

Hardware architectures for posit arithmetic: evolution, trade-offs, and artificial intelligence

Sollapura Kumarswamy Pradeepkumar¹, Masthigondanahally Channabasavaiah Chandrashekhar²

¹Sri Siddhartha Academy of Higher Education, Tumakuru, India

²Department of Electronics and Communications, Sri Siddhartha Academy of Higher Education, Tumakuru, India

Article Info

Article history:

Received Feb 24, 2026

Revised Jun 4, 2026

Accepted Sep 11, 2026

Keywords:

Approximate computing

Compute-in-memory

Computer arithmetic

Deep learning

Hardware accelerators

RISC-V

Tapered precision

ABSTRACT

The IEEE 754 floating-point standard has been the backbone of scientific computing for decades. However, the rapid growth of deep learning (DL) and edge computing has highlighted its drawbacks, mainly in high memory bandwidth and energy usage. To solve this, the posit number system was introduced in 2017. It uses "tapered precision" to dynamically allocate more bits to the "Golden Zone," where artificial intelligence (AI) weights usually fall. Although theoretically better, early posit hardware faced a major "hardware tax" because decoding variable-length regime bits was too slow. This paper presents a detailed survey of posit hardware architectures from 2017 to 2025. We trace its evolution from basic scalar generators to advanced vectorized accelerators and compute-in-memory (CIM) designs. We highlight major improvements, such as leading difference detectors (LDD) and Radix-4 dividers, which have successfully reduced decoding delay by over 40% and energy consumption by 80%. Our comparative study shows that 16-bit posits can effectively replace 32-bit IEEE floats for memory-bound tasks. It saves 50% memory bandwidth without losing accuracy in applications like ResNet inference and climate modeling. Finally, we conclude that with recent RISC-V posit vector units (PVUs), posits demonstrate competitive potential relative to BFLOAT16 and FP8 formats, particularly in memory-bound edge applications.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Sollapura Kumarswamy Pradeepkumar
Sri Siddhartha Academy of Higher Education
Tumakuru-572107, Karnataka, India
Email: pradeepsk13@gmail.com

1. INTRODUCTION

Representing real numbers efficiently is a fundamental requirement in digital computing systems. For more than thirty years, the IEEE 754 standard has acted as the universal framework for floating-point arithmetic, making sure numerical calculations are consistent across different platforms [1], [2]. However, the fixed bit-field allocation of standard floating-point numbers creates major inefficiencies when dealing with modern data-intensive applications. For example, in deep learning (DL), it is observed that the wide dynamic range of 32-bit floats (FP32) is mostly wasted. This happens because neural network weights normally cluster narrowly around zero, which has led to the use of truncated formats like BFLOAT16 [3]. On the other hand, in computational physics, low-precision IEEE formats do not have the information density needed to capture very small changes, leading to problems like "periodic orbit collapse" in chaotic weather simulations [4].

To overcome these structural inefficiencies, Gustafson [5] introduced posit arithmetic in 2017, which was an upgrade from his earlier work on Unum computing [6], [7]. The main feature of the posit

format is tapered precision, which is achieved through a run-length encoded "regime" field. Unlike IEEE 754, where the sizes of the exponent and fraction are fixed during design, posits change this trade-off dynamically. Values near ± 1 are stored with short regime fields, which leaves more bits for the fractional part. Because of this encoding method, a 32-bit posit can provide the dynamic range of a 64-bit float while giving better precision than a 32-bit float for normalized values [8]. Along with this, the standard requires the use of a Quire, which is a large fixed-point exact accumulator. The Quire allows fused dot products to be calculated without catastrophic cancellation errors in linear algebra tasks [9]. Even with these mathematical benefits, people have hesitated to adopt posits because designing the hardware is very complex. Extracting the variable-length regime field requires serial logic components like leading zero detectors (LZD) and dynamic barrel shifters. This usually creates a much longer critical path compared to the simple fixed-field extraction used in IEEE floats [10]. Early studies called this problem the "hardware tax" of posits, pointing out that the extra decoding logic could take up significantly more chip area than standard floating-point units [11]. However, research carried out in the last two years has successfully solved many of these issues. The focus has shifted from just proving that posits work to actively optimizing them for system-level integration. Some of the major recent developments include leading difference detectors (LDD) [12] and radix-4 digit-recurrence dividers [13]. These designs have drastically reduced the delay and power consumption of posit arithmetic. Furthermore, by moving from isolated scalar co-processors to unified RISC-V pipelines [14] and posit vector units (PVUs) [15], researchers have achieved high-throughput SIMD execution. Recent circuit designs have also shown that posit-based compute-in-memory (CIM) macros can achieve an efficiency of 83.23 TFLOPS/W [16].

Existing reviews, such as the recent comprehensive work by Mallasén *et al.* [17], thoroughly document the mathematical properties, software ecosystems, and theoretical error bounds of the posit format. However, there is a distinct gap in the current literature regarding its physical silicon realization. Specifically, how have VLSI architects successfully mitigated the variable-length decoding penalty (historically termed the 'hardware tax') to make real-world deployment feasible? This paper distinguishes itself from recent literature by focusing exclusively on this physical architectural evolution from 2017 to 2025. By tracing the progression from early scalar prototypes to advanced RISC-V PVUs and CIM macros, we provide digital designers with a practical, physically-grounded guide for deploying posit hardware under strict edge artificial intelligence (AI) constraints. To explicitly highlight this novelty, Table 1 contrasts the scope of our survey against prominent existing reviews, demonstrating our unique emphasis on physical VLSI evolution.

