AttenPU: An Area Efficient Attention Processor with Reconfigurable FP8 Precision and Dataflow

Qiawei Zheng
Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
Shenzhen, China
University of Chinese Academy of Sciences
Beijing, China
zhengqiawei23@mails.ucas.ac.cn

Pu Zhou
Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
Shenzhen, China
zhoupu717@foxmail.com

Zheng Wang*
Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
Shenzhen, China
zheng.wang@siat.ac.cn

Zhuoyu Wu
Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
Shenzhen, China
wuzhuoyu11@gmail.com

Yike Li
Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
Shenzhen, China
yike.li@ucdconnect.ie

Zhihao Du
Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
Shenzhen, China
caseytylerg@gmail.com

Chao Chen
Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
Shenzhen, China
chao.chen@siat.ac.cn

Yongkui Yang
Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
Shenzhen, China
yk.yang@siat.ac.cn

Wenqi Fang
Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
Shenzhen, China
wq.fang@siat.ac.cn

Anupam Chattopadhyay
Nanyang Technological University
Singapore, Singapore
anupam@ntu.edu.sg

Abstract

Efficient numerical representation is crucial for deep learning accelerators, especially for large language models (LLMs). The 8-bit-floating-point (FP8) data representation achieves higher precision and fewer quantization efforts than integer, which has been proven inevitable in attention-based accelerators for LLMs. Therefore, area-efficient design techniques for FP8 play a central role in lowering LLMs chip's budget. This paper presents AttenPU, which is built upon reconfigurable FP8 units and supports E4M3 for inference and E5M2 for training. Bidirectional dataflow is exploited to enable AttenPU to interact with FP32 coprocessor to reduce latency. The design achieves a low FP8-to-INT8 area ratio of 1.63×, an area efficiency of 193.5 GFLOPS/mm^2, with an 87.05% reduction in the latency of RTX3090 GPU.

CCS Concepts

• **Hardware → Application specific processors**; • **Computer systems organization → Reconfigurable computing**.

Keywords

Attention Accelerator, FP8, Area Efficient, Reconfigurable

ACM Reference Format:
Qiawei Zheng, Pu Zhou, Zheng Wang, Zhuoyu Wu, Yike Li, Zhihao Du, Chao Chen, Yongkui Yang, Wenqi Fang, and Anupam Chattopadhyay. 2025. AttenPU: An Area Efficient Attention Processor with Reconfigurable FP8 Precision and Dataflow. In *Great Lakes Symposium on VLSI 2025 (GLSVLSI '25), June 30–July 02, 2025, New Orleans, LA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3716368.3735201

*Corresponding author: Zheng Wang, zheng.wang@siat.ac.cn

GLSVLSI '25, New Orleans, LA, USA

ACM ISBN 979-8-4007-1496-2/25/06
https://doi.org/10.1145/3716368.3735201

1 Introduction

With the rapid advancement of artificial intelligence (AI), large language models have been widely applied in Artificial Intelligence

Generated Content (AIGC), playing a significant role in enhancing the intelligence level of human-computer interaction and driving cutting-edge technologies [1][2]. At the core of these models lies the attention mechanism, a critical component that enables their superior performance[3][4]. Building upon attention, the Transformer model dynamically assesses the importance of input information, allowing the model to capture long-range dependencies and complex contextual information, thereby significantly improving its understanding and generation capabilities.

While LLMs achieve impressive performance, their large parameter size and computational requirements pose challenges in terms of storage and processing capabilities. In CNNs, INT8 post-training quantization effectively reduces complexity, but Transformer models exhibit outlier-heavy distributions that exceed INT8's representational range [5]. FP8 offers a broader dynamic range without requiring data-sensitive calibration, allowing simpler and more accurate quantization. Meanwhile, FP8 training is gradually becoming a trend, with Deepseek R1 also adopting this approach[6].

Typical attention accelerators adopt arrays with 4,096 FP8 MAC units, which consume significantly more resources than INT8. Although [7] proposes alternative FP8 formats for ALU efficiency, these are often incompatible with formats used in mainstream training frameworks. [8] presents a compact E3M4 design (3-bit exponent, 4-bit mantissa) using low-bit adders, but Transformer models usually use E4M3 for inference and E5M2 for training [9]. [10]employs design space exploration to generate corresponding accelerated matrix multiplication units for different data types, and it remains challenging to develop more efficient and reconfigurable multiplyaccumulate structures tailored to data characteristics.

