

The AetherFloat Family: Block-Scale-Free Quad-Radix Floating-Point Architectures for AI Accelerators

Keita Morisaki

Independent Hardware/Software Co-Design Researcher
Shinagawa City, Tokyo, Japan
kmta1236@gmail.com

Abstract—The IEEE 754 floating-point standard is the bedrock of modern computing, but its structural requirements—a hidden leading bit, Base-2 bit-level normalization, and Sign-Magnitude encoding—impose significant silicon area and power overhead in massively parallel Neural Processing Units (NPUs). Deep logarithmic barrel shifters expand standard cell area, and microcode traps for subnormal numbers stall pipelines. Furthermore, the industry’s recent shift to 8-bit formats (e.g., FP8 E4M3, OCP MX formats) has introduced a new hardware penalty: the strict necessity of Block-Scaling (AMAX) logic to prevent out-of-bound Large Language Model (LLM) activations from overflowing and degrading accuracy.

The AetherFloat Family is a parameterizable architectural replacement designed from first principles for Hardware/Software Co-Design in AI acceleration. By synthesizing Lexicographic One’s Complement Unpacking, Quad-Radix (Base-4) Scaling, and an Explicit Mantissa, AetherFloat achieves zero-cycle native integer comparability, branchless subnormal handling, and a verified 33.17% area, 21.99% total power, and 11.73% critical path delay reduction across the multiply-accumulate (MAC) unit. Instantiated as AetherFloat-8 (AF8), the architecture relies on a purely explicit 3-bit mantissa. By intentionally trading an implicit bit of precision to eliminate the hidden bit, the hardware multiplier array reduces from 4×4 to 3×3 . Combined with Base-4 scaling, AF8 delivers a substantially wider dynamic range ($\approx 1.22 \times 10^{-4}$ to 57,344 in the preferred hardware-optimized embodiment; up to 229,376 in the mathematically idealized configuration; vs FP8’s $\approx 10^{-2}$ to 448), acting as a “Block-Scale-Free” format for inference that circumvents dynamic scaling microarchitecture. Finally, a novel Vector-Shared 32-bit Galois Stochastic Rounding topology bounds precision variance while neutralizing the vanishing gradients that plague legacy formats. While AF16 serves as a near-lossless bfloat16 replacement via post-training quantization, AF8 is designed as a QAT-first inference format: its Block-Scale-Free property eliminates dynamic AMAX hardware at the cost of requiring quantization-aware fine-tuning for deployment.

Index Terms—Floating-Point Architecture, AI Accelerators, Hardware/Software Co-Design, Quantization, Neural Processing Units.

Disclosure: Patent pending (U.S. Provisional App. No. 63/987,398 & subsequent filings). The simulation framework and Verilog code are provided under an Academic Evaluation License for reproducibility.

I. INTRODUCTION

The IEEE 754 floating-point standard [1] has defined numerical computation for decades. However, its translation to massively parallel Neural Processing Units (NPUs) and Tensor Processing architectures incurs significant silicon area and power overhead. Deep alignment crossbars required by Base-2 normalization expand die space, and traditional logarithmic floating-point multiplier designs struggle to balance hardware efficiency with precision.

Furthermore, as recent deep learning research demonstrates, Large Language Model (LLM) activations systematically exhibit massive outliers. The industry’s recent shift to narrow 8-bit formats (e.g., FP8 [2], OCP MX [3]) and emerging sub-8-bit architectures struggles to natively absorb these outliers due to restricted dynamic ranges, making complex Block-Scaling (AMAX) hardware mandatory to prevent catastrophic overflow. Post-training quantization methods address this at the algorithmic level through weight-level or codebook-based compensation, but fundamentally do not eliminate the underlying hardware penalties.

