Split-Posit Systolic Array: A Resource-Efficient Hardware Accelerator for High-Performance AI Workloads

Sneha Dandekar
IIIT Bangalore
Bangalore, Karnataka, India
dandekarsneha.mahendra@iiitb.ac.in

Arun M
IIIT Bangalore
Bangalore, Karnataka, India
Arun.M004@iiitb.ac.in

Vaishnavi Sharma
IIIT Bangalore
Bangalore, Karnataka, India
vaishnavi.sharma@iiitb.ac.in

Madhav Rao
IIIT Bangalore
Bangalore, Karnataka, India
mr@iiitb.ac.in

Abstract

Modern Deep Neural Network (DNN) and Transformer inference workloads fundamentally rely on efficient matrix multiplication units, with systolic arrays emerging as the most effective architecture. However, as model complexity and data volumes grow, the need for resource-efficient, low-memory-cost hardware becomes critical. While the recently introduced Posit number format offers a compelling alternative to IEEE floating-point by improving dynamic range and precision efficiency, its hardware cost remains prohibitive when implemented at higher precisions. To address this challenge, we propose Split-Posit Architectures, designed to emulate the precision and dynamic range benefits of higher-precision Posits at substantially reduced hardware cost. Our ASIC implementation demonstrates the practicality and scalability of the approach, achieving a 39.98% reduction in silicon area and a 46.74% improvement in energy efficiency per processing element compared to state-of-the-art (SOTA) designs. Integrating these units within a tiled 16×16 Systolic Array, the architecture delivers a peak energy efficiency of 5.37 TOPS/W (Tera-operations per second per Watt). The design has been further evaluated for larger array sizes and multiple Posit configurations, demonstrating competitive performance while maintaining comparable accuracy on standard DNN and Transformer benchmarks, establishing a promising direction for next-generation, resource-efficient AI acceleration.

CCS Concepts

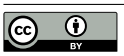
• **Hardware → Application specific integrated circuits**.

Keywords

Posit Arithmetic, Neural Networks, Systolic Arrays, General Matrix Multiplication

ACM Reference Format:

Sneha Dandekar, Arun M, Vaishnavi Sharma, and Madhav Rao. 2026. Split-Posit Systolic Array: A Resource-Efficient Hardware Accelerator for High-Performance AI Workloads. In *International Conference on Parallel Architectures and Compilation Techniques (PACT '26), October 19–22, 2026, Chicago, IL, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3838684.3846833



This work is licensed under a Creative Commons Attribution 4.0 International License.
PACT '26, Chicago, IL, USA
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2915-7/2026/10
https://doi.org/10.1145/3838684.3846833

1 Introduction

The rapid growth of Artificial Intelligence (AI) and Machine Learning (ML) has driven an unprecedented demand for data-intensive models across domains ranging from edge computing to natural language processing. Convolutional Neural Networks (CNNs) transformed visual perception by leveraging spatial locality and weight sharing, while the emergence of Transformers has dramatically expanded the capabilities of deep learning to encompass sequence modeling, multimodal understanding, and generative reasoning. However, the exponential increase in model capacity has been accompanied by a corresponding surge in data volume and computational cost. Consequently, researchers worldwide are striving to enhance resource efficiency without compromising accuracy. Despite their differing architectures and applications, both CNNs and Transformers are fundamentally constrained by the computationally intensive matrix multiplication operations that dominate their workloads, creating significant challenges in terms of energy efficiency, resource utilization, and hardware scalability. These challenges make matrix multiplication an ideal focal point for designing next-generation AI accelerators.

Traditional hardware accelerators, such as systolic arrays [39], have long provided an elegant means to exploit data reuse in dense matrix multiplications. Although its advent dates back decades, their high throughput and predictable performance continue to make systolic arrays a cornerstone in modern AI hardware, inspiring several successive generations [1, 33] built on the same architectural fabric. However, as neural network models grow larger and more irregular, particularly with multi-head attention and structured sparsity in Transformers, the conventional monolithic systolic array faces challenges such as under-utilization, load imbalance, and excessive data movement. Likewise, CNN accelerators optimized for uniform convolution workloads struggle to efficiently adapt to handle heterogeneous layer dimensions and mixed-precision formats emerging in modern models [25, 31]. These limitations underscore the need to rethink the core hardware design and move toward architectures that can dynamically adapt to the evolving computational characteristics of diverse neural workloads.

While Convolutional Neural Networks (CNNs) have been remarkably effective, they inherently involve a large number of parameters and computationally intensive operations. For instance, DenseNet with 264 layers requires approximately 12.5 billion FLOPs and a memory bandwidth of about 400 GB/s [27]. A similar trend is observed in Transformer architectures, which exhibit even higher memory footprints and energy demands. For example, ViT-Large contains 307M parameters and performs 190.7 GFLOPs to achieve 87.76% Top-1 accuracy on ImageNet [13]. These challenges necessitate innovative and effective model compression techniques to balance accuracy and efficiency. While techniques like pruning [17, 19] and dimension reduction [37, 40], quantization has seemed to emerge as most dominant approach [29, 41, 42].

Quantization transforms weights and/or activations from floating point to lower precision representations without altering the network architecture. Over the years, numerous studies have analyzed various numeric formats. J. Lee et al. [28] investigated floating-point to low-bit mappings guided by statistical analysis, while B. Crafton et al. [10] extended this by providing a MAC architecture analysis using a heuristic approach, considering emerging formats such as Microscaling (MX) and Logarithmic numbers (LNS16). Among these, integer quantization remains the most effective in terms of memory cost and area footprint, but it suffers from limited dynamic range and numerical inaccuracy [9, 12, 14, 21, 22, 26]. Since the distribution of neural activations and weights is typically centered around zero, quantization often leads to rounding-to-zero errors that degrade model accuracy. In large-scale deep networks, these quantization errors accumulate across layers, significantly affecting inference quality [15]. Although mixed-precision solutions have been proposed to mitigate this issue, they introduce additional complexity, conversion overhead, and under-utilization of memory bandwidth, making them suboptimal for resource-constrained AI accelerators [4, 30, 35, 43].

Addressing these concerns is where the recently introduced Posit format comes into play [16]. With its adaptive and tapered precision properties, it offers a compelling alternative to conventional quantization schemes [18]. Several studies have demonstrated that 16-bit Posit arithmetic achieves accuracy comparable to Floating-point (FP32), while delivering improved energy efficiency and reduced error sensitivity [5–7, 20, 34]. Conversely, smaller formats such as 8-bit Posits often exhibit irregular convergence behavior, failing to match the accuracy of higher-precision counterparts. However, the hardware implementation of 16-bit Posits remains computationally expensive, primarily due to the large and complex decoder logic required [24, 32]. The study in [11] further identifies variable-length regime decoding as a major contributor to Posit hardware complexity. Since the boundaries between the regime, exponent, and mantissa fields vary with the operand value, conventional implementations require dedicated logic to dynamically decode and reconstruct these fields, resulting in considerable hardware overhead. This observation motivates architectures that can reduce the cost of repeated variable-length decoding through operand pre-processing and efficient resource sharing, rather than simply reducing the operand bitwidth.

To address these challenges, we propose the Split-Posit Architecture, designed to replicate the precision and dynamic range advantages of higher-precision Posits at substantially reduced hardware cost. By decomposing higher-bit Posit computations into multiple lower-bit-width operations, the proposed architecture achieves comparable computational accuracy while significantly reducing decoder complexity, power consumption, and silicon area. The major contributions of this work are:

- A systolic array architecture designed to efficiently support both DNN and Transformer workloads, by leveraging the Split-Posit format.
- A modified hardware micro-architecture that minimizes silicon footprint by reducing decoder complexity across the array network.

The remainder of this paper is as follows. Section 2 provides a basic overview of the Posit format. Section 3 dives into the proposed methodology, including the Symmetric Two-Term Posit approximation and corresponding hardware micro-architecture, covering the processing element design and systolic array organization. Section 4 presents the evaluation framework, covering numerical error characterization, hardware implementation, analysis, and neural network case studies. Finally, section 5 concludes the paper. All the design files are made freely available at [3] for further usage to the designers and research community.

2 Background

2.1 Posit Format

The representation of an N-bit Posit number is illustrated in Figure 1.