Besides low-cost FP8 array, operators in attention such as QK^T involve matrix transposition, which traditionally incurs extra memory access and on-chip buffering, increasing time consumption and hardware resource usage. To accelerate transpose, TranCIM [11] introduces a bitline-transpose structure, but additional circuitry is still required. Furthermore, the low-cost implementation of Softmax operator, which constrains the output data to a range between 0 and 1, usually introduces approximation errors[12]. However, achieving high precision in Softmax usually introduces long latency which dominates the timing of other attention operators. The architecture of efficient attention accelerators faces the following challenges:

- Low physical cost FP8 MAC array supporting both E4M3 and E5M2 formats.
- Minimizing the implementation of transposition.
- Achieving high precision and low latency in Softmax.

To address the aforementioned challenges, this work proposes an attention accelerator named AttenPU with the following novelties:

- Converting standard FP8 numbers to block-based unsigned fixed-point numbers and offloading portions of multiplication to external units, achieving low-overhead and high-precision FP computation.
- Splitting intermediate data into high and low bits and reusing the adder logic to adjust precision between E4M3 and E5M2.
- Exploiting intrinsic bidirectional dataflow in the Output Stationary (OS) MAC array to achieve matrix transposition with zero physical overhead.
- Computing Softmax with a SIMD-style FP32 coprocessor and pipelining Softmax with MAC array to balance long latency.

Taking advantage of the above design principles, AttenPU reduces the area ratio of FP8 to INT8 from 4.07× to 1.63×, achieving a computing area density of 193.5 GFLOPS/mm^2 under 40nm standard-cell technology. Coupling with an FP32 coprocessor executing vectorized RISC-V style Softmax instructions, AttenPU achieves an 87.05% reduction in latency compared to RTX3090 GPU when computing attention kernels.

The rest of the work is organized as follows. Section 2 introduces backgrounds. Section 3 details the methodology for the proposed low-cost FP8 engine. Section 4 illustrates the micro-architectures and dataflow of AttenPU. Section 5 presents the benchmarking and physical results. Section 6 concludes the work.

2 Background

2.1 Characteristics of FP8

Qualcomm's white paper[9] suggests that FP8 networks may be trained and deployed directly in the future, avoiding complex quantization. In this approach, E4M3 is commonly used for forward propagation due to its higher precision and limited dynamic range ($[2^{-9}, 2^8]$), making it suitable for fixed-point conversion. E5M2 is used for backpropagation, offering a wider range ($[2^{-16}, 2^{15}]$) similar to FP16 but with lower precision. Converting E5 to fixed-point requires 32 bits for representation, resulting in 64-bit products and corresponding adders and multiple registers to prevent overflow. However, supporting both E4 and E5 without shared compute logic significantly increases hardware cost.

2.2 Output Stationary Dataflow

In OS dataflow mode, the product results are accumulated within each PE until all input and weight data pass through it. The output typically forms a parallelogram shape, as shown in Fig 1(a) and (b). To facilitate subsequent processing, Fig 1(c) shows that the data is held until the entire row is computed, after which the output matrix represents the multiplication result. The output data is output in the vertical direction, allowing the transposed matrix to be obtained without the full input completion, as shown in Fig 1(d).

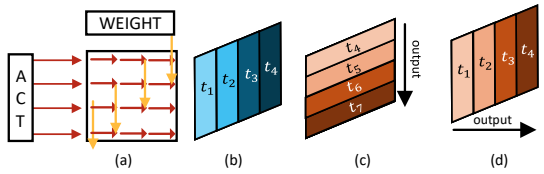

Figure 1: (a) Input dataflow (b) order of data calculation (c) regular output order (d) transposed output order

2.3 Attention Mechanism

The operators in attention [3] are shown in Eq. (1). Attention weights assign varying levels of importance to the elements of the input sequence, thereby capturing contextual information.

$$Attention(Q, K, V) = Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (1)$$

3 Resource-Efficient FP8 MAC Unit

Fig. 2 illustrates the FP computation engine of AttenPU, designed for matrix multiplication and composed of four main components: pre-processing, multiplication and shifting, accumulation, and post-processing. Unlike conventional FP computation, it employs unsigned fixed-point arithmetic for intermediate results, alleviating alignment and accumulation errors. By offloading preprocessing and postprocessing outside the PE array (detailed in Section 4), the design enhances data reuse and leverages sequential output. The FP computation array retains only 4-bit multiplication, shifts, and integer addition, introducing minimal overhead, with 44% of resources offloaded to reduce hardware utilization significantly.