Table 1. Comparison of this work with existing posit surveys and reviews

Reference	Year	Primary scope	Hardware/VLSI focus	Edge AI/IoT focus
Dinechin <i>et al.</i> [11]	2019	Mathematical critique and early hardware cost	Moderate	None
Chien <i>et al.</i> [8]	2020	HPC software and error bounds	None	Low
Mallasén <i>et al.</i> [17]	2025	General posit ecosystem and math	Low	Low
This work	2026	Architectural evolution and mitigation	High	High

2. RESEARCH METHOD

We conducted a systematic literature search utilizing IEEE Xplore, Scopus, and the ACM Digital Library. The primary search string used across all databases was: ("posit arithmetic" OR "posit number system") AND ("hardware" OR "VLSI" OR "FPGA" OR "ASIC" OR "RISC-V" OR "compute-in-memory"), which initially yielded 154 raw records. After removing 12 duplicate entries, 142 unique publications underwent a two-stage screening process. The simplified screening stages are illustrated in Figure 1.

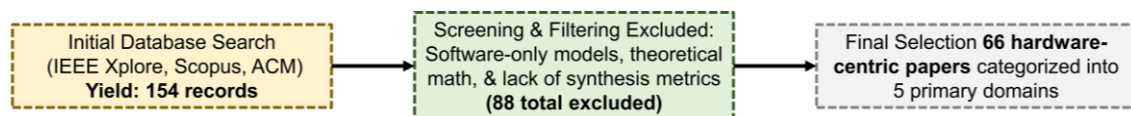


Figure 1. Literature screening diagram detailing the selection of 66 hardware-centric posit architectures

During the title and abstract screening stage, 45 papers focusing exclusively on software simulation or theoretical mathematics were excluded. In the full text screening stage, remaining 97 papers were

evaluated against strict inclusion criteria: i) presentation of a physical hardware implementations (FPGA or ASIC) and ii) reporting quantifiable physical metrics (area, delay, power, or frequency). This resulted in exclusion of 31 papers and final 66 eligible papers were subsequently grouped into five primary domains based on their core architectural contribution: initial hardware generators, advanced decoding optimizations, system-level AI integration, application-specific deployments, and the software-compiler ecosystem.

3. RESULTS AND DISCUSSION

The following subsections detail our findings from the literature, mapping out how posit hardware progressed from early prototype stages into today's heavily optimized designs.

3.1. The hardware tax and early generators (2017–2021)

Converting the posit standard [18] into synthesizable logic introduced a critical challenge: unlike IEEE 754's fixed bit-fields, posits require dynamic field extraction. This architectural penalty in early designs was termed the "hardware tax" by de Dinechin *et al.* [11]. To understand why decoding is complex, the mathematical formulation is important. A posit number is determined by its Sign (S), Regime (R), Exponent (E), and Fraction (F) fields. The architectural mechanism that allows this dynamic range is visualized in Figure 2. Specifically, Figure 2(a) illustrates the precision curve and Figure 2(b) shows the bit layout.

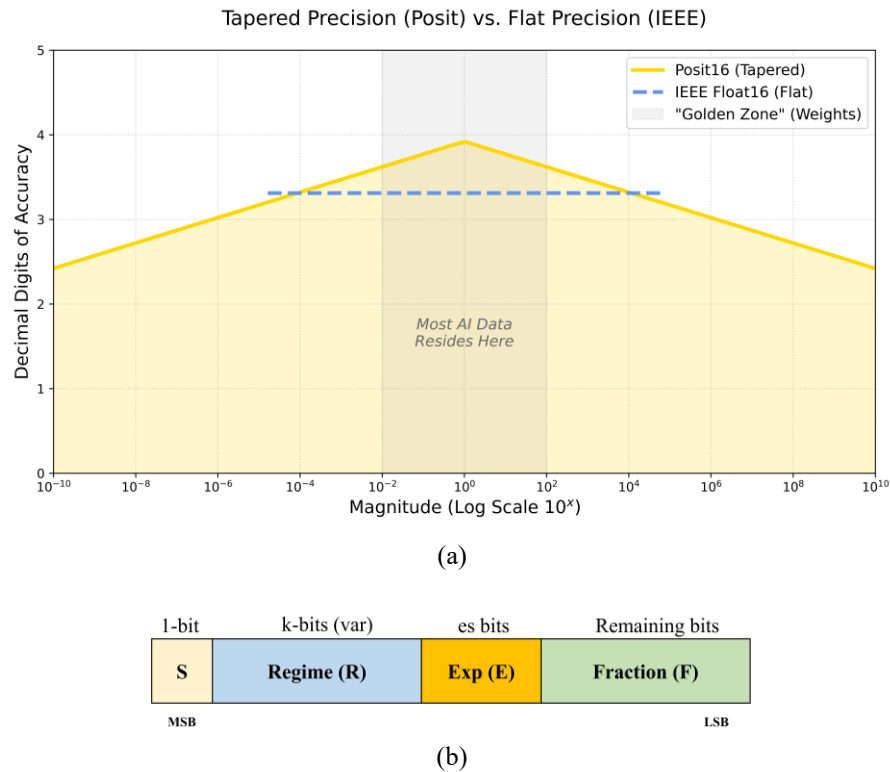


Figure 2. Tapered precision and format layout: (a) precision comparison between 32-bit posit ($es=2$) and IEEE 754 single precision and (b) bit layout of a posit number showing the variable-length regime field (adapted from [17])

The variable-length regime enables tapered precision but requires dynamic decoding shifters. For typical neural network weights (magnitudes of 10^{-1} to 10^1), 32-bit posits provide 27 significand bits (Table 2), yielding 4 more bits of precision than IEEE 754.