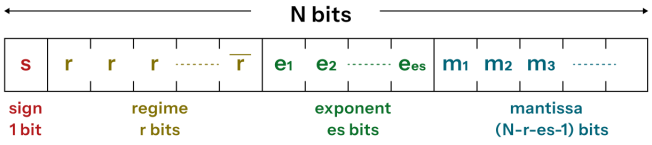


Figure 1: Bit-field composition of an N-bit Posit number, depicting the hierarchical encoding of the sign, regime, exponent, and mantissa fields.

It mainly comprises of four components: sign, regime, exponent, and mantissa. Excluding the most significant sign bit (0 for positive numbers and 1 for negative numbers), the remaining $(N-1)$ bits are hierarchically partitioned to accommodate the regime, exponent, and mantissa fields in decreasing order of precedence. The regime field is encoded as a contiguous sequence of identical bits terminated by an inverted bit, effectively encompassing both magnitude and scale. Specifically, if the bit-stream has r consecutive zeros followed by a one, the regime parameter k is set to $-r$. Conversely, k is set to $r-1$. The next subsequent fixed set of es bits constitutes the exponent field, denoted as e, while the residual bits make up the mantissa m. The mantissa field includes an implicit leading 1 bit, analogous to the normalized IEEE 754 counterpart. The numerical equivalent of a number x in Posit format is deduced from the expression $x = (-1)^s \times useed^k \times 2^e \times (1+m)$, where $useed = 2^{2^{es}}$. The posit format provides a dynamic representable range covering from 2^{-4N+8} to 2^{4N+8}, encompassing two special representations: Zero, denoted by an all-zero bit stream, and NaR (Not a Real), represented by an all-one bit pattern.

3 Proposed Methodology

3.1 Symmetric Two-Term Posit Approximation (STPA) Model

A central challenge in implementing high-precision Posit arithmetic is the superlinear increase in hardware cost with increasing bit width. To mitigate this overhead, we propose the Symmetric Two-Term Posit Approximation (STPA) algorithm, which enables high-precision computations to be emulated using lower-precision units, thereby reducing area and power costs with minimal accuracy degradation. The key idea here is to decompose each high-precision (N, es) operand as the sum of two reduced-precision (N^*, es^*) operands. Consider the baseline operation: $out = A \times B + C$ where, $A, B, C \in$ Posit (N, es). STPA decomposes each operand into two lower-precision components as expressed in the Equation 1, where, $A_i, B_i, C_i \in$ Posit (N^*, es^*).

$$A \approx A_1 + A_2, B \approx B_1 + B_2, C \approx C_1 + C_2, \tag{1}$$

Substituting back, we obtain:

$$out \approx (A_1 + A_2)(B_1 + B_2) + (C_1 + C_2) \tag{2}$$

which expands to:

$$out \approx A_1B_1 + A_1B_2 + A_2B_1 + A_2B_2 + C_1 + C_2 \tag{3}$$

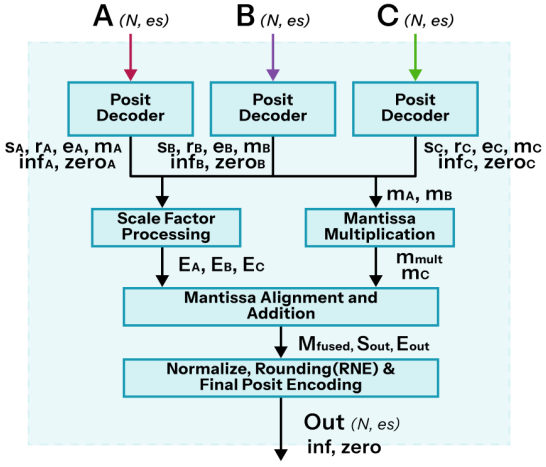


Figure 2: Standard fused multiply and accumulate unit with decoders and encoders inside the MAC unit.

All computations are performed in the reduced-precision (N^*, es^*) domain, leveraging a single low-cost MAC unit reused sequentially, rather than a costly high-precision counterpart. The decomposition procedure is formally detailed in the Algorithm 1.

Briefly, each high-precision operand is first decoded to its real numerical value and mapped to the nearest representable (N^*, es^*) number, yielding the first component ($val1$). The residual, obtained as the difference between the original value and $val1$, is then mapped again to generate the second component $val2$. The original value is thus approximated as $val_1 + val_2$, where each term carries part of the information content of the higher-precision input. Unlike direct quantization to a single reduced-precision Posit, which incurs large relative error, STPA reduces approximation error by converting first-order rounding noise into a second-order residual. This mechanism is analogous to compensated summation in floating-point arithmetic, ensuring quadratic suppression of error. Thus, the combined representation preserves accuracy significantly better than a single reduced-precision approximation, while maintaining low hardware cost. Here, both terms $val1$ and $val2$ belong to the same reduced-precision Posit format. The choice of symmetry is deliberate. An alternative asymmetric decomposition would use components of different Posit formats. For example, combining (N^*, es^*) with a slightly higher-precision (N^{**}, es^{**}). While this might reduce error marginally, it would require two distinct hardware units to support different Posit formats, significantly increasing area and design complexity. Analysis shows that the small gain in accuracy does not justify the large increase in hardware resources. Hence, STPA adopts a symmetric representation, which achieves error suppression while retaining a single, reusable low-precision hardware datapath. Another key observation is the potential increase in memory requirements in some cases, for example, when a 16-bit value is split into two 10-bit components. However, the savings in compute hardware cost outweigh this memory overhead. This trade-off of modest hardware overhead for substantial accuracy retention makes STPA an attractive primitive for accelerator design.

Algorithm 1 Symmetric Two-Term Posit Approximation (STPA) Algorithm

Require: $input$: a Posit-encoded value with configuration (N, es)
Require: (N^*, es^*): target reduced-precision Posit configuration
Ensure: Two reduced-precision values (val_1, val_2) such that $val_1 + val_2 \approx input$

$real_value \leftarrow \textsc{DecodePosit}(input, N, es)$
$val_1 \leftarrow \textsc{RoundToNearest}(real_value, N^*, es^*)$
$residual \leftarrow real_value - val_1$
$val_2 \leftarrow \textsc{RoundToNearest}(residual, N^*, es^*)$

return (val_1, val_2)

3.2 Hardware Micro-architecture

3.2.1 ***Processing Element Design.*** The Processing Element (PE) serves as a fundamental computational unit of the systolic array, performing Split-Posit multiply and accumulate operations with quire-based accumulation. Each PE receives pre-decoded input activations and the weights, which are in split format, eliminating internal splitting and the decoding logic. The split representation divides each $(16, 2)$ Posit operand into two smaller Posit components, simplifying the design. Experimental analysis shows that the $(10, 2)$ appears to be the most efficient in terms of both statistical and hardware metrics. This is explained in detail later on in Section 4. The PE computational core is a Fused Multiply-Accumulate unit (FMA), illustrated in Figure 2. The FMA unit consists of three tightly integrated components that work together to support posit-based fused operations. First, the 10-bit posit multiplication stage performs the standard posit multiply operation, including regime adjustment, exponent addition, and mantissa multiplication. Second, the product is fed into the internal quire accumulator, which