3.1 Pre-processing

Previous works in FP computation often overlook subnormal numbers in the IEEE 754 standard [13], with insufficient experimental evidence to confirm that their elimination does not affect neural network accuracy. When either the activation or weight is subnormal and the other is large, the product usually stays within the representable range of normal numbers.

Therefore, the FP pre-processing primarily achieves the following: handling subnormal numbers, unifying the E4M3 and E5M2 formats, and converting the MANT to the actual value {1.MANT}, padding the high bits of E4 with zeros to form E5, and padding the low bits of M2 with zeros to form M3. The simple pre-processing avoids special cases in subsequent operations. Different data types can share arithmetic units, allowing the array to support multiple data types.

3.2 FP-MAC Unit

3.2.1 Normal Mechanism. In inference, FP8 typically utilizes the E4M3 format, with the computation process shown in Fig. 2 (b) and (c). This study converts the product into an unsigned fixed-point representation to replace FP numbers for accumulation, as shown in Eq.(2). $2^{E_{max}}$ is half of the maximum value of an unsigned fixed-point number, which means there is an overall offset, and N represents the number of elements to be summed.

$$\sum_{i=0}^{N-1}(-1)^S\left(1.M\times 2^E\right)= \sum_{i=0}^{N-1}\left[(-1)^S\{1,M\}\ll E+2^{E_{max}}\right]-N2^{E_{max}} \tag{2}$$

The detailed computation process is as follows: (i) the mantissa of the two operands are multiplied, and their exponents are added to obtain the product's exponent and mantissa; (ii) the mantissa is left-shifted to align with the bit-width of the corresponding exponent, forming a fixed-point number without precision loss; (iii) the value is then adjusted through an offset operation to yield an unsigned fixed-point representation, as shown in Eq. (3).

$$(-1)^S\left(\{1,M\}\ll E\right)+2^{E_{max}}=\begin{cases}100.....1M.....00, S=0\\ 011..(1M)_{2's\ \mathrm{Comp}}..00, S=1\end{cases} \tag{3}$$

During the accumulation phase, fixed-point numbers can be directly processed as integers, obviating the necessity for FP alignment and circumventing the potential for accumulation errors. This approach ensures that precision loss is kept to a minimum. In the case of unsigned numbers, the sole requisite for the recording of overflow occurrences is a register, which effectively prevents underflow during computation.

3.2.2 Reconfigurable Mechanism. Backpropagation during training requires the E5 format, which offers a broader range comparable to FP16 and can be directly converted to fixed-point for processing. However, this doubles the shift and adder resources compared to E4. By modifying the E4M3 MAC structure to convert the product into a block-based unsigned fixed-point format, as shown in Fig. 2 (c) and Eq. (4), reconfigurability is achieved with minimal additional resources. N_1 and N_2 respectively represent the occurrence counts of the lower and higher bits.

$$\begin{cases} O=\sum_{i=0}^{N_1-1}\left[(-1)^S\{1,M\}\ll E+2^{E_{max}}\right] \\ \quad+\sum_{i=0}^{N_2-1}\left[(-1)^S\{1,M\}\ll E+2^{E_{max}}\right]\cdot 2^{32} \\ \quad-\left(N_1\cdot 2^{E_{max}}+N_2\cdot 2^{E_{max}+32}\right) \\ N=N_1+N_2 \end{cases} \tag{4}$$

This supports E4, E5, and mixed formats, with left-shifting considering only the lower five bits of the exponent for block selection. E[5] (the 6th bit) is used to select between the high and low 32 bits, with both sharing the same adder. Compared to accumulating with E4, an additional 32-bit register and selector are introduced to store and select high and low bit data.

3.3 Post-processing

According to Eq. (2) or Eq.(4), after accumulation, the data must be adjusted by subtracting the offset to obtain the true value, followed by extracting the sign of the result and taking its absolute value. Based on the definition of FP numbers, the position of the first occurrence of "1" in the accumulation result is detected, and adding the bias yields the final exponent value. The N consecutive bits following the first "1" represent the mantissa, where both the bias and the value of N depend on the input data type (e.g., E4 or E5) and the output data precision (e.g., FP8 or FP32).

4 AttenPU Micro-architecture

Fig. 3 shows the architecture, where data is fetched from DDR and DRAM via the AXI bus, pre-processed, and stored in weight and activation buffers. After entering the PE array, FP operations are performed with output modes chosen as Fast Shift (FSHIFT) or Normal Shift (SHIFT) based on the layer configuration. Post-processing completes the computation, and once QK^T computation is done, the data is sent to the Float Point Unit (FPU) for Softmax and returned to the buffer for SV processing.