II. RELATED WORK

High-Radix Floating-Point Arithmetic. The use of high-radix exponent bases in floating-point formats dates to the IBM System/360 hexadecimal (Base-16) architecture [4]. Sweeney [5] analyzed the precision variance (“wobble”) inherent in hex normalization, which contributed to the eventual industry migration to Base-2 in IEEE 754 [1]. AetherFloat revisits high-radix scaling in the specific context of deep learning accelerators, where stochastic gradient descent absorbs bounded precision variance, a property unavailable to the general-purpose workloads that motivated Base-2 standardization.

Alternative Number Systems. Gustafson and Yonemoto [6] proposed the Posit number system, which uses tapered precision via a variable-length regime field to maximize accuracy near unity. Posits target a fundamentally different design goal (dynamic precision allocation) than AetherFloat’s focus on hardware datapath simplification

through explicit mantissas, branchless subnormals, and block-scale elimination.

Low-Precision Training and Quantization.

Gupta et al. [7] established that stochastic rounding enables convergence in reduced-precision training, a result extended to 8-bit floating-point by Wang et al. [8]. The Straight-Through Estimator (STE), introduced by Bengio et al. [9], enables gradient flow through non-differentiable quantization boundaries and forms the basis of AetherFloat’s QAT datapath. Nagel et al. [10] provide a comprehensive survey of neural network quantization techniques.

LLM Quantization and Activation Outliers. The challenge of activation outliers in large language models has driven significant algorithmic work. Dettmers et al. [11] demonstrated mixed-precision decomposition to isolate outlier channels. SmoothQuant [12] migrates quantization difficulty from activations to weights via per-channel scaling. GPTQ [13] and AWQ [14] address weight quantization through second-order and activation-aware methods, respectively. These approaches operate at the algorithmic level and are complementary to AetherFloat’s hardware-level approach, which expands native dynamic range to absorb outliers without per-tensor or per-block scaling circuitry.

III. CORE ARCHITECTURAL INNOVATIONS

The AetherFloat Family deviates from legacy floating-point via three synthesized structural optimizations, applicable across any N -bit precision variant:

A. Lexicographic One’s Complement Unpacking

Standard Sign-Magnitude encoding breaks integer comparability. Negative floating-point numbers sort inversely to positive ones, requiring dedicated FPU logic for critical nonlinearities like ReLU ($\max(0, x)$). AetherFloat integrates an order-preserving integer encoding mapping at the hardware level. While One’s Complement representation has historical precedent in integer architectures, its systematic application to floating-point encoding for native hardware comparability in NPU datapaths has not, to our knowledge, been previously explored in the context of AI accelerator design. While Two’s Complement encoding is standard for integer ALUs, it inherently requires carry-propagation logic for sign inversion, which degrades critical path timing. Conversely, One’s Complement restricts the conversion to a single gate delay (bitwise XOR) without carry propagation, optimizing for the ultra-low latency constraints of FPU bypass networks.

AetherFloat utilizes a fixed-field Sign-Magnitude layout, but strictly bitwise-inverts the magnitude bits of negative values (One’s Complement). This yields native monotonic signed-integer lexicographical comparability without introducing multi-cycle adder latencies. The FPU extracts the unsigned $(N - 1)$ -bit Magnitude (U) in a single gate delay via a parallel row of XOR gates driven by the broadcasted sign bit; the one’s complement operation reduces to a bitwise inversion, implementable in a single logic gate. As validated by our C++ hardware emulation, a 1,000,012-element array naturally

sorted using standard integer ALUs exhibited exactly zero monotonicity errors, demonstrating constant-latency integer comparability and zero-cycle FPU bypass.

B. Quad-Radix (Base-4) Scaling and Precision Variance

Legacy formats require deep 4-stage logarithmic barrel crossbars to align Base-2 operands. AetherFloat natively scales the exponent E in Base-4 (powers of 4). Base-4 is specifically selected over higher radices (e.g., Base-8 or Base-16) to balance dynamic range expansion for LLM outlier absorption against the resulting precision variance penalty.