Table 2. The "Golden Zone" of precision (comparison of significant bits)

Magnitude range	Regime bits (k)	Fraction bits available	Precision relative to IEEE float32
-----------------	-----------------	-------------------------	------------------------------------

10^{-1} to 10^1	2	27	+4 bits (superior)
10^{-6} to 10^6	3-5	24-26	+1 to +3 bits (superior)
10^{-9} to 10^9	6	23	Equal (parity point)
$>10^{15}$	>10	<19	Inferior (trade-off)

The main source of delay in posit arithmetic comes from the Regime decoding stage. To find the numerical value, the hardware must first detect the run-length of the Regime bits using a LZD, and then shift the remaining bits using a large barrel shifter [10]. To solve this, early researchers developed parameterized hardware description language (HDL) generators like PACoGen [19] and FloPoCo-posit [20]. These generators proved that posit adders and multipliers could be successfully built on FPGAs, but they confirmed that the decoding and encoding stages took up over 30% of the total chip area [21], [22]. Following the development of early generator frameworks, researchers began targeting more complex arithmetic structures. For instance, several studies introduced power-efficient posit multipliers and multiply-accumulate (MAC) unit generators designed specifically for DL workloads [23], [24]. Around the same time, other teams developed parameterized architectures for division and square-root operations [25], as well as energy-efficient fused MAC designs [26]. However, as these arithmetic operations grew more complex, extensive design-space explorations of exact accumulators and the overall hardware cost confirmed that severe routing bottlenecks still remained [27], [28]. Another unique feature of the posit standard is the Quire, which is a massive exact accumulator (e.g., 512 bits for posit32) shown in Figure 3. Chen *et al.* [29] showed that while the Quire successfully stopped intermediate rounding errors, it consumed a huge amount of register resources. Because of this high area cost, early hardware designs often skipped the Quire, leading to a noticeable split in the literature between "Standard-Compliant" architectures and "Lightweight" architectures that persists today.

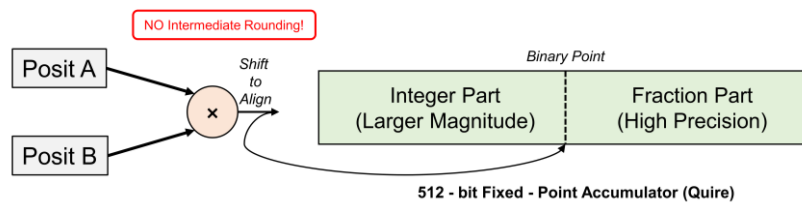


Figure 3. Structure of the Quire register (512-bit for posit32) for exact accumulation without intermediate rounding (adapted from [9])

3.2. Architectural innovations in decoding (2022–2025)

To mitigate the delay overhead associated with the "hardware tax," recent studies have prioritized gate-level optimizations. A notable advancement in this domain is the LDD [12]. Previous architectures relied on a LZD to sequentially count regime bits before initiating mantissa shifts. The LDD bypasses this serial bottleneck by calculating the shift amount in parallel with regime extraction. As detailed in Table 3, this approach achieved a 41.6% delay reduction and a 50% power reduction implemented on a 32 nm ASIC platform [12]. Division operations present another inherent hardware bottleneck due to their high latency. To resolve this, researchers have developed radix-4 digit-recurrence dividers tailored for posit arithmetic [13]. By processing two bits per clock cycle, these customized dividers yield an energy reduction of over 80% on a 28 nm ASIC [13].

Table 3 summarizes this architectural progression and outlines the primary physical trade-offs in modern posit hardware. Early designs, including the Podobas and Matsuoka baseline [10] on a Stratix-V FPGA and PACoGen [19] on a Virtex-7 FPGA, depended on sequential extraction logic, which caused deep critical paths. In contrast, recent designs have explored system-level integration. For instance, the PVU [15] integrates posits into a Zynq FPGA vector unit, utilizing 65,407 LUTs; while this SIMD integration improves AI throughput, it heavily increases datapath and scheduling complexity.

Similarly, posit CIM architectures [16] built on 28 nm CMOS achieve 419 MHz and 83.2 TFLOPS/W for high-throughput AI acceleration, though they require non-standard CIM infrastructure. Beyond the core architectural trade-offs detailed in Table 3, current literature is also expanding into broader system integrations. For example, researchers are exploring complete processor pipelines such as the PERI RISC-V core [30]. Furthermore, several studies propose optimized architectures using approximation methods for non-linear activation functions [31], while recent iterations of fused multiply-add (FMA) units explore the balance between numerical precision and silicon area [26].

Table 3. Quantitative comparison and architectural trade-offs of posit implementations (2017–2025)

Architecture and ref.	Target platform	Quantitative metrics (area/delay/power)	Key architectural trade-off or limitation
Baseline [10]	Stratix-V FPGA	~2,000 ALMs 500 MHz 210 mW	First feasibility study; identified severe sequential decoding overhead in early architectures.
LDD decoder [12]	32 nm ASIC	41.6% delay, 50% power reduction	Parallel difference detection replaces LZD, but routing complexity scales poorly beyond 32-bit precision.
Radix-4 divider [13]	28 nm ASIC	>80% energy reduction (vs. NRD)	Digit-recurrence methods speed up division but significantly increase algorithmic overhead.
Unified FPU [14]	28 nm ASIC	47.9% LUT and 57.4% FF savings	Shared datapath improves resource utilization but restricts architectural flexibility for isolated posit tuning.
PVU [15] (vector unit)	Zynq FPGA	65,407 LUTs 108 Muxes	SIMD integration improves AI throughput but heavily increases datapath and scheduling complexity.
Posit CIM [16]	28 nm CMOS	22.2 μm^2 /cell 419 MHz 83.2 TFLOPS/W	High-throughput AI acceleration, but requires non-standard CIM infrastructure.
PACoGen [19] (generator)	Virtex-7 FPGA	~1,800 LUTs ~125 MHz N/A	Open-source generator, but inherits the sequential LZD bottleneck causing deep critical paths.
FloPoCo [20]	65 nm ASIC	7,615 μm^2 7.94 ns 3.8 mW (32-bit add)	Frequency-directed pipelining improves timing, but sequential extraction dependency remains unresolved.
HUB posit [32]	45 nm ASIC	15% area-delay product (ADP) reduction	Internal routing optimized via implicit LSB, but does not eliminate the sequential extraction bottleneck.
ADEPNET [33]	90 nm ASIC	5% area, 13% power reduction (vs. baseline)	Hybrid-radix optimization improves efficiency but introduces additional control-path complexity.
Approx. mult. [34]	90 nm ASIC	14% area, 16.7% power reduction	Energy efficiency improved through truncation; potentially unsuitable for exact scientific workloads.
RISC-V Core [30]	Artix-7 FPGA	1,797 LUTs 47.4 ns (32-bit FMA)	Early processor integration; limited discussion of large-scale vectorization for AI workloads.
Fixed-posit [35]	28 nm FDSOI	38.5% area, 35.6% delay, 47% power savings	Area reduction achieved by hardwiring regime size, sacrificing dynamic precision benefits.