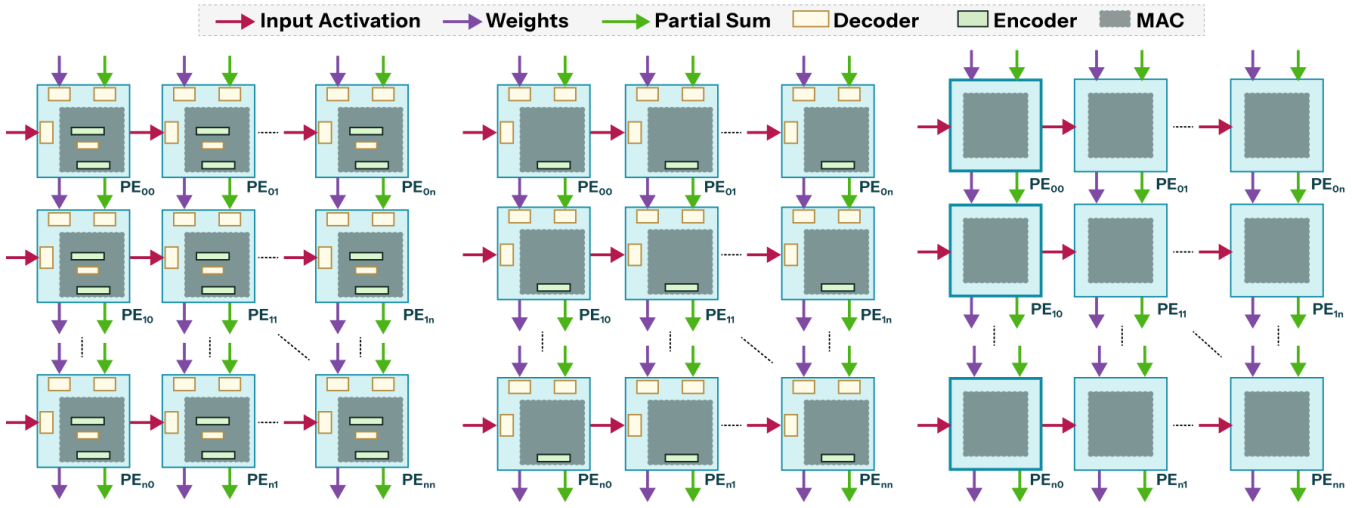


Figure 3: a) Standard systolic array with Posit-enabled Processing Elements (PEs), each incorporating four decoders and two encoders per PE, resulting in a total of $4N^2$ decoders and $2N^2$ encoders across the $N \times N$ array. b) Enhanced systolic array architecture with Fused Multiply-Accumulate support, reducing to a total of $3N^2$ decoders and N^2 encoders. c) Further architectural refinement consisting of $2N(N-1)$ decoders and N encoders across the array.

is a wide register specifically designed to absorb products without any intermediate rounding. The quire supports exact accumulation of partial products by expanding the dynamic range far beyond the considered Posit format, thereby preventing intermediate overflows and eliminating the propagation of rounding errors across multiple MAC cycles. The four partial products corresponding to all the combinations of high and low components are computed sequentially by reusing the same FMA unit. During this process, the FMA continuously chains operations such that each product is immediately accumulated before the next multiplication begins. The resulting output, expressed in the Equation 4 represents the complete split posit MAC operation.

$$acc_{out} = acc_{in} + A_1B_1 + A_1B_2 + A_2B_1 + A_2B_2 \tag{4}$$

3.2.2 ***Systolic Array Organization.*** The proposed Processing Element (PE) is integrated into a systolic array architecture optimized for high performance and efficient data movement. A weight-stationary dataflow is adopted to maximize memory reuse and minimize communication overhead. The array operates with activations flowing horizontally (from left to right, row-wise) and accumulators flowing vertically (from top to bottom, column-wise). The design evolution of the proposed systolic array is illustrated in Figure 3. Initially, the traditional systolic array is extended by replacing conventional fixed/floating-point MAC PEs with Posit-enabled PEs, resulting in a significant area increase due to extensive conversion logic (requiring $4N^2$ decoders and $2N^2$ encoders for a $N \times N$ array). Incorporating Fused Multiply-Accumulate (FMA) units mitigates this overhead, reducing the number of decoders and encoders to $3N^2$ and N^2, respectively. Further optimization is done by rearranging the conversion logic to the array periphery and employing a chained accumulation scheme within the array. This refinement reduces the number of decoders and encoders to 62% for smaller array sizes and approximately 50% as N increases, compared to the traditional architecture. The architecture is illustrated in detail in the Figure 4.

The proposed systolic array operates as a deeply regular and time-synchronized computational fabric in which data, weights, and partial sums flow through a structured pipeline. The decoder elements are arranged at the array periphery, corresponding to each PE. Once decoded, only the component signals are distributed internally to the rest of the array configuration. The split-posit operands are decoded into component signals: sign, packed regime-exponent pair (referred to as scale factor), and mantissa, along with two flags representing zero and infinity. These decoded signals are broadcast horizontally and vertically from dedicated buffers positioned along the edges, forming the foundation for low-power, high-throughput computation. At the left boundary, row buffers receive decoded activation values from global memory and distribute them horizontally across all PEs in the corresponding row. In parallel, column buffers at the top boundary supply decoded weights vertically along each column. Each PE multiplies the incoming activation and weight components, accumulating the resulting partial sums with those received from the PE directly above. This vertical propagation forms a column-wise accumulation chain characteristic of systolic computation, enabling continuous operation with minimal idle cycles. By operating in the quire domain, the design bypasses the normalization and alignment stages typically required for floating-point arithmetic, thereby reducing latency and datapath complexity while ensuring precise accumulation over large matrix operations. Upon completion of the accumulation process, each PE column output is passed to an encoder positioned at the bottom of the array. The encoder performs leading-one detection, regime and exponent reconstruction, mantissa extraction, and rounding to generate the final posit output. This hierarchical approach significantly reduces

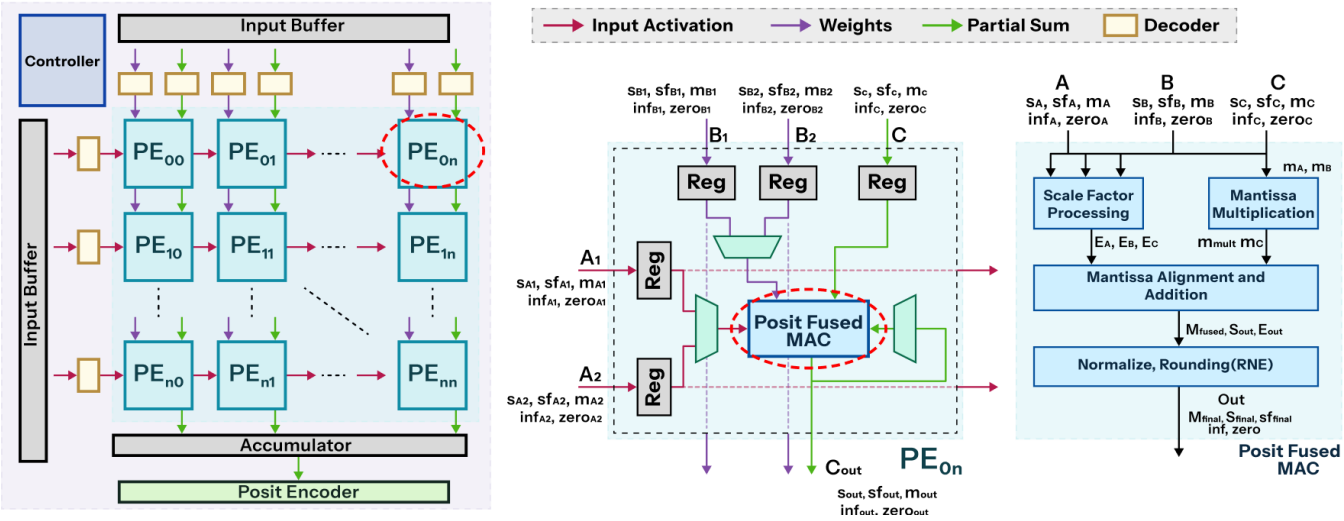


Figure 4: a) Modified systolic array architecture incorporating PEs with peripheral conversion logic and a modified controller to facilitate sequential partial product accumulation. b) Detailed view of the Split-Posit PE illustrating the internal dataflow. c) Split-Posit FMA showing the computation of high-low component combinations and chained quire-based accumulation.

internal data traffic, minimizes redundant logic, and improves both energy and area efficiency.