- Operand alignment shifts occur strictly in 2-bit pairs.
- The deep bit-level crossbar is replaced by an ultra-shallow, highly efficient 2-stage Multiplexer.
- **Precision Variance (“Wobble”):** High-radix floating-point formats introduce precision variance (“wobble”) during multi-bit normalization—a well-documented phenomenon since the IBM System/360 hexadecimal architecture [4], [5]. However, in the context of deep learning, the stochastic gradient descent (SGD) optimization landscape differs fundamentally from general-purpose computation. Prior work has demonstrated that stochastic rounding preserves gradient expectations and absorbs quantization noise in low-precision training [7], [8]. Building upon this foundation, our empirical evaluations indicate that the measured 3.04 dB mean SQNR penalty of Quad-Radix scaling relative to Base-2 is absorbed during training without measurable accuracy degradation at the 16-bit scale (see Fig. 1).

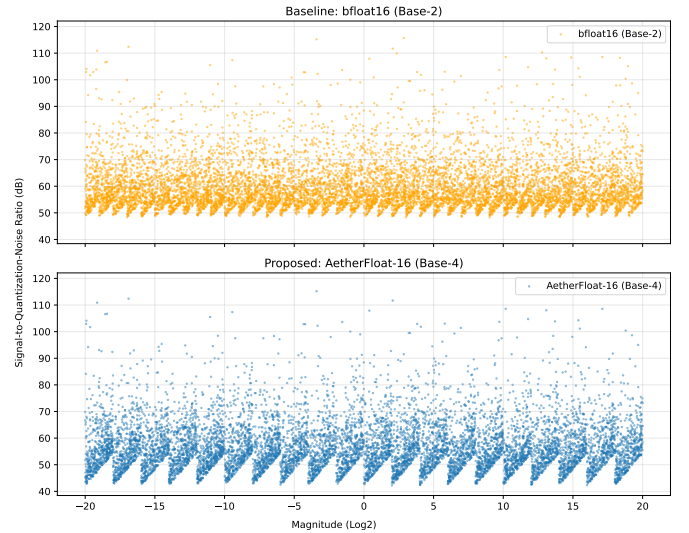


Fig. 1. Wobble Cost Analysis (SQNR). The Quad-Radix (Base-4) scaling introduces a measured mean SQNR penalty of ≈ 3.04 dB relative to Base-2 bfloat16, acting as benign regularization during stochastic gradient descent.

C. Explicit Mantissa & Unified Datapath (No-Trap Subnormals)

AetherFloat abandons the “hidden bit.” The mantissa is explicitly stored.

- **Normal Numbers** ($E > 0$): Hardware enforces that the leading 2-bit pair is non-zero (01, 10, or 11).
- **Branchless Subnormals**: When the exponent hits 0, the hardware simply suspends the leading-pair rule. Subnormals flow through the identical hardware multiplier array, MUX, and adder datapath natively, systematically eliminating the microcode traps and pipeline stalls that hinder standard FPUs.

The Hardware-Software Precision Trade-off: Legacy Base-2 formats like FP8 E4M3 utilize a hidden leading bit, meaning a 3-bit stored mantissa fundamentally requires a 4×4 -bit hardware multiplier. Because AetherFloat-8 utilizes a strictly explicit 3-bit mantissa, it intentionally trades an implicit bit of mathematical precision to shrink the physical silicon footprint. The MAC unit operates on a native 3×3 -bit **partial product array**. This compounding architectural co-design choice fundamentally drives the 33.17% area reduction across the MAC datapath.

