-bit encodings evaluated in this work; all share the E8M0 block scale but differ in element interpretation and reduction arithmetic. **MXINT8** encodes each element as a signed 7-bit integer with no element exponent; a bit flip produces a bounded, linear perturbation proportional to the flipped bit position. **MXFP8-E4M3** adds a 4-bit element exponent atop the 8-bit shared exponent (12 effective exponent bits) and a 3-bit mantissa; a single flip can shift magnitude by 2^1 to 2^8. **MXFP8-E5M2** widens the element exponent to 5 bits with a 2-bit mantissa, trading granularity for dynamic range. **LOG8-SUM** uses the [S:1][FI:4][FE:3] log encoding proposed by Dally [7], where FI is a 4-bit integer log and FE is a 3-bit fractional log; each product is computed as an addition in the log domain, converted to linear via a lookup table, then accumulated through summation. **LOG8-MAX** shares the same encoding as LOG8-SUM but replaces summation with max-reduction in the log domain, eliminating both the LUT conversion and the multiplier.

2.3 Log-Domain Arithmetic and Hardware Cost

Both log formats replace multiplication with a 7-bit integer addition: $p_i = \ell_{w_i} + \ell_{x_i}$. They diverge at reduction:

$$z_j^{\text{LOG8-SUM}} = \textstyle\sum_i \text{lin}(\ell_{w_i} + \ell_{x_i}) \quad \text{(convert, then sum)} \tag{1}$$

$$z_j^{\text{LOG8-MAX}} = \max_i (\ell_{w_i} + \ell_{x_i}) \quad \text{(max, no conversion)} \tag{2}$$

Table 1 compares per-MAC cost for each format on the CAPRI1 rv32ic core. LOG8-MAX costs ~2 cyc/MAC versus ~21 for MXINT8, yielding 2–4.6× speedup. At 180 nm [11], a 7-bit add consumes ~0.03 pJ and a 32-bit integer multiply consumes ~3.1 pJ, giving LOG8-MAX an estimated ~8× energy advantage per MAC operation.

Table 1: Per-MAC cost on rv32ic. LOG8-MAX needs only 2 cycles (ADD+CMP); multiply-based formats require ~20-cycle software multiply.

Format	Multiply	Decode	Accumulate	Cyc/MAC
LOG8-MAX	ADD (1)	–	CMP (1)	~2
LOG8-SUM	ADD (1)	LUT+SHIFT (3)	ADD (1)	~5
MXINT8	smul (20)	–	ADD (1)	~21
MXFP8	smul (20)	decode (5)	ADD (1)	~26

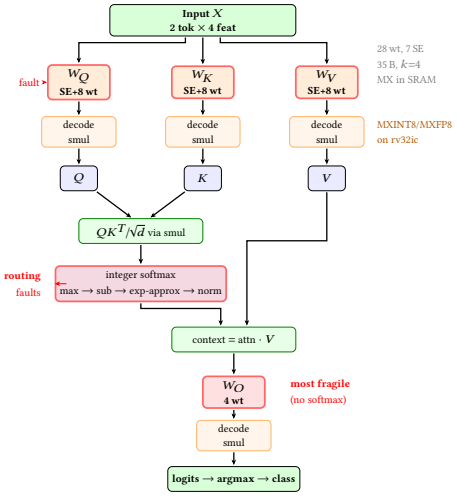

Figure 3: Attention inference on CAPRI1 rv32ic for MXINT8/MXFP8 formats. Each weight matrix stores MX blocks (SE + k=4 elements) in SRAM. LOG8 formats replace decode+smul with log-domain ADD (Table 1). W_O maps to logits without softmax (most fragile).

2.4 Fault Observability

No standard mechanism exists to observe a fault at the instant it corrupts internal state during inference. CAPRI1 (Figure 6) integrates a 12,756-flip-flop scan chain, originally designed for manufacturing test, that captures complete processor state (every 50 cycles). Hamming distance between golden and faulted snapshots converts a binary pass/fail outcome into time-resolved fault propagation analysis.

3 Threat Model and Security Analysis

3.1 Threat Model

The adversary targets SRAM-resident model weights with the goal of *silent misclassification*: the program runs to completion but returns the wrong class.