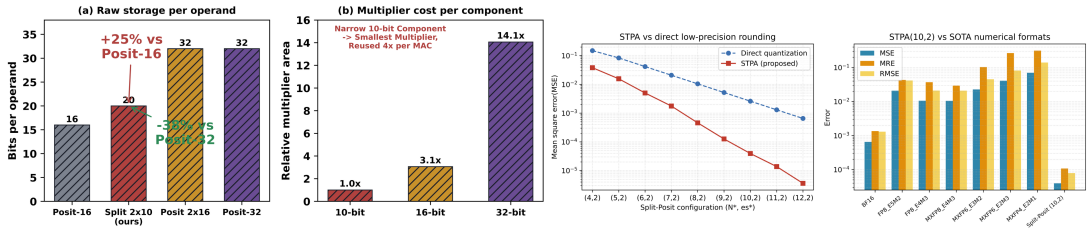
To sustain high throughput, the memory and control subsystems are carefully co-designed. On-chip storage is divided into activation, weight, and output buffers, optimized for a weight-stationary dataflow. Activations are streamed from global memory to row buffers, while weights are pre-loaded into column buffers, and partial sums are accumulated locally within quire registers before being written back to the output buffers. A double-buffered interface enables concurrent data loading and computation, effectively hiding memory latency and sustaining continuous systolic operation. Control across the array is coordinated by a lightweight finite state machine (FSM) that orchestrates data movement, computation sequencing, accumulation, and write-back. Simple handshaking signals between rows and columns maintain precise timing alignments, ensuring deterministic data propagation and scalability to larger array dimensions. Once the systolic pipeline reaches a steady state, a new output is produced every clock cycle, as each PE performs multiply–accumulate operations on different data tiles, achieving near-ideal utilization of processing resources. By combining split-posit peripheral decoding, quire-based accumulation, and distributed control, the architecture achieves a balanced trade-off between numerical precision, area efficiency, and energy consumption. As a result, the systolic pipeline achieves balanced latency, predictable timing behavior, and scalable throughput, making it suitable for both convolution and fully connected layers in deep learning inference workloads.

4 Results and Discussion

4.1 Numerical Error Characterization

To validate the robustness of the proposed methodology, an extensive characterization of numerical error was performed. The analysis includes both statistical evaluations of numerical deviations from the higher-precision counterparts and hardware-level validation to obtain corresponding metrics for area, power, and delay.

All numerical error metrics were computed using 1,000,000 values uniformly sampled from the range $[-1, 1]$, which provides a representative approximation of operand distributions encountered in neural-network workloads. The evaluation initially compares direct Posit16 quantisation with the proposed STPA method and is subsequently extended to reduced-precision formats widely used in current accelerators, including BF16, FP8, and MXFP variants. To avoid drawing conclusions from a single error measure, we evaluate MSE, MRE, and RMSE, as presented in Figure 5(b). Since all three metrics exhibit consistent trends, MSE is adopted as the primary metric throughout the remainder of the paper. Among the evaluated baselines, BF16 produces the lowest numerical error and therefore represents the strongest comparator. Consequently, the accuracy evaluation uses BF16 as the reference baseline. Figure 6 shows the trend of the quantization effect for higher precision components by the STPA model. The numerical deviation is primarily evaluated using the Mean Squared Error (MSE) metric, while additional evaluations with Mean Relative Error (MRE) and Root Mean Square Error (RMSE) exhibit similar trends. After validating the decomposition of the $(16, 2)$ configuration into its split counterparts, the same methodology is extended to other configurations, such as $(12, 2)$ and $(10, 2)$, to demonstrate its generality. The results, obtained across varying systolic array dimensions, show that the numerical error consistently remains on the order of 10^{-4} when employing 22-bit quire registers. While prior studies recommend quire sizes of 128 bits or higher, our analysis shows that a 22-bit quire provides sufficient dynamic range and accumulation precision for neural network workloads, making extended quire lengths unnecessary for maintaining numerical accuracy. Figure 7 shows the trend of



(a) (b)

Figure 5: Numerical error and hardware cost characterization of the proposed Split-Posit approach. a) Comparison of storage overhead and multiplier area with increasing operand width, highlighting the modest memory overhead of Split-Posit and the substantial area savings enabled by narrow 10-bit multipliers. b) Comparison of MSE, MRE, and RMSE across Posit16, STPA, BF16, FP8, and MXFP formats, showing consistent error trends and motivating MSE as the primary evaluation metric.

area against increasing quire-bit register sizes for different Posit configurations. The results clearly indicate that employing smaller quire sizes significantly reduces area overhead, thereby enabling resource-efficient and scalable hardware implementations.

The Split-Posit representation introduces a modest, one-time increase in memory requirements in exchange for substantial savings in replicated compute resources. As shown in Figure 5(a), representing a 16-bit operand using two 10-bit components requires 20 bits of storage, corresponding to a 25% increase over Posit-16. However, this requirement remains 38% lower than the 32-bit SRAM storage that would otherwise be needed to provide a comparable dynamic range. Importantly, this additional storage is restricted to the operand buffers and is incurred only once, whereas the computational hardware is replicated across the processing elements (PEs).

The benefits become more pronounced when considering multiplier cost. Multiplier area increases super-linearly with operand width: relative to a 10-bit multiplier with normalized area of 1.0×, the corresponding areas for 16-bit and 32-bit multipliers increase to 3.1× and 14.1×, respectively. Since the multiplier constitutes a major contributor to both PE area and power and is replicated across all 256 PEs in the 16×16 array, reducing the multiplier width to 10 bits provides approximately a 3× reduction in multiplier cost per PE. This observation also motivates the sequential execution of the Split-Posit partial products. Rather than replicating multiple wide multipliers, the four required partial products are generated by reusing a single narrow multiplier across successive operations. Although this approach introduces additional execution passes, the reduction in multiplier complexity more than compensates for the associated overhead, resulting in approximately 50% lower area and 77% lower power per PE. Thus, the relatively small 25% increase in buffer storage is substantially outweighed by the savings achieved in the replicated compute hardware.

4.2 Hardware Implementation and Analysis

4.2.1 ***Experimental Setup.*** The proposed architecture is modeled with Verilog HDL and synthesized using Cadence Genus in 45nm technology node. The entire RTL-GDSII flow is implemented on the optimal synthesized netlist using Cadence Innovus.

Since prior studies have already established 16-bit Posits as a viable replacement for FP32 units, we adopt 16-bit Posit units as our baseline and decompose them into smaller configurations. Analysis is performed across multiple configurations ranging from $(4, 1)$ to $(12, 4)$ with corresponding metrics summarized in Table 1. To maintain generality, results are presented per processing element (PE), though the architecture can be readily scaled to larger systolic array dimensions. The same methodology can be extended to 12-bit and 10-bit Posit decompositions, which exhibit similar performance and efficiency trends. Experimental results show that $(16, 2)$ split into $(10, 2)$ configuration achieves an optimal trade-off between hardware efficiency and numerical accuracy. Specifically, it concedes a silicon area of 9799.371$\mu m2$, a power consumption of 3.83mW , and a critical path delay of 1.970ns which is smaller when compared to the baseline $(16, 2)$ configuration with silicon area of 16328.226$\mu m2$, a power consumption of 7.19mW , and a critical path delay of 1.96ns. This represents a 39.98% reduction in silicon footprint and 46.74% savings in power. Although the Split-MAC configuration has a comparable critical path delay, but requires 1.2× more clock cycles to complete the standard $(16, 2)$ MAC operation (as defined in Section 3.1), the overall trade-off is justified by the substantial savings in footprint and power cost. Furthermore, when integrated within the systolic array, the chained-accumulator structure partially mitigates this latency overhead, leading to a more efficient and balanced hardware design.

A key observation from Table 1 is the close proximity of the measured delay values across the different configurations. Although the value of N varies across configurations, the overall delay is primarily determined by the critical-path delay of the dominant MAC operation on the mantissa. In Posit arithmetic, the mantissa computation is performed using unsigned integer operations, with the accumulation carried out in the quire. As discussed in Section 4.1, a fixed quire width of 22 bits is selected as the optimal configuration. Consequently, the quire width remains unchanged across the different Split-Posit configurations, making the critical-path delay largely independent of N. Therefore, the observed delay remains comparable across configurations, despite variations in the operand decomposition.