IV. THE PARAMETERIZED FORMAT FAMILY & BIT LAYOUT

An AetherFloat number is encoded as a standard N -bit signed integer to achieve zero-latency hardware unpacking:

```
1 // High-Level Behavioral Model: Zero-latency hardware
  unpack logic for an N-bit AetherFloat
2 // X is provided as an N-bit signed integer
3 intN_t mask = X >> (N - 1);
4 uintN_t U = (X ^ mask) & (((uintN_t)1 << (N - 1)) - 1);
5 uintN_t S = ((uintN_t)X >> (N - 1)); // Extract Sign Bit
```

A. AetherFloat-8 (AF8): Block-Scale-Free Inference

A primary microarchitectural challenge in modern 8-bit inference is the limited dynamic range of FP8 E4M3 (≈ 448.0) [2], which necessitates dynamic block-scaling (AMAX) hardware to prevent overflow from LLM activation outliers [11], [12]. Because AF8 uses Base-4 scaling, its exponent multiplier grows exponentially faster. The term “Block-Scale-Free” refers specifically to the elimination of runtime AMAX scaling hardware; AF8 requires offline quantization-aware training (QAT) for deployment (Section VII).

- **Sign (S):** 1 bit
- **Exponent (E):** 4 bits. Base-4 scaling (Bias 7). Dynamic Range: $\approx 1.22 \times 10^{-4}$ to 57,344.¹
- **Explicit Mantissa (M):** 3 bits.
- **One-Step Underflow:** Because AF8 explicitly maps 3 mantissa bits, suspending the “non-zero leading pair” rule for subnormals forces the top two bits to 00. Therefore,

¹In a first, mathematically idealized embodiment, the full exponent range yields a contiguous domain up to 229,376 (i.e., $M=7$, value = $(7/2) \times 4^8$) with a subnormal floor of $\approx 1.22 \times 10^{-4}$ (i.e., $M=1$ at 2^{-13}). In the preferred hardware-optimized embodiment described herein, the maximum exponent state ($E = 15$) is reserved for NaN/Inf exception traps, capping the practical normal limit at 57,344, and the minimum hardware quantum is defined via the explicit mantissa subnormal rule ($M = 1$) at $\approx 1.22 \times 10^{-4}$. Both configurations represent valid architectural mappings depending on the target accelerator’s pipeline exception-handling requirements.

$M \in \{0, 1\}$. AF8 supports exactly **one** non-zero subnormal quantum ($M = 1$), acting as a highly efficient branchless one-step underflow without pipeline traps.

Contrast with OCP Microscaling (MX): Recent industry consortiums have championed OCP MX formats [3] to solve the FP8 outlier crisis by amortizing a single Base-2 block-exponent across a vector of activations. This introduces additional hardware complexity: architectures must stall to extract block maximums (AMAX), and a massive outlier forces the shared exponent to scale up, quantizing neighboring smaller values to zero. AF8 bypasses the strict need for dynamic shared block-exponents entirely. However, to support hybrid architectures, the high-radix Base-4 exponent can still be optionally amortized across a memory-block, combining AetherFloat’s scale-free properties with block-level exponent clustering.

B. AetherFloat-16 (AF16): The bfloat16 Replacement

AF16 achieves an equivalent practical macro-range to Google’s bfloat16 [15] utilizing a coarser exponent grid and explicit subnormals.

- **Sign (S):** 1 bit
- **Exponent (E):** 7 bits. Base-4 scaling (Bias 63). Dynamic Range: $\approx 10^{-38}$ to $\approx 10^{38}$.
- **Explicit Mantissa (M):** 8 bits.

V. MATHEMATICAL ENCODING RULES

Let M be the integer interpretation of the explicitly stored mantissa bits. For AF16, $M \in [0, 255]$ with the radix point implicitly placed after the top 2 bits (division by 64.0). For AF8, $M \in [0, 7]$ with the radix point implicitly placed after the top 2 bits (division by 2.0).

A. Rule 1: Normal Numbers ($E > 0$)

- **Constraint:** The leading 2-bit pair of M must be non-zero.
- **Value (AF16):** $(-1)^S \times (M/64.0) \times 4^{(E-63)}$
- **Value (AF8):** $(-1)^S \times (M/2.0) \times 4^{(E-7)}$

B. Rule 2: Subnormals ($E = 0$)

- **Constraint:** The normalization rule is suspended.
- **Value (AF16):** $(-1)^S \times M \times 2^{-130}$
- **Value (AF8):** $(-1)^S \times M \times 2^{-13}$, providing a minimum subnormal quantum of $2^{-13} \approx 1.22 \times 10^{-4}$. Because $E = 0$ and $E = 1$ yield the same mantissa scale, a unified branchless subnormal datapath is possible without additional MUX stages.