SRAM-resident weights are vulnerable to several physical attack mechanisms. EMFI induces transient bit flips through localized EM pulses. Voltage glitching corrupts SRAM cells during read or write operations. Power-supply manipulation can wipe partial SRAM content. Among these, we adopt a single-bit fault model because it isolates format-dependent behavior from multi-bit masking effects and represents a profiled attacker with calibrated glitch parameters. We assume the attacker knows weight byte addresses, consistent with white-box embedded fault analysis. We exclude instruction corruption, control-flow hijacking, and denial-of-service faults.

3.2 SE and Element Faults

Attention-based models dominate edge AI inference [16], and each weight matrix in the attention dataflow stores MX blocks containing both SE and element bytes. Figure 3 shows the fault paths through this dataflow; Section 4 analyzes how each format responds to faults at each stage.

Under the single-bit fault model, a weight fault targets either the shared exponent or an element byte. An SE fault (Level 1, Figure 4(a)) shifts all k block values by $2^{\pm\Delta}$; all five formats are equally vulnerable at this level. An element fault (Level 2, Figure 4(b)) corrupts one

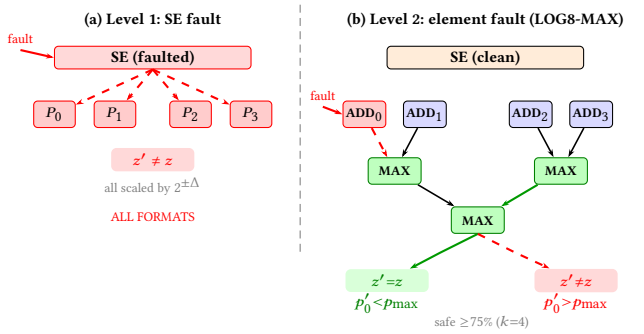

Figure 4: Two-level fault behavior. (a) SE fault shifts all k=4 elements by $2^{\pm\Delta}$; all formats are equally vulnerable. (b) Element fault in LOG8-MAX: MAX discards faults below the maximum (Δz=0); summation-based formats always propagate.

decoded value. In summation-based formats, every corrupted term shifts the accumulated output. In LOG8-MAX, the max operator discards the corrupted term if its value remains below the current maximum.

3.3 Security Properties

The following properties formalize the containment guarantees that follow from this two-level structure.

Definition 1 (Classification Margin). $m = z_{\hat{y}} - \max_{j \neq \hat{y}} z_j > 0$. *A fault is* exploitable *iff it changes the predicted class* $\hat{y}$.

Fault amplification. Summation-based formats propagate every faulted term to the output. Worst-case amplification: MXINT8 254×, LOG8-SUM 256× (integer-part flip at b=3: 2^8), MXFP8-E5M2 65,536×. For LOG8-MAX, the fault on term k produces:

$$\Delta z_j^{\mathrm{LD}} = \begin{cases} 0 & \text{if } (\ell_{w'_k} + \ell_{x_k}) \leq M_{-k} \\ (\ell_{w'_k} + \ell_{x_k}) - M_{-k} & \text{otherwise} \end{cases} \tag{3}$$

where $M_{-k} = \max_{i \neq k}(\ell_{w_i} + \ell_{x_i})$. The perturbation is exactly zero when the faulted term does not become the new maximum.

Property 1 (Local Containment). *Let* $p^* = \max_i p_i$ *and* $g_k = p^* - p_k$. *A single-bit fault on term* k *is absorbed whenever* $\delta_b \leq g_k$. *Under uniform targeting, the containment rate is at least* $(n-1)/n$.

Property 2 (Dual-Condition Requirement). *A fault changes the predicted class only when* (C1) *it propagates through the max-reduction* and (C2) *the resulting logit shift exceeds the classification margin* m.

These are structural observations under the single-bit model, not general security guarantees. Section 6 validates these bounds experimentally.

4 Format-Dependent Fault Behavior

4.1 Fault Sensitivity of MX Encodings

As Figure 2 shows, an SE fault corrupts all k elements, while an element fault corrupts one value. The impact of an element fault depends on how the format decodes magnitude and how it accumulates partial products.