4.2.2 ***Hardware Implementation.*** To ensure a fair and comprehensive comparison, the proposed architecture is benchmarked against existing State-of-the-Art (SOTA) designs spanning a broad range of numerical formats, including floating-point, fixed-point,

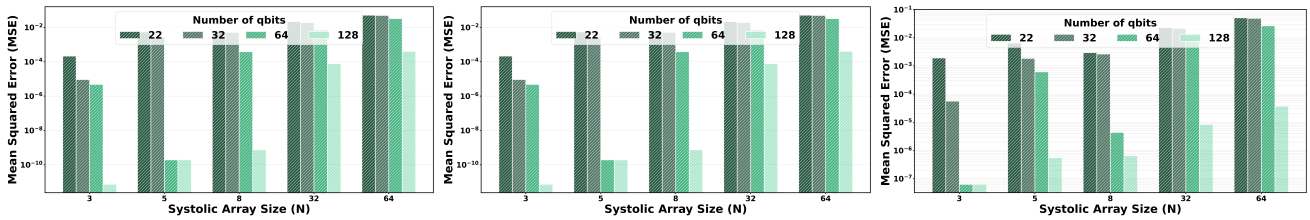


Figure 6: Mean Squared Error (MSE) analysis across various quire-bit (qbit) widths for a) 16-bit Posit, b) 12-bit Posit, and c) 10-bit Posit configurations, evaluated across systolic array sizes ranging from 3×3 to 64×64.

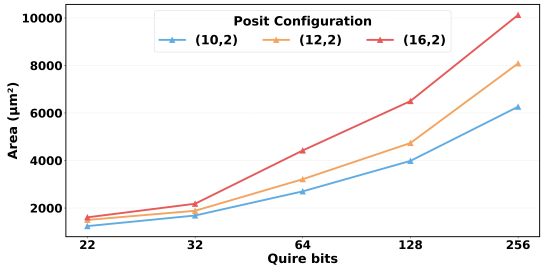


Figure 7: Analysis of silicon area footprint across varying Quire bits for $(16, 2)$, $(12, 2)$, $(10, 2)$ Posit configurations, evaluated for a fixed $N \times N$ systolic fabric.

integer, and Posit-based accelerators, as summarized in Table 2. Although integer-based implementations such as [2] report lower silicon area, this reduction is primarily attributed to their simplified arithmetic units. Their datapaths and control logic are considerably less complex, enabling compact physical implementations. However, this hardware efficiency comes at the cost of numerical accuracy, with integer-based designs generally experiencing significant accuracy degradation, particularly for deeper neural networks and attention-based architectures that require greater dynamic range and numerical precision.

Similarly, [34] explores logarithmic Posit representations at 2-, 4-, and 8-bit precisions, achieving substantial area savings at the expense of numerical fidelity.For example,the $(8, 2)$ Posit format provides a maximum mantissa of only 3 bits, which limits precision and can lead to considerable accuracy degradation for deeper and attention-driven networks. At the extreme low-precision end, the 2-bit Posit representation provides virtually no dedicated bits for the mantissa or exponent, relying primarily on the sign and regime fields. Consequently, its ability to represent values with sufficient precision and dynamic range is severely constrained, further limiting its applicability to numerically demanding workloads. Furthermore, [23] achieves 1.88× higher throughput than the proposed baseline; however, this performance improvement comes at the cost of a 97.63% increase in hardware area. Such a substantial increase in hardware complexity can be undesirable in resource-constrained environments, where area and energy efficiency are critical design considerations. Therefore, the higher throughput does not necessarily translate into a more favorable overall design point. In contrast, the proposed architecture provides a more balanced trade-off among throughput, numerical accuracy,

Table 1: Hardware metrics evaluated per processing element across various Split-Posit configurations.

Configuration	Area(μm^2)	Power(mW)	Delay(ns)
(4,1)	5509.789	2.25	1.968
(4,2)	5372.361	2.20	1.980
(5,1)	6673.167	2.80	1.968
(5,2)	6366.636	2.57	1.968
(6,1)	6860.356	2.87	1.980
(6,2)	7012.485	2.76	1.970
(7,1)	7444.946	3.12	1.969
(7,2)	7529.794	3.01	1.969
(8,1)	8125.937	3.55	1.968
(8,2)	8087.55	3.55	1.968
(8,3)	7627.173	2.97	1.970
(8,4)	7767.462	2.97	1.970
(9,1)	9756.238	4.11	1.979
(9,2)	9375.748	3.84	1.980
(9,3)	8927.647	3.45	1.969
(9,4)	8746.491	3.22	1.968
(10,1)	10406.555	4.47	1.970
(10,2)	9799.371	3.83	1.970
(10,3)	10432.097	4.40	1.968
(10,4)	9520.199	3.79	1.969
(11,1)	12285.958	5.48	1.970
(11,2)	10614.574	4.47	1.967
(11,3)	10583.085	4.31	1.970
(11,4)	10083.795	3.94	1.968
(12,1)	13511.129	6.14	1.970
(12,2)	12580.72	5.41	1.969
(12,3)	11025.009	4.44	1.980
(12,4)	11785.371	4.39	1.971

Note: The measured delay values are primarily determined by the fixed 22-bit quire width, which remains unchanged across all Split-Posit configurations.

and hardware efficiency. Rather than optimizing a single metric at the expense of others, the proposed design maintains competitive computational performance while avoiding the substantial area overhead associated with higher-throughput architectures and the accuracy limitations inherent to highly quantized designs. This balance makes the proposed architecture a more practical and scalable solution for resource-constrained deep-learning accelerators.

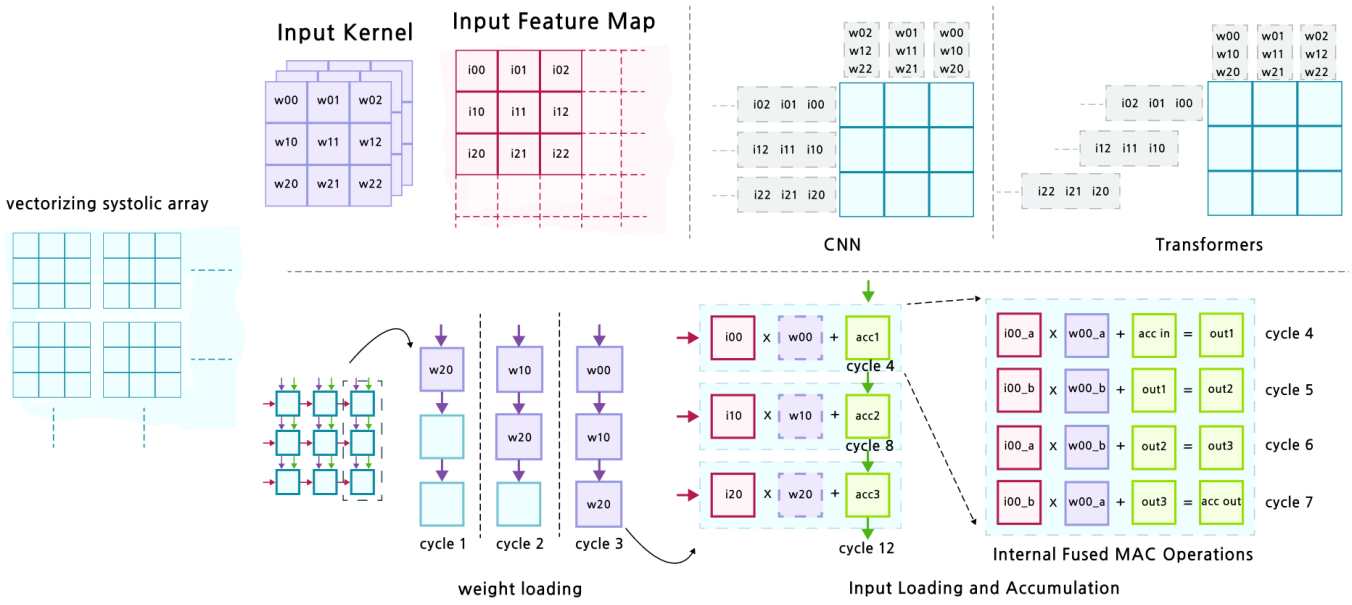


Figure 8: Cycle-level execution flow illustrating the operation of a 3×3 kernel mapped onto a tiled systolic fabric, including the per-cycle data movement of the kernel weights and input feature map for both CNNs and Transformer workloads.

Table 2: Comparison of hardware utilization and throughput with existing and relevant State-Of-The-Art (SOTA) works.