C. Area-Optimized Vector-Shared Stochastic Rounding

As 8-bit training crucially relies on stochastic rounding (SR) to preserve small gradient updates [7], [8], a robust PRNG is required. The non-trivial hardware cost of per-element random number generation in reduced-precision training is a known constraint. To eliminate the silicon penalty of per-ALU random number generation, a single 32-bit Galois LFSR sits at the SIMD vector-lane level and broadcasts a random bit-vector to

TABLE I
AETHERFLOAT FORMAT SPECIFICATIONS

Format	Base	Bias	Mantissa Scale	Normal Constraint	Subnormal Range	Special Values
AF16	4	63	/64.0	Leading Pair \neq 00	$M < 64$	Extremity Sorting
AF8	4	7	/2.0	Leading Pair \neq 00	$M \in \{0, 1\}$	Extremity Sorting

an array of MAC units (e.g., 1 PRNG per 16 MACs). Our ablations (Fig. 2) demonstrate this structurally amortized topology natively prevents vanishing gradients without inducing over-correlation collapse. In QAT forward passes, stochastic rounding is disabled (deterministic mode) to mirror physical hardware inference behavior. During the backward pass, stochastic rounding is applied to gradient accumulations using the vector-shared topology (as evaluated in the ablation study in Fig. 2). These two roles are distinct: the LFSR-based SR hardware is an *on-device training* circuit designed to stochastically quantize gradient accumulations during backpropagation. Standard inference deployment is purely deterministic; therefore, during software QAT simulations, the forward pass is deterministic, while the Straight-Through Estimator (STE) combined with backward-pass stochastic rounding enables simulated gradient flow across the non-differentiable boundary.

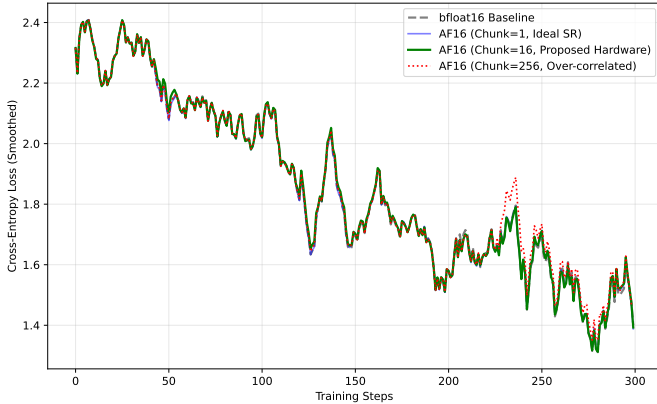


Fig. 2. Vector-Shared Stochastic Rounding Ablation. Simulating Qwen2.5-7B next-token prediction confirms our Chunk=16 configuration effectively tracks ideal independent SR while amortizing hardware costs.

VI. NATIVE LEXICOGRAPHICAL COMPARABILITY

By applying the One’s Complement global sign wrapper, all AetherFloat variations achieve monotonic integer sorting. Special values utilize the maximum exponent, natively sorting to the absolute extremities of the integer domain (e.g., -NaN sorts to -32,768).

AI Workload Impact:

- 1) **Zero-Cycle FPU Bypass:** Critical non-linearities like ReLU ($\max(0, x)$) or Max-Pooling can be executed natively on cheap, integer-only SIMD ALUs.
- 2) **Branchless Fault Tolerance:** Tensors can be thresholded using native integer operations to filter out NaNs, entirely bypassing costly `isnan()` branching.