4.1 FP MAC Array

As detailed in Section 3, Fig. 3 (a) illustrates that activations and weights are pre-processed and stored in the activation and weight buffers, respectively. The computational array consists of 64 micro-processing units, each containing 1 PE LEADER and 63 PE WORKERS. Activations are distributed across PEs, while weights are

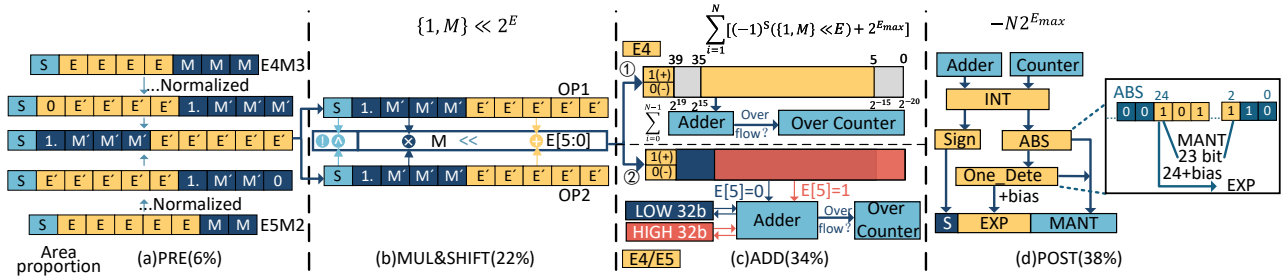

Figure 2: Illustration of proposed FP8 MAC in four stages with area proportion, achieving only 1.61× area of INT8 MAC.

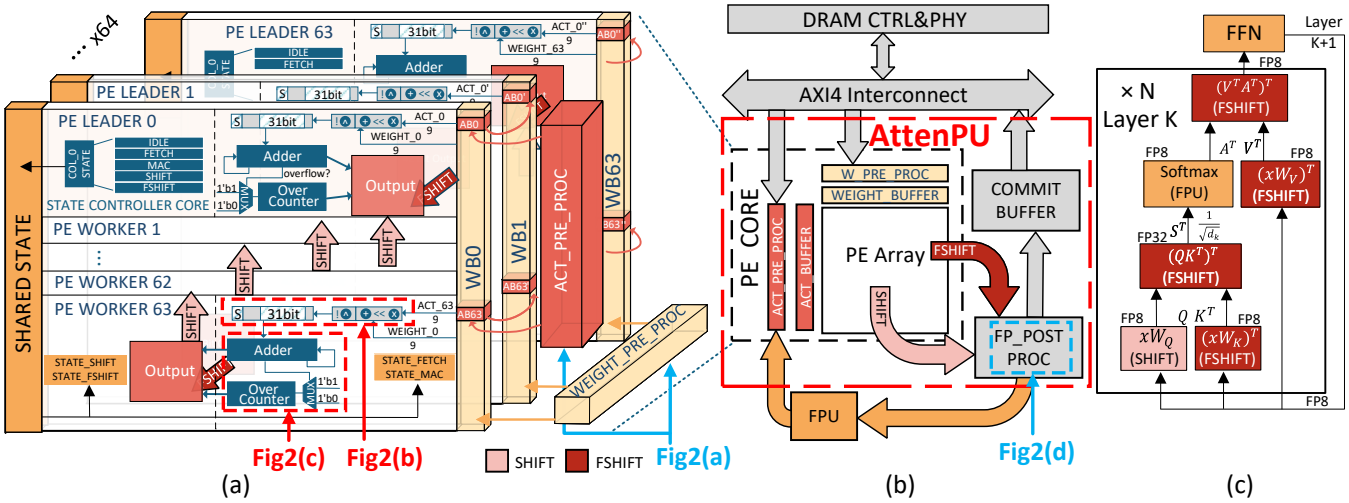

Figure 3: Illustration of (a) core architecture (b) AttenPU and FPU interaction (c) attention operators (color mapped in (b)).

shared and passed to the next micro-processing unit in the following clock cycle. The PE LEADER manages state transitions (IDLE, Computing, and Data Movement), with PEs operating independently and passing results sequentially between units.

In SHIFT mode, the micro-unit delays output until all computations complete, ensuring result consistency with full matrix multiplication. In contrast, FSHIFT mode enables immediate output from the first unit and next-cycle output from the second, significantly accelerating throughput.