Note: the quantitative metrics in Table 3 are drawn from diverse studies. Because these architectures were synthesized across different platforms (e.g., FPGA vs. ASIC) and technology nodes (e.g., 28 nm vs. 90 nm), a single normalized baseline is not available. Consequently, these comparisons highlight general architectural trends rather than direct, one-to-one performance equivalents.

3.3. System-level integration and artificial intelligence accelerators

While optimizing individual ALUs was a necessary first step, the ultimate goal for the posit format is its deployment in large-scale computing and AI accelerators. In the last two years, the focus has moved from standalone mathematical units to full system-on-chip (SoC) environments. Because of its open-source nature, the RISC-V instruction set architecture has become the standard platform for testing posit hardware [36]. Several recent works have successfully integrated PVUs directly into RISC-V pipelines. Figure 4 shows a PVU integrated alongside a standard integer core, enabling efficient single instruction, multiple data (SIMD) execution for DL matrix multiplications. In AI, memory bandwidth often bottlenecks performance more than pure compute speed. Comparative studies have shown that 16-bit posits (posit16) can effectively replace standard 32-bit floats (FP32) for memory-bound tasks like ResNet inference [37]–[43]. This substitution provides a $2\times$ reduction in memory traffic without any noticeable drop in classification accuracy. Furthermore, cutting-edge designs in 2024 and 2025 have started implementing posits inside CIM macros. By performing the MAC calculations directly inside the SRAM arrays, these CIM architectures have reported peak energy efficiencies of up to 83.23 TFLOPS/W [16]. Consequently, posits demonstrate competitive potential relative to BFLOAT16 and FP8 in selected AI and memory-bound workloads, though commercial hurdles remain.

3.4. Application-specific deployments

While early evaluations focused heavily on theoretical error bounds, recent literature has shifted toward application-specific hardware testing. Posit architectures have been successfully deployed and evaluated in digital signal processing (DSP), particularly in fast Fourier transform (FFT) accelerators, radio astronomy, weather modeling, sparse linear solvers, conjugate gradient methods, and optical flow [44]–[48]. Additionally, several researchers have explored the format's efficiency in specialized domains such as cryptography, universal coding, and comparative precision studies [49]–[51]. Across these studies, the consensus indicates that while control logic overhead remains, the reduction in memory footprint provides a net positive energy saving, particularly in optimized multipliers and activation functions [52]–[54]. Beyond high-performance computing, the reduced memory footprint of posit arithmetic makes it highly attractive for resource-constrained embedded systems, FPGA-based neural networks, and internet of things (IoT) edge devices [55]–[57]. In edge AI scenarios, such as smart sensors or wearable health monitors, memory bandwidth and SRAM leakage often dominate the total power budget [58], [59]. By utilizing 8-bit or 16-bit

posit formats, designers can cut memory traffic in half compared to standard FP32 implementations while maintaining sufficient inference accuracy, even under critical hardware faults or in genome variant calling [60], [61]. Furthermore, deploying PVUs on reconfigurable FPGA fabrics allows embedded engineers to dynamically tailor the arithmetic precision to the specific needs of the IoT application, balancing the area overhead of the extraction logic against the system's strict energy constraints [15], [56].

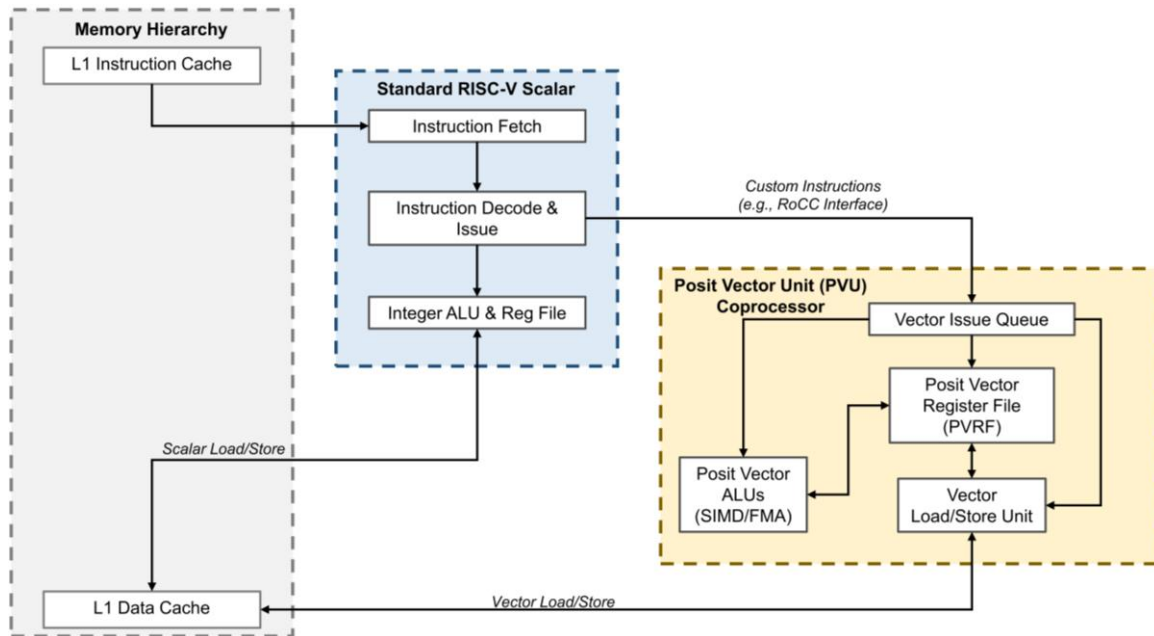


Figure 4. Integration of a PUV within a standard RISC-V processor pipeline (source: authors' original creation, illustrating the architectural concept from [15])

3.5. The software and compiler gap

Beyond physical implementation, hardware acceleration is only viable if it is supported by a robust software ecosystem. A growing body of literature has focused on developing custom C/C++ libraries, LLVM backend extensions, high-level .NET software implementations, and compiler optimization passes to map high-level code directly to posit ALUs [62]. While challenges remain in auto-vectorization and native language support, these software frameworks are critical for bridging the gap between theoretical hardware performance and practical deployment, especially for image and video processing or algorithm-hardware co-design [63], [64].