VII. EMPIRICAL EVALUATION & COMPARATIVE ANALYSIS

A. LLM Accuracy & Boundary Conditions

PyTorch Post-Training Quantization (PTQ) simulations against Qwen2.5-7B were executed across standard language tasks (Table II). All AetherFloat quantization and dequantization operations employ precision upcasting to IEEE 754 float64 intermediate representation prior to computing quantization boundaries, ensuring bit-exact correspondence with native hardware behavior and eliminating double-rounding artifacts at subnormal transitions.

Analysis & Architectural Boundary Conditions:

- 1) **AF16 Near-Parity:** AF16 behaves as a near-lossless drop-in replacement for bfloat16 (+0.0012 PPL on WikiText-2; +0.0009 HellaSwag Acc), demonstrating that the Quad-Radix precision variance acts benignly at the 16-bit scale.
- 2) **The 8-bit Hardware/Software Trade-off:** FP8 maintains strong PTQ performance strictly because it relies on dynamic AMAX block-scaling hardware. AF8 intentionally eliminates this hardware entirely to remain Block-Scale-Free. Consequently, AF8 is architecturally a *QAT-first* format: its deployment path requires quantization-aware fine-tuning rather than drop-in PTQ.
- 3) **Gradient Underflow Discovery:** As observed in Table II, applying AF8 natively in a pure PTQ environment results in measurable degradation. Because AF8 strictly avoids AMAX block-scaling, small converged LLM weights routinely drop below its absolute minimum positive value ($\approx 1.22 \times 10^{-4}$), causing them to flush to zero.

Therefore, we recommend AF8 strictly as an Inference format deployed via Quantization-Aware Training (QAT). Using the Straight-Through Estimator (STE) [9], which bypasses the non-differentiable quantizer during backpropagation, AF8 demonstrates viable gradient flow throughout training (Figure 3). In this initial 200-step feasibility run, both formats exhibit transient loss spikes; however, the FP8 baseline demonstrates greater late-training instability (loss ≈ 3.8 at step 150) compared to AF8’s stronger recovery (loss ≈ 3.0 at step 150, vs. BF16 baseline ≈ 2.8). Within this single-seed feasibility window, AF8 shows favorable convergence behavior relative to FP8 by step 150; however, multi-seed trials with extended training horizons are required to establish statistical significance. Future work will also explore integrating activation-aware weight quantization algorithms (e.g., AWQ [14]) to protect salient weights while relying on AF8’s native range to absorb outliers.

TABLE II
ZERO-SHOT BENCHMARKS FOR PTQ ON QWEN2.5-7B

Metric	BF16 Baseline	AF16 (Ours)	FP8 E4M3 + AMAX	AF8 (Ours)
WikiText-2 PPL (\downarrow)	8.7368	8.7380	8.7951	10.6389
PIQA Acc (\uparrow)	0.7884	0.7878	0.7894	0.7775
HellaSwag Acc (\uparrow)	0.5990	0.5999	0.5975	0.5545

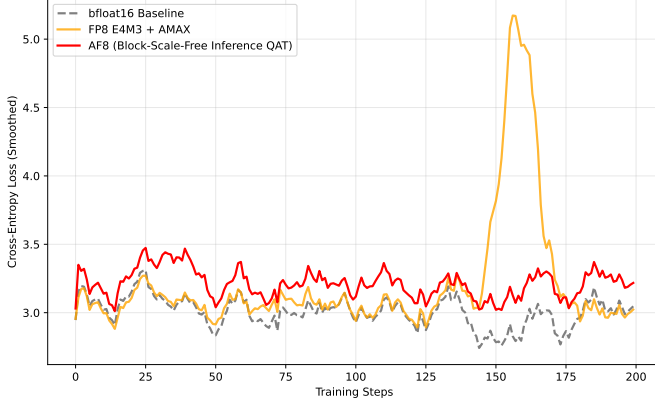


Fig. 3. 8-bit QAT Convergence (Qwen2.5-7B: AF8 vs FP8). Both formats exhibit transient loss spikes, with FP8 showing greater mid-to-late-training instability. AF8 demonstrates favorable convergence behavior by step 150 (single seed), confirming viable STE gradient flow without AMAX logic.