FSHIFT essentially switches the direction of the PE array output to either right or bottom (as in Fig. 1 (d)). Its hardware implementation involves minimal components, primarily featuring bidirectional shift wires and various extra states in the controller FSM. When the sequence length exceeds 64, each FSHIFT output corresponds to a block-wise transpose. Therefore, by modifying the starting address of each block's output storage, the entire matrix transpose can be completed. This mode produces the **transposed matrix**, and the bi-directional output array can select the appropriate output mode based on the matrix requirements, quickly providing either the original or transposed matrix.

4.2 Softmax

In addition to the FP8 PE array, an SIMD-based FP32 unit (FPU) is designed to accelerate Softmax, SiLu, and LayerNorm. As shown in Fig. 4, the FPU comprises 16 single-precision units organized into four lanes, supporting arithmetic, accumulation, min/max, and exponential operations. Given the importance of the exponential operation in Softmax computation, we implemented a fully pipelined exponential unit that leverages a two-level LUT and a three-term Taylor series approximation[14], accommodating an input range of (-32, 32). Additionally, the FPU is equipped with vector, scalar, and integer register files and provides load/store access to local and shared caches through address mapping.

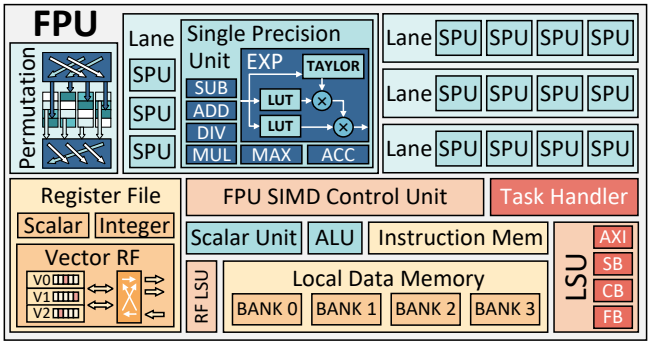

Figure 4: Overview of FPU architecture components.

FLAT partitions the Softmax operation by behavioural granularity [15]. Similarly, the FPU uses the PE array size as the granularity, processing 64×4-row blocks. It adopts a FlashAttention-inspired Softmax computation [16], using max-value subtraction to avoid

numerical instability and applying block-wise execution to improve parallelism. The three-stage algorithm is detailed in Algorithm 1 and Fig. 5.

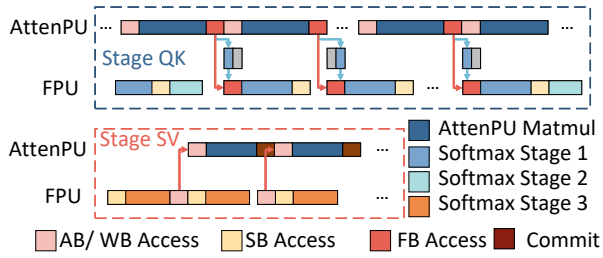

Figure 5: Timing of data exchange between MAC and FPU. Stage QK: AttenPU sends result of $Q \times K^T$ to FPU via FB. Stage SV: FPU sends result of $Softmax$ to AttenPU for $S \times V$ via WB. AttenPU and FPU are pipelined. AB/ WB/ SB/ FB: activation / weight / stage / float buffer

Algorithm 1: Vectorized Softmax processor on FPU

for $j \leftarrow 1$ **to** n **do**
  Stage 1:
  for $i \leftarrow 1$ **to** n **do**
    $A_i^{(j)} = Q_i K_j^T$ // AttenPU output A
    $M_i^{(j)} = \text{vmax}(A_i^{(j)})$ // partial maximum
    $P_i^{(j)} = \text{vexp}(A_i^{(j)} - M_i^{(j)})$ // exponent value
    $S_i^{(j)} = \text{vacc}(P_i^{(j)})$ // partial sum
  Stage 2:
  $M^{(j)} = M_1^{(j)}, \cdots, M_n^{(j)}$ and $S^{(j)} = S_1^{(j)}, \cdots, S_n^{(j)}$
  $M_j = \text{vmax}(M^{(j)})$ // global maximum
  $K_j = \text{vexp}(M^{(j)} - M_j)$ // scaling factor vector
  $S_j = \text{vacc}(K_j \circ S^{(j)})$ // global sum
  $L_j = \text{vdiv}(K_j, S_j)$ // final coefficient to normalize
  Stage 3:
  $P^{(j)} = P_1^{(j)}, \cdots, P_n^{(j)}$
  $O_j = L_j \circ P^{(j)}$ // final result
return $O = O_1, \cdots, O_m$

Stage1: For each block of AttenPU's computation results, the partial row-wise maximum values are calculated. Each element is subtracted by the corresponding maximum value and sent to the exponential unit, while the partial sums are computed simultaneously;
Stage2: Based on the partial maximum values and partial sums from Stage 1, we calculate the global maximum value and global sum, which are used to compute the linear coefficients for each block's exponential result;
Stage3: The exponential results from Stage 1 are multiplied by the corresponding linear coefficients to compute the final result.