3.6. Open challenges and future directions

Despite the significant architectural milestones achieved between 2017 and 2025, the universal adoption of posit arithmetic is currently hindered by several systemic challenges that future research must address: i) verification complexity: the variable-length nature of the regime field makes formal hardware verification exceptionally difficult. Unlike IEEE 754, where datapaths are fixed, verifying all edge cases in dynamic posit shifters requires massive computational overhead during the VLSI testing phase [60]; ii) software ecosystem immaturity: as highlighted in subsection 3.5, while hardware accelerators (like PVUs) now exist, the compiler toolchains required to map high-level code (C++/Python) to these units remain immature. Without robust, auto-vectorizing compilers, software developers cannot easily utilize the underlying hardware efficiency; and iii) the FP8 ecosystem dominance: posits are competing directly against the FP8 format [65], which currently enjoys massive industry backing from NVIDIA, Intel, and ARM. To realistically compete with these established formats, future research must prove substantial energy-delay benefits in physical silicon, moving beyond initial FPGA prototypes like recent multi-precision systolic arrays [66].

4. CONCLUSION

We have systematically reviewed the evolution of posit arithmetic hardware from 2017 through 2025. Early hardware designers heavily avoided the format because of the excessive "hardware tax" caused by variable-length regime extraction. Recent gate-level innovations, specifically LDD and custom radix-4 dividers, have largely bypassed those early delay bottlenecks. We also observe a definitive architectural shift away from isolated scalar units and toward integrated RISC-V PVUs and CIM designs. In memory-bound scenarios like edge AI inference, the posit16 format effectively acts as a drop-in replacement for standard 32-bit floats. This substitution provides a direct halving of memory bandwidth while maintaining target accuracy. Despite these gains, actual universal adoption remains blocked by several hard constraints. The silicon area needed for dynamic extraction logic is still too high for extreme low-power IoT sensors. Even outside of hardware limits, the format suffers from immature software standardization, massive verification overhead for dynamic datapaths, and very little commercial deployment. Without robust, auto-vectorizing compilers, software engineers cannot easily target posit accelerators. Moving forward, the research community needs to focus on hybrid posit-IEEE systems and hardware-software compiler co-design to close this gap. Continuous VLSI advancements have certainly made posit arithmetic a physically viable competitor to BFLOAT16 and FP8. However, one limitation of this survey is that it relies on data from diverse technology nodes and platforms, making a perfectly normalized comparison difficult. Future benchmarking across a single, unified physical design flow will be needed to quantify these hardware trade-offs more accurately. Ultimately, the long-term success of the posit format depends on solving these remaining verification hurdles and building a mature software ecosystem.

FUNDING INFORMATION

Authors state no funding involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Sollapura Kumarswamy	✓	✓			✓	✓		✓	✓		✓			
Pradeepkumar														
Masthigondanahally		✓			✓		✓			✓		✓		
Channabasavaiah														
Chandrashekhar														

C : **C**onceptualization

M : **M**ethodology

So : **S**oftware

Va : **V**alidation

Fo : **F**ormal analysis

I : **I**nvestigation

R : **R**esources

D : **D**ata Curation

O : Writing - **O**riginal Draft

E : Writing - Review & **E**ditng

Vi : **V**isualization

Su : **S**upervision

P : **P**roject administration

Fu : **F**unding acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

Data availability is not applicable as no new data were created or analyzed.

REFERENCES

- [1] "IEEE Standard for Floating-Point Arithmetic," IEEE Std 754-2019, 2019, doi: 10.1109/IEEESTD.2019.8766229.
- [2] D. Goldberg, "What every computer scientist should know about floating-point arithmetic," *ACM Computing Surveys*, vol. 23, no. 1, pp. 5–48, Mar. 1991, doi: 10.1145/103162.103163.
- [3] D. Kalamkar *et al.*, "A Study of BFLOAT16 for Deep Learning Training," *arXiv*, 2019, doi: 10.48550/arXiv.1905.12322.
- [4] M. Klöwer, P. V. Coveney, E. A. Paxton, and T. N. Palmer, "Periodic orbits in chaotic systems simulated at low precision," *Scientific Reports*, vol. 13, no. 1, 2023, doi: 10.1038/s41598-023-37004-4.
- [5] J. L. Gustafson, "Posit Arithmetic," in *Conference on Next Generation Arithmetic (CoNGA 2017)*, 2017.
- [6] J. L. Gustafson, "A Radical Approach to Computation with Real Numbers," *Supercomputing Frontiers and Innovations*, vol. 3, no. 2, 2016, doi: 10.14529/jsfi160203.