Magnitude sensitivity. A bit flip in MXINT8 shifts the decoded value by an integer step proportional to the bit position, so the error grows linearly. MXFP8-E4M3 has 12 effective exponent bits (4 element + 8 shared); a single flip can shift magnitude by 2^1 to 2^8, so the error grows exponentially. MXFP8-E5M2 widens the element exponent to 5 bits and reduces the mantissa to 2 bits; this extends range but makes each mantissa bit more influential under fault. LOG8-SUM and LOG8-MAX encode values in log space; the worst-case fractional-bit flip produces $\leq 1.84\times$ error ($2^{7/8} \approx 1.84$), bounded by the 3-bit fractional field.

Reduction path. MXINT8, MXFP8-E4M3, MXFP8-E5M2, and LOG8-SUM accumulate through summation, so every corrupted term contributes to the final output. LOG8-MAX replaces summation with max-reduction, where only the largest term determines the output; a faulted term that remains below the maximum therefore has no effect on the result.

4.2 Shared-Exponent Amplification

As Figure 4(a) illustrates, an SE bit flip rescales all k block values by $2^{\pm\Delta}$ at once. This $k\times$ fan-out distinguishes MX from per-element formats, where a fault perturbs only one weight. The worst-case amplification depends on how each format decodes the shared exponent. MXFP8-E4M3 compounds the SE shift with its 4-bit element exponent, reaching up to 65,536×. MXFP8-E5M2 reaches 32,768× because its wider exponent range maps the same SE flip to a larger decoded value. LOG8-SUM amplifies up to 256× when a flip hits the integer part of the log field. MXINT8 limits amplification to 254× because its linear decode contains no element exponent. LOG8-MAX has the same SE vulnerability as LOG8-SUM, but its max-reduction filters the error at the element level (Section 4.4).

4.3 Attention Dataflow and Fault Paths

The attention mechanism [6] projects input X through weight matrices W_Q, W_K, W_V and computes:

$$Q = XW_Q,\ K = XW_K,\ V = XW_V \tag{4}$$

$$\mathrm{Attn} = \mathrm{softmax}(QK^T/\sqrt{d_k})\,V \tag{5}$$

Figure 3 shows how CAPRI1 executes this dataflow with MX-encoded weights stored in SRAM. The softmax output determines which tokens contribute to each position. A fault in W_Q or W_K corrupts QK^T and distorts this distribution. In MXFP8-E4M3, a single exponent-field flip can shift a key magnitude by 2× to 64×; this is large enough to invert the softmax output, causing the model to route Token 0's information to Token 1's position and vice versa. We call this failure mode *attention routing inversion*. The output projection W_O maps context directly to logits without any normalizing layer, so faults in W_O pass to the classifier without attenuation.

4.4 Log-Domain Fault Filtering

In LOG8-MAX, the max-reduction operator selects only the largest term from each dot product, as Figure 4(b) illustrates. A fault on a non-winning term does not change the output because the corrupted value remains below the current maximum. For k=4, at least 75% of element faults target non-winners and produce no effect on the result. A misclassification therefore requires two conditions: the faulted term must overtake the winner, *and* the resulting logit shift must exceed the classification margin. Summation-based formats lack both conditions because every corrupted term shifts the accumulated output and propagates to the classifier.

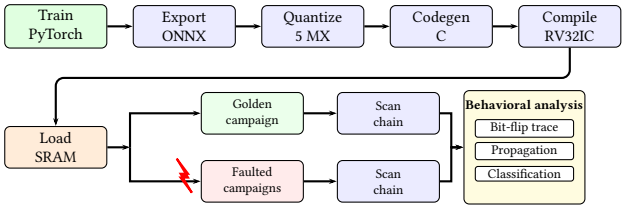

Figure 5: End-to-end fault injection and analysis pipeline, common to gate-level simulation and fabricated CAPRI1 chip. Golden and faulted campaigns run independently; scan-chain captures feed behavioral analysis that traces each bit flip from initial corruption to classification impact.