For input sequences longer than 64, processing is performed in $N \times N$ blocks of length 64. After processing each batch, AttenPU stores the results in an intermediate cache and issues a task command via a FIFO queue to activate the FPU, subsequently resuming the matrix computation. To hide the FPU's latency, Stages 1 and 2 are executed in parallel with AttenPU's QK^T computation, and Stage 3 with the SV computation. Shifting linear multiplication to the SV fetch stage balances the FPU's load, reduces response time during the QK^T phase, and aligns intermediate cache usage with AttenPU's speed. The trade-off is a small, acceptable storage overhead to store the linear coefficients for each block.

4.3 Dataflow

In the attention computation process, the combination of SHIFT and FSHIFT modes enables efficient output calculation. SHIFT mode generates output across the micro-unit, while FSHIFT generates outputs within the same micro-unit, resulting in a transposed matrix without an explicit transpose operation. This reduces memory access and accelerates matrix multiplication. Fig 3 (c) illustrates the dataflow and output mode selection, ensuring efficient data movement through mode switching.

To enhance efficiency, AttenPU adapts data width during computation, using FP8 for input and weight data, and unsigned fixed-point for intermediate results. This reduces resource consumption while maintaining precision. For Softmax, FP32 input and FP8 output are used to balance precision and resource utilization, addressing the challenge of low-bit Softmax degradation.

5 Evaluation

The FP32 FPU coprocessor and AttenPU with FP8 MAC array are completely designed in Verilog. Logic synthesis at 300MHz clock frequency under SMIC 40nm standard-cell library is performed for physical estimations of AttenPU.

5.1 Performance

By eliminating inter-unit synchronization, FSHIFT further accelerates matrix multiplication. Meanwhile, the well-designed interaction mechanism between the AttenPU and FPU minimizes the latency of the Softmax process as much as possible.

Using the BERT-base model with input word vectors of length 128, a comparison of attention latency with commercial processors is presented in Table 1. Experimental results show that AttenPU achieves an 87.05% reduction in latency compared to the NVIDIA RTX3090 GPU.

Table 1: Latency comparison with state-of-the-art designs.

	Matrix Dimensions	This work	RTX3090	Apple M1
Softmax(QK^T)	(128×64)× (64x128)	4.12us	39.87us	151.14us
SV	(128×128)× (128×64)	3.41us	18.27us	57.85us
Total Latency	-	7.53us	58.14us	208.99us
Freq (MHz)	-	300	1395	2064

5.2 Hardware resources

Table 2 lists the area breakdown of each component within a single FP8 multiply-accumulate unit. Table 3 shows the area of a single

multiply-accumulate unit under different precision conditions, revealing that the FP processing engine in our design is significantly smaller than a pure FP array. Additionally, Fig. 2 shows that 44% of the workload is offloaded outside the array, resulting in minimal overhead. Therefore, the FP processing engine in AttenPU performs FP operations with only a slightly larger area than the INT8 unit.

Table 2: Area decomposition of proposed FP8 MAC unit.

	Pre	Mac &Shift	Add	Post	Total
Area(μm^2)	158.0	520.7	813.7	927.2	2419.6
Proportion	6%	22%	34%	38%	100%

Table 3: Area comparison with relevant MAC unit.

	AttenPU FP[1]	INT[2]	FP[3]	FP8 MAC[17]
Area(μm^2)	2419.6	869.4	3903.2	3081.6

[1]FP8 MUL and FP32 ACC (optimized design according to Fig 2).
[2]INT8 MUL and INT32 ACC.
[3]FP8 MUL and FP16 ACC (standard design).

Fig 6 compares the area and power consumption of 64 × 64 AttenPU array with INT and standard FP arrays when the computation units are switched. Without considering the peripheral circuitry, the area for FP operations in this work is reduced to **1.63x** that of the INT array. Including the peripheral circuits, the total area is only **1.31x** larger, and the power consumption is reduced from 2.17x to **1.57x**, which proves AttenPU's much more compact design compared to standard FP8 array.