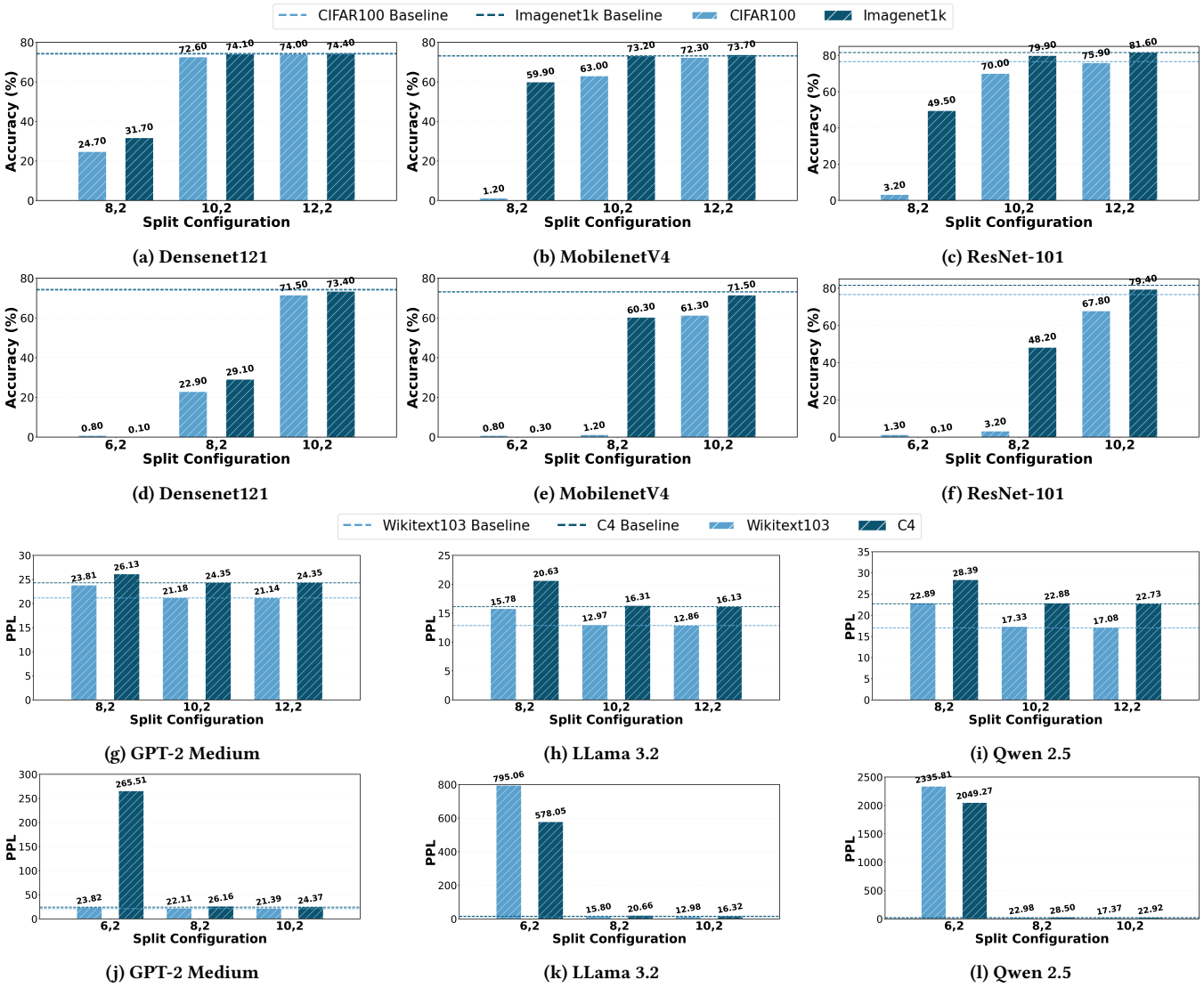
	Our Work	**[23]**	**[2]**	**[8]**	**[34]**	**[38]**
Technology	45nm	65nm	65nm	40nm	28nm	12nm
Format	Split Posit	Fixed 8-bit	int(1, 16)	Fixed 16-bit	Log-Posit 2/4/8	FP-4/8
Operating Conditions (Voltage, Frequency)	1.20V, 500MHz	1.20V, 200MHz	0.9V, 100MHz	0.99V, 500MHz	NA †	1.00V, 717MHz
Array Size	16x16	16x16	16x16	3x3 (block size 3)	8x8	NA †
Throughput ($GOPS$)	418.75	786.84	110.70	156.57	167.96	477.55
Area (mm^2)	1.69	3.34	0.93	2.08	0.034	418.18
Power (mW)	77.88	152.88	31.98	165.60	NA †	412.64
Area Efficiency ($GOPS/mm^2$)	247.78	236.05	118.86	75.27	4866.91	1.14
Power Efficiency ($TOPS/W$)	5.37	5.14	3.46	0.94	–	1.16

Note: All the Hardware Parameters (Area and Power) and Throughput values are normalized to *45nm* using DeepScaleTool [36].
† The paper does not report values.

4.3 Neural Network Case Studies

Standard neural networks were evaluated on the proposed method to demonstrate the practical viability. The initial experiments involve integrating the proposed Split-Posit systolic array for running conventional Convolutional Neural Network (CNN) architectures by replacing their convolution layers with the designed hardware. The study was further extended to Transformer models, where the systolic array was utilized to accelerate the matrix multiplication units responsible for attention and feed-forward operations. All experiments were standardized using a 16×16 array configuration. For convolution layers requiring smaller kernel sizes, tiling was employed to ensure consistent hardware utilization under the same area constraints. To maintain high utilization across varying convolution kernel sizes, especially in early layers of CNNs, a tiling strategy was adopted. Convolutions with spatial dimensions smaller than the 16×16 compute fabric were decomposed into sub-tiles, enabling consistent PE engagement and eliminating idle cycles.

Figure 8 illustrates the complete execution timeline of the systolic array processing a single 3×3 convolution tile, with pipelined operations that allow data to traverse the architecture. The data arrangement for the convolution operation and general matrix multiplication is shown in Figure 8. The compute pipeline is divided into three phases: weight loading, activation streaming, and PE compute propagation, each optimized for fully pipelined data movement. The computation begins with a 3-cycle weight loading stage. Kernel weights are fetched from corresponding SRAM banks and routed through the peripheral column decoders responsible for generating the Split-Posit representation. The resulting decoded weight components are broadcast vertically into the 16×16 PE mesh. Each column of PEs receives the decoded weight segment corresponding to its kernel, which is then latched into internal PE



(a) Densenet121 (b) MobilenetV2 (c) ResNet-101

(d) Densenet121 (e) MobilenetV2 (f) ResNet-101

(g) GPT-2 Medium (h) LLama 3.2 (i) Qwen 2.5

(j) GPT-2 Medium (k) LLama 3.2 (l) Qwen 2.5

Figure 9: Performance trends of CNN and Transformer benchmarks evaluated for $(16, 2)$ and $(12, 2)$ Split Posit configurations

registers. This mechanism ensures one-time weight loading, eliminating redundant memory fetches during subsequent activation sweeps. After weight setup, input activations are streamed horizontally through the peripheral row decoders. These decoders apply the same Split-Posit decomposition used for weights, but optimized for high-throughput reuse. The decoded activation fragments traverse the horizontal interconnects, where each PE pairs the incoming activation with its pre-loaded weight. To reduce off-chip bandwidth, activation rows are reused across multiple PE columns, ensuring that each activation contributes to multiple MAC operations before being discarded. Double buffering is employed at both input and output boundaries, enabling overlap of SRAM accesses with array computation. This decouples compute from memory latency and sustains a continuous data pipeline. Each PE performs a 4-cycle MAC sequence for every spatial tile position, consisting of Split-Posit multiplication, normalization, and accumulation. During the final cycle, partial sums are forwarded downward to the next PE in the column, thus realizing the classic wavefront propagation characteristic of systolic architectures. This hierarchical topology ensures deterministic data movement and minimal interconnect overhead. When scaled to the full 16×16 deployment, the architecture achieves a peak energy efficiency of 5.37 TOPS/W, driven by reduced-width Split-Posit arithmetic and optimal operand reuse. The same architecture is extended to Transformer models where evaluation is done on the matrix multiplication component. For Transformer inference workloads, input matrices (Q, K, V, and feed-forward layers) were sent in a skewed fashion to match the systolic propagation pattern. This ensures that partial products move diagonally across

the array with no structural stalls. The skewing eliminates idle bubbles and maximizes PE occupancy, which is particularly important in attention score computation where dimensionality is large and dense.