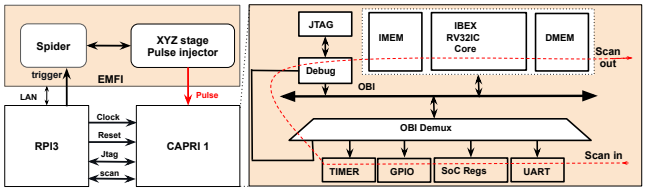

Figure 6: CAPRI1 SoC architecture. The Ibex RV32IC core executes format-specific attention inference from DFF-based SRAM. MX weights (35 bytes) reside at known DMEM addresses for targeted single-bit fault injection. The 12,756-FF scan chain captures complete internal state every 50 cycles.

5 Experiments

5.1 Experimental Flow

Figure 5 shows the end-to-end fault injection pipeline for evaluating all five MX formats on the CAPRI1 platform. The pipeline begins by training a model in PyTorch and exporting it to ONNX. The ONNX model is then quantized into five MX encodings from *identical* FP32 values (k=4). Format-specific C implementations are cross-compiled to RV32IC binaries and loaded into SRAM, placing the weight block at known addresses. Each fault campaign flips one weight bit before execution, and a matching golden run provides the baseline. The 12,756-FF scan chain captures complete processor state every 50 cycles during the golden run and the faulted run. Hamming distance between these two snapshots measures fault propagation over time. This behavioral analysis traces each bit flip from initial corruption through intermediate state disturbance to final classification impact.

This pipeline executes on two platforms. *Gate-level simulation* with SDF timing runs all four workloads and provides deterministic single-bit control over every weight byte. The *fabricated CAPRI1 chip*, shown in Figure 6, validates the primary workload on silicon and confirms simulation predictions. CAPRI1 is a 0.18 μm RISC-V SoC (2×2 mm^2, Ibex CV32E20, RV32IC, 1.8 V) with 1 KB DFF-based SRAM and a 12,756-FF scan chain. For the primary workload, EMFI experiments use a coil probe positioned 1 mm above the die on a motorized XYZ stage, delivering ±20 V pulses of 4–8 ns width. Each format undergoes 280 EMFI campaigns (35 bytes × 8 bits). Table 2 compares simulation and CAPRI1 results.

5.2 Workloads

The primary workload is a single-head attention binary classifier sized to fit CAPRI1's 1 KB memory while preserving every stage relevant to fault analysis: projection (W_Q, W_K, W_V, 4×2 each), softmax routing, value aggregation, and output projection (W_O, 2×2). Totals: 28 weights, 35 packed bytes per format. All five encodings are quantized from identical FP32 values, ensuring any resilience difference is attributable solely to format choice. The architecture and fault paths follow Figure 3. Table 2 compares simulation and CAPRI1 EMFI results for this workload across all five formats.

Table 2: Tiny Attention: simulation vs. CAPRI1 EMFI. 280 single-bit SRAM faults per format (35 bytes × 8 bits).

	Misclass.		Rate (%)	
Format	SIM	CAPRI1	SIM	CAPRI1
MXINT8	3/280	1/280	1.1	0.4
MXFP8-E4M3	**25/280**	**9/280**	**8.9**	**3.2**
MXFP8-E5M2	6/280	5/280	2.1	1.8
LOG8-SUM	3/280	1/280	1.1	0.4
LOG8-MAX	**0/280**	1/280	**0.0**	0.4

Three additional workloads extend coverage via gate-level simulation: a MoE attention classifier (28 weights), an MLP anomaly detector (114 parameters), and a keyword-spotting attention model (KWS, 106 parameters, 4 tokens). All are quantized identically to the primary workload. Table 3 reports cycle counts and available fault data across all four.

5.3 Metrics

Three metrics quantify each format's fault behavior. *Misclassification rate* counts silent failures where the faulted output differs from the golden output (Tables 2 and 3). *Cycle count* measures execution cost per format per workload (Table 3). *Scan-chain Hamming distance* tracks fault propagation from initial bit flip to final state, capturing intermediate disturbance and self-healing (Table 8).

Of the 280 faults per format on the primary workload, 56 target shared-exponent bytes (7 SE × 8 bits) and 224 target element bytes (28 elements × 8 bits). This decomposition enables direct measurement of SE amplification (Table 6). Energy estimates use 180 nm operation costs from [11] (Table 1) and are not measured on chip.