B. Hardware Impact Summary

Verilog datapaths were synthesized via Yosys and mapped to the open-source SkyWater 130nm PDK. Power was estimated via OpenSTA using a verified 20% global toggle/static probability rate. As seen in Table III, trading one bit of mantissa precision to eliminate the hidden bit yields substantial standard cell area reductions.

C. Limitations & Future Work

This study has several limitations that scope the conclusions above. First, all hardware metrics are derived from the SkyWater 130nm educational PDK; production FinFET nodes (e.g., TSMC 5nm) may shift absolute area and power figures, although relative trends are expected to hold. Second, all software evaluations use PyTorch emulation rather than native hardware execution; while precision upcasting ensures bit-exact quantization boundaries, end-to-end system-level effects (e.g., memory bandwidth, pipeline interactions) remain unevaluated. Third, the QAT convergence results (Fig. 3) are based on a single-seed 200-step feasibility run on one model (Qwen2.5-7B); multi-seed, full-epoch training across diverse architectures (e.g., vision transformers, mixture-of-experts) is required to establish statistical significance and broader external validity. Fourth, the AF8 PTQ gap (Table II) confirms that AF8 is not a drop-in PTQ replacement; real-world deployment requires quantization-aware fine-tuning, and the cost of that retraining has not yet been benchmarked. Fifth, the construct validity of the precision variance hypothesis (SQNR 3.04 dB wobble acting as benign regularization) has been empirically

validated but not yet supported by a formal convergence-bound proof. Finally, Lexicographic One’s Complement encoding has not been evaluated against adversarial numerical edge cases (e.g., catastrophic cancellation in long summation chains).

VIII. CONCLUSION

The AetherFloat Family provides a design point that combines the comparability properties of signed integers with the dynamic range requirements of deep learning. By accepting a measured mean SQNR variance (≈ 3.04 dB mean penalty) via Quad-Radix scaling and strategically trading a single bit of mantissa precision to eliminate the hidden bit, it addresses the major bottlenecks of legacy floating-point: deep alignment crossbars, hidden-bit pipeline traps, FPU comparison latency, and 8-bit Block-Scaling overhead.

Empirical evaluation using the SkyWater 130nm PDK verifies that bypassing the hidden bit shrinks the hardware multiplier, yielding a **33.17% area** and **21.99% total power** reduction across the MAC core. Furthermore, PyTorch simulations against Qwen2.5-7B validate that AetherFloat-8 operates as a “Block-Scale-Free” inference format that eliminates dynamic AMAX hardware, natively accommodating LLM activation outliers within its expanded dynamic range and demonstrating viable gradient flow under QAT via STE [9], with favorable convergence behavior relative to the FP8 baseline within the feasibility run. This hardware simplification comes at the cost of requiring quantization-aware fine-tuning rather than drop-in post-training quantization.

CODE AVAILABILITY & IP DISCLOSURE

To facilitate peer review while preserving commercial intellectual property rights, the complete PyTorch hardware-software simulation suite, open-source SkyWater 130nm PDK Verilog generation, and C++ emulation frameworks are publicly available under an **Academic Evaluation License** at: <https://github.com/k1832/aetherfloat-validation> (and within the Computer Program Listing Appendix of the corresponding patent specification). Commercial deployment, hardware synthesis integration, or utilization in proprietary architectures strictly requires a commercial IP license.