- [7] J. L. Gustafson, *The End of Error: Unum Computing*. CRC Press (Chapman and Hall/CRC), 2015, doi: 10.1201/b18092.
- [8] S. W. D. Chien, I. B. Peng, and S. Markidis, "Posit NPB: Assessing the Precision Improvement in HPC Scientific Applications," in *Parallel Processing and Applied Mathematics*, Springer International Publishing, 2020, pp. 301–310, doi: 10.1007/978-3-030-43229-4_26.
- [9] P. W. Group, "Standard for Posit Arithmetic (2022)," 2022. [Online]. Available: https://posithub.org/docs/posit_standard-2.pdf.
- [10] A. Podobas and S. Matsuoka, "Hardware Implementation of POSITs and Their Application in FPGAs," in *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, May 2018, pp. 138–145, doi: 10.1109/ipdpsw.2018.00029.
- [11] F. de Dinechin, L. Forget, J.-M. Muller, and Y. Uguen, "Posits: the good, the bad and the ugly," in *Proceedings of the Conference for Next Generation Arithmetic 2019*, Mar. 2019, pp. 1–10, doi: 10.1145/3316279.3316285.
- [12] J. Sun and Y. Lao, "Efficient Data Extraction Circuit for Posit Number System: LDD-Based Posit Decoder," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 43, no. 6, pp. 1919–1923, 2024, doi: 10.1109/tcad.2023.3347295.
- [13] G. B. R. Murillo, J. Villalba-Moreno, and A. A. Del Barrio, "Digit-Recurrence Posit Division," *arXiv*, 2025, doi: 10.48550/arXiv.2511.02494.
- [14] Q. Li, C. Fang, L. Huang, J. Lin, and Z. Wang, "Enable Lightweight and Precision-Scalable Posit/IEEE-754 Arithmetic in RISC-V Cores for Transprecision Computing," *arXiv*, 2025, doi: 10.48550/arXiv.2505.19096.
- [15] X. Wu, Y. Wang, T. Zhao, J. Qin, Z. Liang, and J. Fu, "PVU: Design and Implementation of a Posit Vector Arithmetic Unit (PVU) for Enhanced Floating-Point Computing in Edge and AI Applications," *arXiv*, 2025, doi: 10.48550/arXiv.2503.01313.
- [16] Y. Wang *et al.*, "34.1 A 28nm 83.23TFLOPS/W POSIT-Based Compute-in-Memory Macro for High-Accuracy AI Applications," in *2024 IEEE International Solid-State Circuits Conference (ISSCC)*, Feb. 2024, pp. 566–568, doi: 10.1109/isscc49657.2024.10454567.
- [17] D. Mallasén, R. Murillo, G. Botella, and A. A. Del Barrio, "Navigating Posit Arithmetic: A Comprehensive Survey of Principles, Hardware, and Applications," *ACM Computing Surveys*, vol. 58, no. 5, pp. 1–36, Nov. 2025, doi: 10.1145/3772284.
- [18] Posit Working Group, "PositStandardDocumentation Release 3.2-draft," 2018. [Online]. Available: https://posithub.org/docs/posit_standard.pdf.
- [19] M. K. Jaiswal and H. K.-H. So, "PACoGen: A Hardware Posit Arithmetic Core Generator," *IEEE Access*, vol. 7, pp. 74586–74601, 2019, doi: 10.1109/access.2019.2920936.
- [20] R. Murillo, A. A. Del Barrio, and G. Botella, "Customized Posit Adders and Multipliers using the FloPoCo Core Generator," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, Oct. 2020, pp. 1–5, doi: 10.1109/iscas45731.2020.9180771.
- [21] R. Chaurasiya *et al.*, "Parameterized Posit Arithmetic Hardware Generator," in *2018 IEEE 36th International Conference on Computer Design (ICCD)*, Oct. 2018, pp. 334–341, doi: 10.1109/iccd.2018.00057.
- [22] M. K. Jaiswal and H. K.-H. So, "Architecture Generator for Type-3 Unum Posit Adder/Subtractor," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2018, pp. 1–5, doi: 10.1109/iscas.2018.8351142.
- [23] H. Zhang, J. He, and S.-B. Ko, "Efficient Posit Multiply-Accumulate Unit Generator for Deep Learning Applications," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2019, pp. 1–5, doi: 10.1109/iscas.2019.8702349.
- [24] H. Zhang and S.-B. Ko, "Design of Power Efficient Posit Multiplier," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 5, pp. 861–865, May 2020, doi: 10.1109/tcsii.2020.2980531.
- [25] A. Raveendran, S. Jean, J. Mervin, D. Vivian, and D. Selvakumar, "A Novel Parametrized Fused Division and Square-Root POSIT Arithmetic Architecture," in *2020 33rd International Conference on VLSI Design and 2020 19th International Conference on Embedded Systems (VLSID)*, Jan. 2020, pp. 207–212, doi: 10.1109/vlsid49098.2020.00053.
- [26] R. Murillo, D. Mallasén, A. A. Del Barrio, and G. Botella, "Energy-Efficient MAC Units for Fused Posit Arithmetic," in *2021 IEEE 39th International Conference on Computer Design (ICCD)*, Oct. 2021, pp. 138–145, doi: 10.1109/iccd53106.2021.00032.
- [27] Y. Uguen and F. De Dinechin, "Design-space exploration for the Kulisch accumulator," *Hyper Articles en Ligne (HAL)*, 2017.
- [28] Y. Uguen, L. Forget, and F. de Dinechin, "Evaluating the Hardware Cost of the Posit Number System," in *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*, 2019, pp. 106–113, doi: 10.1109/fpl.2019.00026.
- [29] J. Chen, Z. Al-Ars, and H. P. Hofstee, "A matrix-multiply unit for posits in reconfigurable logic leveraging (open)CAPI," in *CoNGA '18: Proceedings of the Conference for Next Generation Arithmetic*, 2018, pp. 1–5, doi: 10.1145/3190339.3190340.
- [30] S. Tiwari, N. Gala, C. Rebeiro, and V. Kamakoti, "PERI: A Configurable Posit Enabled RISC-V Core," *ACM Transactions on Architecture and Code Optimization*, vol. 18, no. 3, pp. 1–26, Apr. 2021, doi: 10.1145/3446210.
- [31] M. Cococcioni, F. Rossi, E. Ruffaldi, and S. Saponara, "Fast Approximations of Activation Functions in Deep Neural Networks when using Posit Arithmetic," *Sensors*, vol. 20, no. 5, p. 1515, Mar. 2020, doi: 10.3390/s20051515.
- [32] R. Murillo, J. Hormigo, A. A. Del Barrio, and G. Botella, "HUB Meets Posit: Arithmetic Units Implementation," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 71, no. 1, pp. 440–444, Jan. 2024, doi: 10.1109/tcsii.2023.3307488.
- [33] A. A. Jonnalagadda, U. A. Kumar, R. Thotli, S. Sardesai, S. Veeramachaneni, and S. E. Ahmed, "ADEPNET: A Dynamic-Precision Efficient Posit Multiplier for Neural Networks," *IEEE Access*, vol. 12, pp. 31036–31046, 2024, doi: 10.1109/access.2024.3369695.
- [34] H. Zhang and S.-B. Ko, "Efficient Approximate Posit Multipliers for Deep Learning Computation," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 13, no. 1, pp. 201–211, Mar. 2023, doi: 10.1109/jetcas.2022.3231642.
- [35] V. Gohil, S. Walia, J. Mekie, and M. Awasthi, "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, Oct. 2021, doi: 10.1109/tcsii.2021.3072217.
- [36] A. Bocco, Y. Durand, and F. De Dinechin, "SMURF: Scalar Multiple-precision Unum Risc-V Floating-point Accelerator for Scientific Computing," in *Proceedings of the Conference for Next Generation Arithmetic 2019*, Mar. 2019, pp. 1–8, doi: 10.1145/3316279.3316280.
- [37] P. Micikevicius *et al.*, "Mixed Precision Training," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.
- [38] J. Lu, C. Fang, M. Xu, J. Lin, and Z. Wang, "Evaluations on Deep Neural Networks Training Using Posit Number System," *IEEE Transactions on Computers*, vol. 70, no. 2, pp. 174–187, Feb. 2021, doi: 10.1109/tc.2020.2985971.
- [39] G. Raposo, P. Tomas, and N. Roma, "Positnn: Training Deep Neural Networks with Mixed Low-Precision Posit," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 7908–7912, doi: 10.1109/icassp39728.2021.9413919.