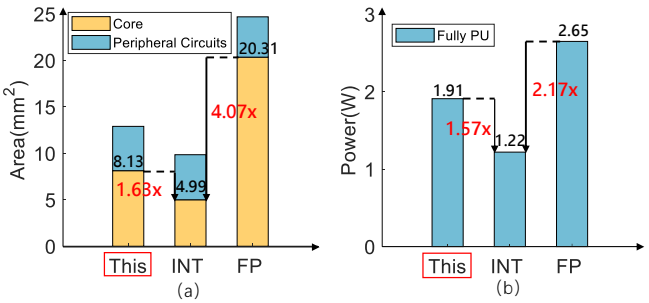

Figure 6: Area and power comparison for 64 × 64 MAC array Synthesized results using SMIC 40nm standard cell

Table 4: Resource utilization of FP32 FPU on FPGA

FPGA type	AMD UltraScale Alveo U280	
Synthesized frequency	250MHz	
Configuration	4 SPU	16 SPU
LUT	30182	98322
FF	25279	88235
BRAM	18	48
DSP	72	276
Balanced AttenPU's FP8 MAC Topology	32x32	64x64

The FPU is currently implemented on FPGA IPs and thus only FPGA resources are reported in Table 4. It is noted that several FPU configurations are shown to balance the latency of AttenPU according to timing diagram in Fig. 5.

5.3 Accuracy

We evaluated the inference of the BERT-base model on the SST-2 dataset, comparing the mean squared error (MSE) of each layer's attention output in FP8-E4M3 (AttenPU) with standard and straight FP32 (Python) results, as shown in Fig 7. Due to FP8's effective data range coverage, the MSE remained small, increasing slightly across attention layers. Experimental findings demonstrated that the model's accuracy decreased slightly from 92.43% in FP32 to 92.09% in FP8, achieving substantial reductions in memory and hardware resource usage while maintaining high accuracy.

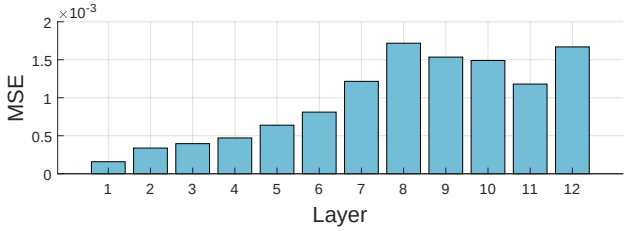

Figure 7: MSE of AttenPU outputs on BERT-base model.

5.4 Benchmarking with Other Designs

We benchmark our design with other FP8 AI engines. As shown in Table 5, the computational density of AttenPU excels among relevant designs when extrapolating the area of design to the same technology node. AttenPU achieves 2.496 TFLOPS at 300 MHz under 40nm technology, with a compact area of 12.90 mm^2. This translates to a high computational density of 193.5 GFLOPS/mm^2, demonstrating the effectiveness of our FP8 E4M3/E5M2 architecture and accelerator-level optimization.

Table 5: Benchmarking with state-of-the-art designs.

	This work	JSSC' 2021[17]	JSSC' 2023[18]	ISCAS' 2024[19]
Tech	40nm	40nm	28nm	28nm
Freq (MHz)	300	180	340	160
Precision	FP8 E4M3/E5M2	FP8	FP8/16	FP4/8+ FP16/32
Area (mm^2)	12.90	6.23	16.4	1.84
TFLOPS	2.496	0.567	3.7(sparsity)	0.157
GFLOPS/mm^2	193.5	91.0	225.6	85.3

6 Conclusion

In this work, we designed an FP8 processing engine that is efficient and precision-adjustable in area. We integrated it into an attention accelerator, AttenPU, which supports bidirectional output data flow. AttenPU reduces the FP array area from 4.07x of the integer version to 1.67x, and power consumption from 2.17x to 1.57x. Coupling with an FP32 coprocessor, the system achieves an 87.05% reduction in latency for operators combining MatMul, Transpose and Softmax compared to RTX3090 GPU.

Acknowledgments

This work was funded by NSFC under Grant No.62372442, State Key Lab of Processors, Institute of Computing Technology, CAS under Grant No.CLQ202308, Guangdong Basic and Applied Basic Research Foundation under Grant No.2023A1515012842, Shenzhen Science and Technology Program under Grant No.JCYJ20220818101607015.