6 Results

This section begins with the fault attack summary on workloads (Section 6.1), narrows to per-format ranking on the primary workload (Section 6.2), and traces the disparity to three mechanisms (Sections 6.3–6.5).

6.1 Fault Attack Summary on Workloads

Table 3 summarizes fault counts across all four workloads. LOG8-MAX and MXINT8 achieve the lowest misclassification counts on three of four workloads. Vulnerability rankings vary by workload: MXFP8-E4M3 is most vulnerable on the primary workload (25/280) but has zero misclassifications on E2. The following subsections analyze the primary workload (Exp 1: Tiny Attention) in detail.

Table 3: Cross-workload comparison on CAPRI1 (rv32ic, 20 MHz). Cycles from gate-level simulation with SDF timing; E1 additionally validated on fabricated silicon. Faults: misclassifications from single-bit SRAM injection.

	Cycles			Faults			
Format	E1/2	E3	E4	E1	E2	E3	E4
LOG8-MAX	**1,622**	**2,615**	**17,695**	**0**	12	5	0
LOG8-SUM	3,985	8,028	39,017	3	20	6	8
MXINT8	4,256	10,854	36,612	3	10	**0**	**1**
MXFP8-E5M2	5,407	11,850	32,207	6	21	20	24
MXFP8-E4M3	5,511	12,122	34,603	25	0	22	5

E1: Self-Attention (28 wt, /280). E2: Mixture-of-Experts Attention (28 wt, /280).
E3: Multi-Layer Perceptron (114 p, /912). E4: Keyword-Spotting Attention (106 p, /848).
E1/2: same weight count; cycles shown once.

6.2 Vulnerability Ranking

Table 4 ranks all five formats on the primary workload. MXFP8-E4M3 is 8× more vulnerable than MXINT8 (8.9% vs 1.1%), while LOG8-MAX achieves zero misclassifications. The two floating-point formats account for 31 of 37 total misclassifications (83.8%); MXINT8 and LOG8-SUM each contribute 3. Table 5 breaks this down by weight matrix and confirms that the ranking holds across all four projection matrices.

Table 4: Misclassification vulnerability across five MX formats (Exp 1). Each format undergoes 280 single-bit SRAM faults (35 bytes × 8 bits).

Rank	Format	Misclass.	Rate (%)	Dominant mechanism
1	**LOG8-MAX**	**0/280**	**0.0%**	Self-healing
2	MXINT8	3/280	1.1%	MSB-only
2	LOG8-SUM	3/280	1.1%	W_V mid-bit
4	MXFP8-E5M2	6/280	2.1%	W_V/W_O
5	MXFP8-E4M3	25/280	**8.9%**	Routing inversion

Table 5: Vulnerability heatmap: misclassification count by format and weight matrix (Exp 1). MXFP8-E4M3's W_O (9/40 fault sites) is the most fragile; LOG8-MAX is uniformly zero.

	SE	W_Q	W_K	W_V	W_O
LOG8-MAX	**0**	**0**	**0**	**0**	**0**
MXINT8	1	**0**	**0**	**0**	2
LOG8-SUM	**0**	**0**	**0**	3	**0**
MXFP8-E5M2	2	**0**	**0**	3	1
MXFP8-E4M3	6	3	5	2	**9**

0 | 1–2 | 3–5 | 6–8 | ≥9

6.3 Shared-Exponent Amplification

Table 4 identifies *which* formats are vulnerable; the next three subsections trace *why*. Table 6 separates misclassifications into SE faults (56 per format) and element faults (224 per format). MXINT8 and MXFP8-E5M2 both exhibit 2.0× SE amplification: per byte, an SE fault is twice as likely to cause misclassification as an element fault. MXFP8-E4M3 has lower amplification (1.3×) because its element faults are also highly damaging due to the two-level exponent. On Exp 1, LOG8-SUM and LOG8-MAX produce zero SE misclassifications; the log encoding bounds SE-induced magnitude shifts.

Table 6: Shared-exponent (SE) amplification. SE: 7 bytes × 8 bits = 56 faults. Element: 28 bytes × 8 bits = 224 faults.