REFERENCES

- [1] IEEE, “IEEE Standard for Floating-Point Arithmetic,” *IEEE Std 754-2019*, pp. 1–84, Jul. 2019.
- [2] P. Micikevicius *et al.*, “FP8 formats for deep learning,” *arXiv preprint arXiv:2209.05433*, 2022.
- [3] B. D. Rouhani *et al.*, “OCP Microscaling Formats (MX) Specification Version 1.0,” *Open Compute Project*, Sep. 2023. [Online]. Available: <https://www.opencompute.org/documents/ocp-microscaling-formats-mx-v1-0-spec-final-pdf>

TABLE III
HARDWARE EXECUTION OVERHEAD (TOTAL MAC UNIT)

Feature	FP8 E4M3 (Baseline)	AF8 (Ours)	Delta
Multiplier Footprint	4×4 bit array	3×3 bit array*	Smaller
Total MAC Area	1018.48 μm^2	680.65 μm^2	-33.17%
Critical Path Delay	2426.30 ps	2141.60 ps	-11.73%
Total MAC Power	84.60 μW	66.00 μW	-21.99%[†]
Area\timesDelay Product	2,471,138 $\mu\text{m}^2 \cdot \text{ps}$	1,457,680 $\mu\text{m}^2 \cdot \text{ps}$	-41.01%
Block-Scaling	Required (AMAX ALU)	Scale-Free	Eliminated
Comparability	FPU Logic Required	Zero-Cycle (ALU)	Native

*Note: The FP8 baseline requires a 4×4 multiplier array to accommodate its hidden bit. AF8 utilizes an explicit 3-bit mantissa, intentionally trading 1 bit of strict mathematical precision to reduce the physical silicon multiplier array area. These metrics reflect the multiply-accumulate datapath in isolation; encode/decode logic, exception handling, and dynamic AMAX circuitry are excluded from comparison. Excluding AMAX circuitry yields a conservative baseline, as a system-level comparison including AMAX would further highlight the area reductions of the scale-free AF8 architecture.

[†]The 21.99% reduction reflects total MAC power (dynamic switching + static leakage) as formally verified by OpenSTA. Isolated dynamic toggle-rate analysis yields a higher reduction of 34.89%; the total power metric is reported throughout this paper as the conservative, silicon-representative figure.

- [4] G. M. Amdahl, G. A. Blaauw, and F. P. Brooks, Jr., "Architecture of the IBM System/360," *IBM J. Res. Dev.*, vol. 8, no. 2, pp. 87–101, Apr. 1964.
- [5] D. W. Sweeney, "An analysis of floating-point addition," *IBM Syst. J.*, vol. 4, no. 1, pp. 31–42, 1965.
- [6] J. L. Gustafson and I. T. Yonemoto, "Beating Floating Point at its Own Game: Posit Arithmetic," *Supercomput. Front. Innov.*, vol. 4, no. 2, pp. 71–86, 2017.
- [7] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep Learning with Limited Numerical Precision," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2015, pp. 1737–1746. arXiv:1502.02551.
- [8] N. Wang *et al.*, "Training deep neural networks with 8-bit floating point numbers," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 31, 2018. arXiv:1812.08011.
- [9] Y. Bengio, N. Léonard, and A. Courville, "Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation," *arXiv preprint arXiv:1308.3432*, 2013.
- [10] M. Nagel, M. Fournarakis, R. A. Amjad, Y. Bondarenko, M. van Baalen, and T. Blankevoort, "A White Paper on Neural Network Quantization," *arXiv preprint arXiv:2106.08295*, 2021.
- [11] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, "LLM.int8(): 8-bit matrix multiplication for transformers at scale," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 35, 2022. arXiv:2208.07339.
- [12] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han, "SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2023. arXiv:2211.10438.
- [13] E. Frantar, S. Ashkboos, T. Hoefer, and D. Alistarh, "GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2023. arXiv:2210.17323.
- [14] J. Lin *et al.*, "AWQ: Activation-aware Weight Quantization for On-Device LLM Compression and Acceleration," in *Proc. Mach. Learn. Syst. (MLSys)*, 2024. arXiv:2306.00978.
- [15] S. Wang and P. Kanwar, "BFloat16: The secret to high performance on Cloud TPUs," *Google Cloud Blog*, Aug. 2019. [Online]. Available: <https://cloud.google.com/blog/products/ai-machine-learning/bfloat16-the-secret-to-high-performance-on-cloud-tpus>