- [40] R. Murillo, A. A. Del Barrio, and G. Botella, "Deep PeNSieve: A deep learning framework based on the posit number system," *Digital Signal Processing*, vol. 102, p. 102762, 2020, doi: 10.1016/j.dsp.2020.102762.
- [41] S. H. F. Langroudi, T. Pandit, and D. Kudithipudi, "Deep Learning Inference on Embedded Devices: Fixed-Point vs Posit," in *2018 1st Workshop on Energy Efficient Machine Learning and Cognitive Computing for Embedded Applications (EMC2)*, Mar. 2018, pp. 19–23, doi: 10.1109/emc2.2018.00012.
- [42] T. Glint *et al.*, "Hardware-Software Codesign of DNN Accelerators Using Approximate Posit Multipliers," in *Proceedings of the 28th Asia and South Pacific Design Automation Conference*, Jan. 2023, pp. 469–474, doi: 10.1145/3566097.3567866.
- [43] M. S. Kim, A. A. Del Barrio, L. T. Oliveira, R. Hermida, and N. Bagherzadeh, "Efficient Mitchell's Approximate Log Multipliers for Convolutional Neural Networks," *IEEE Transactions on Computers*, vol. 68, no. 5, pp. 660–675, May 2019, doi: 10.1109/tc.2018.2880742.
- [44] T. K. Gunaratne, "Evaluation of the Use of Low Precision Floating-Point Arithmetic for Applications in Radio Astronomy," in *Next Generation Arithmetic*, Springer Nature Switzerland, 2023, pp. 155–170, doi: 10.1007/978-3-031-32180-1_10.
- [45] M. Klöwer, P. D. Düben, and T. N. Palmer, "Number Formats, Error Mitigation, and Scope for 16-Bit Arithmetics in Weather and Climate Modeling Analyzed With a Shallow Water Model," *Journal of Advances in Modeling Earth Systems*, vol. 12, no. 10, Oct. 2020, doi: 10.1029/2020ms002246.
- [46] L. Hunhold and J. Quinlan, "Evaluation of Bfloat16, Posit, and Takum Arithmetics in Sparse Linear Solvers," *Qeios*, Jan. 2025, doi: 10.32388/dlwjbj.
- [47] D. Mallasén, "Leveraging Posits for the Conjugate Gradient Method," M.S. Thesis, Complutense University of Madrid (Universidad Complutense de Madrid - UCM), Spain, 2022.
- [48] V. Saxena *et al.*, "Brightening the Optical Flow through Posit Arithmetic," in *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, Apr. 2021, pp. 463–468, doi: 10.1109/isqed51717.2021.9424360.
- [49] S. S. Da Silva, L. G. Nardo, E. Nepomuceno, J. Yudi, B. S. Rego, and J. Arias-García, "A Hardware-Efficient Chaotic PRNG Exploring Posit Arithmetic for Secure Image Encryption," *IEEE Access*, vol. 13, pp. 209813–209828, 2025, doi: 10.1109/access.2025.3642045.
- [50] P. Lindstrom, S. Lloyd, and J. Hittinger, "Universal coding of the reals: alternatives to IEEE floating point," in *Proceedings of the Conference for Next Generation Arithmetic*, Mar. 2018, pp. 1–14, doi: 10.1145/3190339.3190344.
- [51] J. Besseling and A. Renström, "A comparative study of IEEE 754 32-bit Float and Posit 32-bit floating point format on precision," B.S. Thesis, KTH Royal Institute of Technology (Kungliga Tekniska högskolan), Stockholm, Sweden, 2020.
- [52] A. A. Jonnalagadda, A. K. Uppugunduru, S. Veeramachaneni, and S. E. Ahmed, "Design of Energy Efficient Posit Multiplier," in *Proceedings of the Great Lakes Symposium on VLSI 2023*, 2023, pp. 645–651, doi: 10.1145/3583781.3590227.
- [53] A. A. Jonnalagadda, R. Thotli, S. Veeramachaneni, U. A. Kumar, and S. E. Ahmed, "Energy-Efficient Decoding and Encoding Hardware for Optimized Posit Arithmetic," *IEEE Embedded Systems Letters*, vol. 17, no. 3, pp. 131–134, 2025, doi: 10.1109/les.2024.3485002.
- [54] M. Cococcioni, F. Rossi, E. Ruffaldi, and S. Saponara, "Vectorizing posit operations on RISC-V for faster deep neural networks: experiments and comparison with ARM SVE," *Neural Computing and Applications*, vol. 33, no. 16, pp. 10575–10585, Feb. 2021, doi: 10.1007/s00521-021-05814-0.
- [55] S. Nambi, S. Ullah, S. S. Sahoo, A. Lohana, F. Merchant, and A. Kumar, "ExPAN(N)D: Exploring Posits for Efficient Artificial Neural Network Design in FPGA-Based Systems," *IEEE Access*, vol. 9, pp. 103691–103708, 2021, doi: 10.1109/access.2021.3098730.
- [56] M. Zolfagharijad, M. Kamal, A. Afzali-Khusha, and M. Pedram, "Posit Process Element for Using in Energy-Efficient DNN Accelerators," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 6, pp. 844–848, 2022, doi: 10.1109/tvlsi.2022.3165510.
- [57] Y. Wang, D. Deng, L. Liu, S. Wei, and S. Yin, "PL-NPU: An Energy-Efficient Edge-Device DNN Training Processor With Posit-Based Logarithm-Domain Computing," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 10, pp. 4042–4055, Oct. 2022, doi: 10.1109/tcsi.2022.3184115.
- [58] S. Walia, B. V. Tej, A. Kabra, J. Devnath, and J. Mekie, "Fast and Low-Power Quantized Fixed Posit High-Accuracy DNN Implementation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 1, pp. 108–111, Jan. 2022, doi: 10.1109/tvlsi.2021.3131609.
- [59] D. Mallasén, A. A. Del Barrio, and M. Prieto-Matias, "Big-PERCIVAL: Exploring the Native Use of 64-Bit Posit Arithmetic in Scientific Computing," *IEEE Transactions on Computers*, vol. 73, no. 6, pp. 1472–1485, 2024, doi: 10.1109/tc.2024.3377890.
- [60] J. E. R. Condia, J.-D. Guerrero-Balaguera, R. L. Sierra, and M. S. Reorda, "Investigating and Mitigating Critical Faults in Floating-Point and Posit Arithmetic Hardware," *IEEE Transactions on Emerging Topics in Computing*, vol. 13, no. 4, pp. 1605–1617, Oct. 2025, doi: 10.1109/tetc.2025.3615827.
- [61] T. Xu, S. Rixner, and A. L. Cox, "An FPGA Accelerator for Genome Variant Calling," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 16, no. 4, pp. 1–29, 2023, doi: 10.1145/3595297.
- [62] Z. Lehoczy, A. Retzler, R. Tóth, A. Szabó, B. Farkas, and K. Somogyi, "High-level .NET software implementations of unum type I and posit with simultaneous FPGA implementation using Hastlayer," in *Proceedings of the Conference for Next Generation Arithmetic*, Mar. 2018, pp. 1–7, doi: 10.1145/3190339.3190343.
- [63] A. Ramachandran, J. Gustafson, A. Roy, R. A. Ansari, and R. Daruwala, "PositIV: A Configurable Posit Processor Architecture for Image and Video Processing," in *2022 25th Euromicro Conference on Digital System Design (DSD)*, Aug. 2022, pp. 93–100, doi: 10.1109/dsd57027.2022.00022.
- [64] A. Ramachandran, Z. Wan, G. Jeong, J. Gustafson, and T. Krishna, "Algorithm-Hardware Co-Design of Distribution-Aware Logarithmic-Posit Encodings for Efficient DNN Inference," in *Proceedings of the 61st ACM/IEEE Design Automation Conference*, 2024, pp. 1–6, doi: 10.1145/3649329.3656544.
- [65] P. Mickevicius *et al.*, "FP8 Formats for Deep Learning," *arXiv*, 2022, doi: 10.48550/arXiv.2209.05433.
- [66] G. Raut, D. Selvakumar, P. J. Edavoor, and R. Thakur, "Multi-Precision SIMD Posit Systolic Array Accelerator for Deep Learning," in *2025 IEEE 5th International Conference on VLSI Systems, Architecture, Technology and Applications (VLSI SATA)*, May 2025, pp. 1–6, doi: 10.1109/vlsisata65374.2025.11070109.