The proposed architecture was validated across a range of neural network models and datasets. Specifically, evaluations were conducted using DenseNet121, MobilenetV4, and ResNet101 on the CIFAR-100 and ImageNet-1 K datasets. Additionally, LLM-based model evaluations were conducted using GPT-2, Llama 3.2, and Qwen 2.5 on Wikitext103 and Crawler C4 datasets. All models were trained with BF16 precision and subsequently evaluated under various split-posit configurations, and the corresponding accuracy/perplexity PPL scores were recorded. Since the observed trends were consistent across all configurations, Figure 9 depicts the results for $(16, 2)$ and $(12, 2)$ decompositions. Across all models and datasets, the split-posit methodology exhibits only negligible degradation relative to the baseline while operating at substantially lower hardware cost. Another key observation is that the results consistently indicate that the $(10, 2)$ offers the most favorable balance between accuracy and hardware efficiency. These results effectively capture the trade-off between numerical precision and inference accuracy, highlighting the robustness of the proposed Split-Posit framework across diverse model architectures and datasets.

5 Conclusion

This work proposes the Split-Posit architectures as a viable and scalable alternative to conventional higher-precision Posit implementations. By integrating Split-Posit arithmetic within a systolic array fabric, the proposed design achieves an effective balance between the hardware efficiency of low-bit-width operations and the numerical accuracy benefits of higher precision. By decomposing higher-bit Posit computations into lower-bit operations, the proposed architecture achieves negligible accuracy loss across standard neural network benchmarks. Further architectural refinements in the systolic organization reduce decoder complexity and overhead, thereby enhancing memory efficiency and throughput. Tiled across a 16×16 array configuration, the architecture achieves a peak energy efficiency of 5.37 TOPS/W, making it suitable for both edge and high-performance inference platforms. The ASIC implementation results further validate the practicality of this approach, achieving a 39.98% savings in silicon area and a 46.74% improvement in energy efficiency compared to state-of-the-art designs. These results highlight the Split-Posit architecture's potential as a foundational building block for next-generation AI accelerators, capable of scaling efficiently across diverse workloads while maintaining precision, reducing memory traffic, and minimizing hardware cost.

References

[1] 2023. TPU v4: An Optically Reconfigurable Supercomputer for Machine Learning with Hardware Support for Embeddings, author=Norman P. Jouppi and George Kurian and Sheng Li and Peter Ma and Rahul Nagarajan and Lifeng Nai and Nishant Patil and Suvinay Subramanian and Andy Swing and Brian Towles and Cliff Young and Xiang Zhou and Zongwei Zhou and David Patterson. arXiv:2304.01433 [cs.AR] https://arxiv.org/abs/2304.01433

[2] Renzo Andri, Lukas Cavigelli, Davide Rossi, and Luca Benini. 2016. YodaNN: An Ultra-Low Power Convolutional Neural Network Accelerator Based on Binary Weights. In *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. 236–241. doi:10.1109/ISVLSI.2016.111

[3] Anonymous. 2025. Split-Posit-Systolic-Array. https://sites.google.com/view/split-posit-systolic-array?usp=sharing.

[4] Giorgos Armeniakos, Alexis Maras, Sotirios Xydis, and Dimitrios Soudris. 2025. MaRVIn: A Cross-Layer Mixed-Precision RISC-V Framework for DNN Inference, from ISA Extension to Hardware Acceleration. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2025), 1–1. doi:10.1109/TCAD.2025.3615505

[5] M Arun and Madhav Rao. 2025. Efficient POSIT Multiplier with Multi-flag Priority Encoding and Multistage Booth Processing. In *2025 28th Euromicro Conference on Digital System Design (DSD)*. 285–291. doi:10.1109/DSD67783.2025.00048

[6] Mahati Basavaraju, Vinay Rayapati, and Madhav Rao. 2024. POCO: Hardware Characterization of Activation Functions using POSIT-CORDIC Architecture. In *2024 IEEE International Symposium on Circuits and Systems (ISCAS)*. 1–5. doi:10.1109/ISCAS58744.2024.10558487

[7] Zachariah Carmichael, Hamed F. Langroudi, Char Khazanov, Jeffrey Lillie, John L. Gustafson, and Dhireesha Kudithipudi. 2019. Deep Positron: A Deep Neural Network Using the Posit Number System. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 1421–1426. doi:10.23919/DATE.2019.8715262

[8] Kuo-Wei Chang and Tian-Sheuan Chang. 2020. Efficient Accelerator for Dilated and Transposed Convolution with Decomposition. In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. 1–5. doi:10.1109/ISCAS45731.2020.9180402

[9] Qinyu Chen, Yan Huang, Rui Sun, Wenqing Song, Zhonghai Lu, Yuxiang Fu, and Li Li. 2020. An Efficient Accelerator for Multiple Convolutions From the Sparsity Perspective. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 28, 6 (2020), 1540–1544. doi:10.1109/TVLSI.2020.2976454

[10] Brian Crafton, Xiaochen Peng, Xiaoyu Sun, Ashwin Lele, Bo Zhang, Win-San Khwa, and Kerem Akarvardar. 2025. Finding the Pareto Frontier of Low-Precision Data Formats and MAC Architecture for LLM Inference. In *2025 62nd ACM/IEEE Design Automation Conference (DAC)*. 1–7. doi:10.1109/DAC63849.2025.11132989

[11] Florent de Dinechin, Luc Forget, Jean-Michel Muller, and Yohann Uguen. 2019. Posits: the good, the bad and the ugly. In *Proceedings of the Conference for Next Generation Arithmetic 2019* (Singapore, Singapore) *(CoNGA'19)*. Association for Computing Machinery, New York, NY, USA, Article 6, 10 pages. doi:10.1145/3316279.3316285

[12] Zihao Deng, Sayeh Sharify, Xin Wang, and Michael Orshansky. 2025. Mixed-Precision Quantization for Deep Vision Models with Integer Quadratic Programming. In *2025 62nd ACM/IEEE Design Automation Conference (DAC)*. 1–7. doi:10.1109/DAC63849.2025.11132777

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *CoRR* abs/2010.11929 (2020). arXiv:2010.11929 https://arxiv.org/abs/2010.11929

[14] Razvan-Gabriel Dumitru, Vikas Yadav, Rishabh Maheshwary, Paul-Ioan Clotan, Sathwik Tejaswi Madhusudhan, and Mihai Surdeanu. 2024. Layer-Wise Quantization: A Pragmatic and Effective Method for Quantizing LLMs Beyond Integer Bit-Levels. arXiv:2406.17415 [cs.CL] https://arxiv.org/abs/2406.17415

[15] Ruihao Gong, Yifu Ding, Zining Wang, Chengtao Lv, Xingyu Zheng, Jinyang Du, Haotong Qin, Jinyang Guo, Michele Magno, and Xianglong Liu. 2024. A Survey of Low-bit Large Language Models: Basics, Systems, and Algorithms. arXiv:2409.16694 [cs.AI] https://arxiv.org/abs/2409.16694

[16] Gustafson and Yonemoto. 2017. Beating Floating Point at its Own Game: Posit Arithmetic. *Supercomput. Front. Innov.: Int. J.* 4, 2 (June 2017), 71–86. doi:10.14529/jsfi170206

[17] Song Han, Huizi Mao, and William J. Dally. 2016. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. arXiv:1510.00149 [cs.CV] https://arxiv.org/abs/1510.00149

[18] Soheil Hashemi, Nicholas Anthony, Hokchhay Tann, R. Iris Bahar, and Sherief Reda. 2017. Understanding the impact of precision quantization on the accuracy and energy of neural networks. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. 1474–1479. doi:10.23919/DATE.2017.7927224

[19] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. 2018. *AMC: AutoML for Model Compression and Acceleration on Mobile Devices*. Springer International Publishing, 815–832. doi:10.1007/978-3-030-01234-2_48

[20] Shen-Fu Hsiao, Sin-Chen Lin, Guan-Lin Chen, Shih-Hua Yang, Yen-Che Yuan, and Kun-Chih Chen. 2024. Design of an Efficient Deep Neural Network Accelerator Based on Block Posit Number Representation. In *2024 International VLSI Symposium on Technology, Systems and Applications (VLSI TSA)*. 1–4. doi:10.1109/VLSITSA60681.2024.10546390

[21] Xing Hu, Yuan Cheng, Dawei Yang, Zhihang Yuan, Jiangyong Yu, Chen Xu, and Sifan Zhou. 2024. I-LLM: Efficient Integer-Only Inference for Fully-Quantized Low-Bit Large Language Models. arXiv:2405.17849 [cs.LG] https://arxiv.org/abs/2405.17849

[22] Yaqi Hu, Zhuqing Yuan, Weichen Gao, Sheng Zhang, and Yongpan Liu. 2025. An Integer-only Quantization Framework for Edge Deployment of Large Language Models. In *2025 IEEE International Symposium on Circuits and Systems (ISCAS)*.