Format	SE misclass.	SE rate	Elem misclass.	Elem rate	Amplif.
MXINT8	1/56	1.8%	2/224	0.9%	**2.0×**
MXFP8-E5M2	2/56	3.6%	4/224	1.8%	**2.0×**
MXFP8-E4M3	6/56	10.7%	19/224	8.5%	**1.3×**
LOG8-SUM	0/56	0.0%	3/224	1.3%	0
LOG8-MAX	0/56	0.0%	0/224	0.0%	N/A

6.4 Attention Routing Inversion

The second mechanism targets the attention dataflow in Figure 3. Table 7 classifies each misclassification by corruption type. MXFP8-E4M3 alone exhibits **severe routing inversion**: eight faults in W_K (5) and W_Q (3) invert the 2×2 attention matrix from $\left[\begin{smallmatrix}1 & 254\\ 254 & 1\end{smallmatrix}\right]$ to $\left[\begin{smallmatrix}254 & 1\\ 1 & 254\end{smallmatrix}\right]$ (out of 256), routing Token 0's information to Token 1's position. MXFP8-E4M3's 4-bit element exponent shifts a decoded key magnitude by 2× to 64× per bit flip, sufficient to invert the softmax output. No other format produces this failure: MXINT8 lacks an element exponent, MXFP8-E5M2's wider range absorbs single-bit shifts, and the log encoding in LOG8-SUM/LOG8-MAX bounds error magnitude.

Table 7: Corruption type classification. Severe routing: attention shift ≥253/256. Only MXFP8-E4M3 exhibits routing inversion.

Format	Severe rout.	Mild rout.	Value corr.	Output corr.	Total
MXINT8	0	0	0	2	3[†]
MXFP8-E5M2	0	0	3	3	6
MXFP8-E4M3	**8**	1	2	14	25
LOG8-SUM	0	0	3	0	3
LOG8-MAX	0	0	0	0	0

[†]1 MXINT8 fault causes misclass. without detectable intermediate corruption.

W_O is the most fragile matrix in MXFP8-E4M3: 14 of 25 misclassifications (56%) originate from its 40 fault sites (35% per-bit vulnerability). W_O maps context directly to logits without softmax normalization, so errors pass to the classifier without attenuation.

6.5 Log-Domain Self-Healing

The third mechanism is specific to LOG8-MAX. Table 8 lists fault propagation metrics for all five formats. In LOG8-MAX, **88 of 280 faulted runs (31.4%) converge back to the exact golden state by program completion**, despite initial pipeline disturbance. This *self-healing* differs from error masking: the internal state returns to the exact golden bit pattern, confirmed by zero Hamming distance at the final scan-chain capture.

Table 8: Fault propagation summary (mean across 280 faults per format). Self-heal = faults where final HD returns to zero.

Format	Peak HD	Final HD	Affected FFs	Self-healed	Misclass.
MXINT8	125	90	303	0	3
LOG8-SUM	148	105	370	0	3
LOG8-MAX	182	118	431	**88**	**0**
MXFP8-E5M2	288	232	737	0	6
MXFP8-E4M3	313	256	782	0	25

Two mechanisms reinforce self-healing. Log-domain multiplication (addition of 7-bit integers) bounds the worst-case perturbation to $2^{0.5} \approx 1.41\times$ per fractional-bit flip. Integer softmax normalization (sum to 256) then absorbs these bounded errors during attention-row renormalization. Together, bounded perturbation and softmax absorption produce convergence despite substantial intermediate disturbance (mean peak HD = 182). LOG8-SUM shares the same encoding but accumulates through summation, so every faulted term shifts the output and no self-healing occurs. MXFP8-E4M3 has the worst propagation (mean peak HD = 313, 782 affected FFs) because its two-level exponent creates unbounded error growth that softmax cannot absorb.

Table 9 validates the predicted containment bounds from Section 3 against measured values on the primary workload. Deviations remain within 2.3% across all five formats.

Table 9: Predicted vs. measured fault containment (Δz_j=0), Exp 1.

Format	Predicted	Measured	Δ
MXINT8	0.125^a	0.130	0.005
MXFP8-E4M3	$\sim 0.50^b$	0.506	0.006
MXFP8-E5M2	$\sim 0.30^b$	0.323	0.023
LOG8-SUM	0.125^a	0.132	0.007
LOG8-MAX	$\geq 0.971^c$	**0.982**	0.011

[a]Zero-weight fraction k_0/n. [b]Includes NaN-clamping. [c]Property 1.