References

[1] Jacob Devlin. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[2] Minghan Li, Shuai Li, Xindong Zhang, and Lei Zhang. 2024. Univs: Unified and universal video segmentation with prompts as queries. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3227–3238.

[3] A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* (2017).

[4] Sneha Chaudhari, Varun Mithal, Gungor Polatkan, and Rohan Ramanath. 2021. An attentive survey of attention models. *ACM Transactions on Intelligent Systems and Technology (TIST)* 12, 5 (2021), 1–32.

[5] Andrey Kuzmin, Mart Van Baalen, Yuwei Ren, Markus Nagel, Jorn Peters, and Tijmen Blankevoort. 2022. Fp8 quantization: The power of the exponent. *Advances in Neural Information Processing Systems* 35 (2022), 14651–14662.

[6] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948* (2025).

[7] Wenzhe Zhao, Qiwei Dang, Tian Xia, Jingming Zhang, Nanning Zheng, and Pengju Ren. 2023. Optimizing FPGA-Based DNN accelerator with shared exponential floating-point format. *IEEE Transactions on Circuits and Systems I: Regular Papers* (2023).

[8] Chen Wu, Mingyu Wang, Xinyuan Chu, Kun Wang, and Lei He. 2021. Low-precision floating-point arithmetic for high-performance fpga-based cnn acceleration. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)* 15, 1 (2021), 1–21.

[9] Mart van Baalen, Andrey Kuzmin, Suparna S Nair, Yuwei Ren, Eric Mahurin, Chirag Patel, Sundar Subramanian, Sanghyuk Lee, Markus Nagel, Joseph Soriaga, et al. 2023. FP8 versus INT8 for efficient deep learning inference. *arXiv preprint arXiv:2303.17951* (2023).

[10] Jinming Zhuang, Zhuoping Yang, and Peipei Zhou. 2023. High performance, low power matrix multiply design on acap: from architecture, design challenges and dse perspectives. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1–6.

[11] Fengbin Tu, Zihan Wu, Yiqi Wang, Ling Liang, Liu Liu, Yufei Ding, Leibo Liu, Shaojun Wei, Yuan Xie, and Shouyi Yin. 2022. TranCIM: Full-digital bitline-transpose CIM-based sparse transformer accelerator with pipeline/parallel reconfigurable modes. *IEEE Journal of Solid-State Circuits* 58, 6 (2022), 1798–1809.

[12] Nazim Altar Koca, Anh Tuan Do, and Chip-Hong Chang. 2023. Hardware-efficient Softmax Approximation for Self-Attention Networks. In *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1–5.

[13] Mariko Tatsumi, Silviu-Ioan Filip, Caroline White, Olivier Sentieys, and Guy Lemieux. 2022. Mixing low-precision formats in multiply-accumulate units for DNN training. In *2022 International Conference on Field-Programmable Technology (ICFPT)*. IEEE, 1–9.

[14] Zbigniew Hajduk. 2017. High accuracy FPGA activation function implementation for neural networks. *Neurocomputing* 247 (2017), 59–61.

[15] Sheng-Chun Kao, Suvinay Subramanian, Gaurav Agrawal, Amir Yazdanbakhsh, and Tushar Krishna. 2023. Flat: An optimized dataflow for mitigating attention bottlenecks. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*. 295–310.

[16] Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. 2024. Flashattention-3: Fast and accurate attention with asynchrony and low-precision. *arXiv preprint arXiv:2407.08608* (2024).

[17] Jeongwoo Park, Sunwoo Lee, and Dongsuk Jeon. 2021. A neural network training processor with 8-bit shared exponent bias floating point and multiple-way fused multiply-add trees. *IEEE Journal of Solid-State Circuits* 57, 3 (2021), 965–977.

[18] Shreyas Kolala Venkataramanaiah, Jian Meng, Han-Sok Suh, Injune Yeo, Jyotishman Saikia, Sai Kiran Cherupally, Yichi Zhang, Zhiru Zhang, and Jae-Sun Seo. 2023. A 28-nm 8-bit Floating-Point Tensor Core-Based Programmable CNN Training Processor With Dynamic Structured Sparsity. *IEEE Journal of Solid-State Circuits* 58, 7 (2023), 1885–1897.

[19] Wei Lu, Han-Hsiang Pei, Jheng-Rong Yu, Hung-Ming Chen, and Po-Tsang Huang. 2024. A 28nm Energy-Area-Efficient Row-based pipelined Training Accelerator with Mixed FXP4/FP16 for On-Device Transfer Learning. In *2024 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1–5.