1–5. doi:10.1109/ISCAS56072.2025.11043993

[23] Dongseok Im, Donghyeon Han, Sungpill Choi, Sanghoon Kang, and Hoi-Jun Yoo. 2019. DT-CNN: Dilated and Transposed Convolution Neural Network Accelerator for Real-Time Image Segmentation on Mobile Devices. In *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*. 1–5. doi:10.1109/ISCAS.2019.8702243

[24] Manish Kumar Jaiswal and Hayden K.-H. So. 2019. PACoGen: A Hardware Posit Arithmetic Core Generator. *IEEE Access* 7 (2019), 74586–74601. doi:10.1109/ACCESS.2019.2920936

[25] Aaron Jarmusch, Nathan Graddon, and Sunita Chandrasekaran. 2025. Dissecting the NVIDIA Blackwell Architecture with Microbenchmarks. arXiv:2507.10789 [cs.DC] https://arxiv.org/abs/2507.10789

[26] Jiedong Lang, Zhehao Guo, and Shuyu Huang. 2024. A Comprehensive Study on Quantization Techniques for Large Language Models. arXiv:2411.02530 [cs.LG] https://arxiv.org/abs/2411.02530

[27] Hamed F. Langroudi, Zachariah Carmichael, John L. Gustafson, and Dhireesha Kudithipudi. 2019. PositNN Framework: Tapered Precision Deep Learning Inference for the Edge. In *2019 IEEE Space Computing Conference (SCC)*. 53–59. doi:10.1109/SpaceComp.2019.00011

[28] Janghwan Lee, Youngdeok Hwang, and Jungwook Choi. 2023. Finding Optimal Numerical Format for Sub-8-Bit Post-Training Quantization of Vision Transformers. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 1–5. doi:10.1109/ICASSP49357.2023.10096798

[29] Yanbiao Liang, Huihong Shi, and Zhongfeng Wang. 2025. M2-ViT: Accelerating Hybrid Vision Transformers With Two-Level Mixed Quantization. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 33, 5 (2025), 1492–1496. doi:10.1109/TVLSI.2024.3525184

[30] Fangxin Liu, Ning Yang, Zongwu Wang, Xuanpeng Zhu, Haidong Yao, Xiankui Xiong, Li Jiang, and Haibing Guan. 2025. BLOOM: Bit-Slice Framework for DNN Acceleration with Mixed-Precision. In *2025 62nd ACM/IEEE Design Automation Conference (DAC)*. 1–7. doi:10.1109/DAC63849.2025.11133246

[31] Fasi M, Higham NJ, Mikaitis M, and Pranesh S. 2021. Numerical behavior of NVIDIA tensor cores. https://doi.org/10.7717/peerj-cs.330

[32] Raul Murillo, Alberto A. Del Barrio, and Guillermo Botella. 2020. Customized Posit Adders and Multipliers using the FloPoCo Core Generator. In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. 1–5. doi:10.1109/ISCAS45731.2020.9180771

[33] Thomas Norrie, Nishant Patil, Doe Hyun Yoon, George Kurian, Sheng Li, James Laudon, Cliff Young, Norman P. Jouppi, and David Patterson. 2020. Google's Training Chips Revealed: TPUv2 and TPUv3. In *2020 IEEE Hot Chips 32 Symposium (HCS)*. 1–70. doi:10.1109/HCS49909.2020.9220735

[34] Akshat Ramachandran, Zishen Wan, Geonhwa Jeong, John Gustafson, and Tushar Krishna. 2024. Algorithm-Hardware Co-Design of Distribution-Aware Logarithmic-Posit Encodings for Efficient DNN Inference. In *Proceedings of the 61st ACM/IEEE Design Automation Conference* (San Francisco, CA, USA) *(DAC '24)*. Association for Computing Machinery, New York, NY, USA, Article 326, 6 pages. doi:10.1145/3649329.3656544

[35] Vasilis Sakellariou, Vassilis Paliouras, Ioannis Kouretas, Hani Saleh, and Thanos Stouraitis. 2025. A Mixed-Precision RNS DNN Accelerator. In *2025 IEEE International Symposium on Circuits and Systems (ISCAS)*. 1–5. doi:10.1109/ISCAS56072.2025.11043322

[36] Satyabrata Sarangi and Bevan Baas. 2021. DeepScaleTool: A Tool for the Accurate Estimation of Technology Scaling in the Deep-Submicron Era. In *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. 1–5. doi:10.1109/ISCAS51556.2021.9401196

[37] Yubo Shi, Meiqi Wang, Siyi Chen, Jinghe Wei, and Zhongfeng Wang. 2021. Transform-Based Feature Map Compression for CNN Inference. arXiv:2106.12850 [eess.IV] https://arxiv.org/abs/2106.12850

[38] Thierry Tambe, Jeff Zhang, Coleman Hooper, Tianyu Jia, Paul N. Whatmough, Joseph Zuckerman, Maico Cassel Dos Santos, Erik Jens Loscalzo, Davide Giri, Kenneth Shepard, Luca Carloni, Alexander Rush, David Brooks, and Gu-Yeon Wei. 2023. 22.9 A 12nm 18.1TFLOPs/W Sparse Transformer Processor with Entropy-Based Early Exit, Mixed-Precision Predication and Fine-Grained Power Management. In *2023 IEEE International Solid-State Circuits Conference (ISSCC)*. 342–344. doi:10.1109/ISSCC42615.2023.10067817

[39] Yu Wang, Gu-Yeon Wei, and David Brooks. 2019. Benchmarking TPU, GPU, and CPU Platforms for Deep Learning. *CoRR* abs/1907.10701 (2019). arXiv:1907.10701 http://arxiv.org/abs/1907.10701

[40] Feng Xiong, Fengbin Tu, Man Shi, Yang Wang, Leibo Liu, Shaojun Wei, and Shouyi Yin. 2020. STC: Significance-aware Transform-based Codec Framework for External Memory Access Reduction. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*. 1–6. doi:10.1109/DAC18072.2020.9218522

[41] Zhuoquan Yu, Huidong Ji, Yue Cao, Junfu Wu, Xiaoze Yan, Lirong Zheng, and Zhuo Zou. 2025. DuoQ: A DSP Utilization-aware and Outlier-free Quantization for FPGA-based LLMs Acceleration. In *2025 62nd ACM/IEEE Design Automation Conference (DAC)*. 1–7. doi:10.1109/DAC63849.2025.11132816

[42] Yunxiang Zhang, Gengchen Sun, Lizhi Fang, Biao Sun, and Wenfeng Zhao. 2025. NN-AdderNet: Nonnegative and Sparse Weight Optimization Towards Ultra-Low Bitwidth AdderNet Quantization and Compression. In *2025 62nd ACM/IEEE Design Automation Conference (DAC)*. 1–7. doi:10.1109/DAC63849.2025.11132907

[43] Yu Zhang, Mingzi Wang, Lancheng Zou, Wulong Liu, Hui-Ling Zhen, Mingxuan Yuan, and Bei Yu. 2024. MixPE: Quantization and Hardware Co-design for Efficient LLM Inference. arXiv:2411.16158 [cs.LG] https://arxiv.org/abs/2411.16158