6.6 MXFP8-E4M3 Failure Mode Analysis

Among the 25 MXFP8-E4M3 misclassifications, large exponent shifts dominate (9, 36%), followed by mantissa value changes (6, 24%), sign flips (4, 16%), and SE overflow/underflow (6, 24%). The exponent field dominates: 52% originate from bits 4–7.

7 Discussion

Scope and limitations. The primary workload has 28 weights and produces a binary classification. The 95% Clopper–Pearson upper bound on LOG8-MAX misclassification rate is 1.3% (0/280). Containment and self-healing are structural properties of the max-reduction operator, but accuracy on larger models requires separate validation.

Format choice as a security parameter. MX element encoding creates an 8× vulnerability disparity under identical fault conditions (Table 4). The disparity originates from each encoding's decode arithmetic: linear error growth (MXINT8), exponential error growth (MXFP8-E4M3), and bounded log-domain error (LOG8-SUM/LOG8-MAX). Format selection in safety-critical edge deployments must account for fault resilience alongside accuracy and efficiency.

Speed, energy, and resilience trade-off. LOG8-MAX leads speed (2–4.6×, Table 3) and energy (Table 1), but its max-reduction limits golden accuracy to 50% on the multi-token attention workload. MXINT8 achieves the strongest resilience (0.12% on Exp 4) with full golden accuracy. LOG8-SUM balances all three axes: multiply-free (∼5 cyc/MAC), accurate on attention workloads, and 0.66%–7.1% fault rate. LOG8-MAX is best suited to MAC-heavy layers where one product dominates (e.g., attention routing), but its accuracy degrades on layers that require accurate summation of many equal-magnitude terms. LOG8-SUM avoids this limitation by converting back to linear before summation, retaining full accuracy at moderate cost (∼5 cyc/MAC). No single format dominates; the choice depends on the deployment priority.

Implications for MX hardware design. MXFP8-E4M3 is widely adopted yet the most vulnerable format tested. Three design responses are available: (1) LOG8-MAX as an alternative when the max-reduction approximation is acceptable, (2) targeted hardening of W_O (the most fragile matrix across all formats), and (3) attention weight distribution monitoring for fault detection. Breier et al. [15] identify INT8 as the most resilient standard format under EMFI; the MX comparison here extends that finding with shared-exponent amplification as an additional fault vector absent from per-element formats.

8 Related Work

MX hardware. MXDOTP [8] and VMXDOTP [9] add MX dot-product instructions to RISC-V cores; Cuyckens et al. [10] build precision-scalable MX hardware for robotics. None evaluate fault behavior.

DNN fault attacks. Rakin et al. [2] target bit flips in FP32 weights; Yao et al. [3] extend this to multi-layer chains. Hong et al. [4] quantify degradation under random upsets, and Breier et al. [5] inject faults into fixed-point accelerators. Breier et al. [15] compare four standard formats under EMFI and find that floating-point formats collapse while integer formats remain functional. None of these studies examines MX block-scaled formats, where shared-exponent amplification introduces a distinct fault mechanism.

Table 10: Comparison with prior DNN fault injection studies. MX = microscaling block formats. SE = shared-exponent analysis.

Work	Formats	MX	SE	Platform	Model	Faults
Rakin [2]	FP32	×	×	Sim.	ResNet	Targeted
Hong [4]	FP32	×	×	Sim.	DNN	Random
Yao [3]	FP32	×	×	DRAM	DNN	Rowhammer
Breier [5]	INT8	×	×	FPGA	DNN	Laser/EMFI
Breier [15]	4 std	×	×	SRAM	CNN	EMFI
Reagen [14]	FP32	×	×	Sim.	DNN	Random
This work	**5 MX**	✓	✓	**ASIC + Sim**	**Attn./MoE/MLP**	**Exhaustive**

Fault tolerance. TMR, ABFT, and Razor protect at area or latency cost; Unicorn-CIM [12] adds ECC with 8.98% overhead. LOG8-MAX's max operator serves as both reduction and fault filter at zero additional hardware cost.