BIOGRAPHIES OF AUTHORS

Sollapura Kumarswamy Pradeepkumar    received his B.E. degree in Electronics and Communication Engineering in 2009 and his M.Tech. degree in VLSI Design and Embedded Systems in 2011. He previously worked as a trainee engineer at the Electronics and Radar Development Establishment (LRDE), DRDO, Bangalore. Currently, he serves as an assistant professor in the Department of Electronics and Communication Engineering at Kalpataru Institute of Technology, Tiptur, Karnataka, and is a research scholar at Sri Siddhartha Academy of Higher Education, Tumakuru. His research interests include energy-efficient VLSI systems, FPGA-based design, RISC-V architectures, and computer arithmetic. He has also contributed to open-source hardware design, notably developing a 4 KB SRAM IP using the Sky130 technology node and OpenRAM compiler. He can be contacted at email: pradeepsk13@gmail.com.



Masthigondanahally Channabasavaiah Chandrashekhar    is a professor and head of the Department of Electronics and Communication Engineering at Sri Siddhartha Institute of Technology (SSIT), SSAHE, Tumakuru, Karnataka, India. He has extensive academic and research experience, with interests spanning thin film devices, low-power VLSI design, digital signal processing, and embedded systems. He actively guides research on emerging arithmetic standards and their implementation in next-generation computing architectures. He has successfully implemented two projects under INUP at IIT Bombay. He is also a member of the Academic Council of SSAHE. He can be contacted at email: mccshekhar@gmail.com.