Log-number systems. LNS-Madam [13] replaces multipliers with log-domain adders but retains linear-domain summation. Dally [7] proposes the LOG8 encoding adopted here; LOG8-MAX removes the conversion bottleneck via max-reduction and gains fault containment as a structural byproduct. Table 10 positions this study against prior work.

9 Conclusion

Arithmetic structure, not just bit width, is a security parameter in MX accelerator design. This paper presents the first fault analysis of five MX-format encodings on a RISC-V SoC, covering four workloads with the primary workload validated on fabricated silicon. Format choice alone creates substantial vulnerability disparity through three MX-specific mechanisms: block-shared exponent amplification, attention routing inversion, and log-domain fault filtering. No single format dominates all axes: LOG8-MAX leads speed (2–4.6×) and energy, MXINT8 leads resilience (0.12%) with full accuracy, and LOG8-SUM balances all three. Future work will expand post-silicon EMFI beyond the primary workload and evaluate larger models.

References

[1] B. Darvish Rouhani et al. OCP Microscaling formats (MX) specification, v1.0. Open Compute Project, 2023.

[2] A. S. Rakin, Z. He, and D. Fan. Bit-flip attack: Crushing neural network with progressive bit search. In *Proc. ICCV*, 2019, pp. 1211–1220.

[3] F. Yao, A. S. Rakin, and D. Fan. DeepHammer: Depleting the intelligence of deep neural networks through targeted chain of bit flips. In *Proc. USENIX Security*, 2020, pp. 1463–1480.

[4] S. Hong, P. Frigo, Y. Kaya, C. Giuffrida, and T. Dumitraş. Terminal brain damage: Exposing the graceless degradation in DNNs under hardware fault attacks. In *Proc. USENIX Security*, 2019, pp. 497–514.

[5] J. Breier, X. Hou, D. Jap, L. Ma, S. Bhasin, and Y. Liu. Practical fault attack on deep neural networks. In *Proc. ACM CCS*, 2018, pp. 2204–2206.

[6] A. Vaswani et al. Attention is all you need. In *Proc. NeurIPS*, 2017.

[7] W. J. Dally. Trends in deep learning hardware: Log number systems for energy-efficient inference. Invited talk, Stanford SystemX Alliance, 2024.

[8] G. Islamoglu et al. MXDOTP: A RISC-V ISA extension for microscaling (MX) dot products. In *Proc. IEEE ASAP*, 2025, pp. 81–84.

[9] M. Wipfli et al. VMXDOTP: A RISC-V vector ISA extension for microscaling (MX) format acceleration. In *Proc. DATE*, 2026.

[10] S. Cuyckens, X. Yi, N. Satya Murthy, C. Fang, and M. Verhelst. Efficient precision-scalable hardware for microscaling (MX) processing in robotics learning. In *Proc. ISLPED*, 2025.

[11] M. Horowitz. Computing's energy problem (and what we can do about it). In *Proc. ISSCC*, 2014, pp. 10–14.

[12] Q. Li, Y. Liang, and W. Cao. Unicorn-CIM: Uncovering the vulnerability and improving the resilience of high-precision compute-in-memory. In *Proc. ISLPED*, 2025, pp. 1–6.

[13] J. Zhao, S. Dai, R. Venkatesan, B. Zimmer, M. F. Ali, M.-Y. Liu, B. Khailany, W. J. Dally, and A. Anandkumar. LNS-Madam: Low-precision training in logarithmic number system using multiplicative weight update. *IEEE Trans. Comput.*, vol. 71, no. 12, pp. 3179–3190, 2022.

[14] B. Reagen et al. Ares: A framework for quantifying the resilience of deep neural networks. In *Proc. DAC*, 2018, pp. 1–6.

[15] J. Breier, Š. Kučerák, and X. Hou. The weight of a bit: EMFI sensitivity analysis of embedded deep learning models. *arXiv:2602.16309*, Feb. 2026.

[16] G. Agosta, A. Galimberti, and D. Zoni. Deep learning on RISC-V platforms at the edge: A perspective on the hardware and software support. *ACM Comput. Surv.*, vol. 58, no. 5, art. 121, 2025.