StreamAttention: Energy-Efficient and High-Utilization Attention on Systolic Hardware

Anonymous Authors[1]

Abstract

Attention is the dominant compute cost in modern transformers and grows quadratically with the sequence length. FlashAttention (Dao et al., 2022) cuts the memory cost through tiling and online softmax computation, but modern accelerators are optimized for matrix multiplication and offload softmax to much lower-throughput vector units, stalling the pipeline. We present StreamAttention, an accelerator co-designed with FlashAttention-2 (Dao, 2024) that sustains continuous streaming of operands on a single systolic array of multiply-accumulate (MAC) units. We map online attention to MAC recurrences that fit the systolic dataflow exactly, and evaluate the softmax exponential as a Chebyshev polynomial with the same MAC units. A four-stage pipeline overlaps every attention phase with no idle cycles between tiles, keeping operands continuously streaming through the array. StreamAttention achieves 95–98 % utilization against ~40 % for the closest peer SystolicAttention (Lin et al., 2025). We demonstrate that the Chebyshev approximation has negligible impact on Llama-3.2-1B (WikiText-103 Perplexity), ViT-Base (ImageNet Top-1), and BERT-Large (SQuAD F1). Fusing softmax onto the array eliminates per-tile SRAM round-trips of the score and softmax matrices, cutting per-layer attention energy by up to ~2.8× and average attention power by up to ~2.9× vs. our baseline. This comes at the cost of a ~34 % area overhead and +6 % power on pure matrix multiplication over a standard systolic-based matrix unit. We synthesize against the open source SkyWater 130 nm process design kit for area and power; SRAM energies are from CACTI 7 at 90 nm.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country.**AUTHORERR: Missing \icmlcorrespondingauthor.**

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

1. Introduction

Self-attention is the dominant compute cost in modern transformers (Vaswani et al., 2017) and scales as $O(n^2 d)$ in the sequence length n. In LLM inference this cost concentrates in the prefill phase (Patel et al., 2024), which sets the time-to-first-token a user perceives. Materializing the intermediate score matrix $S \in \mathbb{R}^{n \times n}$ on chip quickly becomes intractable. For example, context windows reaching a million tokens (Gemini Team, Google, 2024; Anthropic, 2026) would require several terabytes of on-chip memory, far exceeding any realistic system. FlashAttention (Dao et al., 2022; Dao, 2024; Shah et al., 2024; Zadouri et al., 2026) addresses this by processing the matrices in blocks and computing softmax *online*. This avoids materializing the full score matrix, and turns attention from a memory-bound to a compute-bound workload.

The remaining bottleneck is the hardware itself. Modern accelerators such as TPUs (Jouppi et al., 2017; 2023; Google, 2023), AWS NeuronCore (Amazon Web Services, 2024), and GPUs (Choquette et al., 2021) are heavily optimized for matrix multiplication, but offload non-linear operations such as softmax to a separate on-chip vector unit with much lower throughput (roughly 16× lower on the NVIDIA A100 (Dao, 2024)). The separation also forces SRAM round-trips between the two units, increasing power consumption. Specialized FlashAttention implementations remedy the throughput gap through asynchrony and dataflow specialization (Shah et al., 2024; Zadouri et al., 2026), raising utilization to 70–75%, but do not eliminate the round-trips. In datacenters, the lower utilization is amortized by batching across concurrent requests, where throughput rather than per-tile efficiency drives the design. The SRAM-roundtrip energy, in contrast, scales quadratically with the sequence length per request and is simply paid through the electricity bill. On edge devices however, where the model runs locally with energy as the binding constraint, every idle cycle and every round-trip is paid in latency and a shorter battery life.

Systolic arrays (Kung & Leiserson, 1979; Kung, 1982) have recently emerged as an effective architecture for accelerating attention, due to their inherent parallelism, data reuse, and pipelining. Recent systolic accelerators either

reach near-full utilization at the cost of dedicated softmax units (Wang et al., 2023; 2025), or sustain only ~40% utilization on a single array (Lin et al., 2025).

To address this gap, we present **StreamAttention**, an accelerator that runs every step of the attention layer on a single systolic array without sacrificing utilization or its ability to do general matrix multiplication. Figure 1 shows the proposed design, and outlines how we co-design it with the FlashAttention algorithm. Specifically, our contributions are:

- **Reformulate online attention in terms of a sequence of MAC-friendly recurrences** that map directly onto a systolic array (Section 2).
- **Design a single-array architecture that supports all attention computations** on a shared MAC core (Section 2).
- **Evaluate the softmax exponential via Chebyshev polynomials** on the same MAC units and demonstrate that it has negligible impact on downstream tasks. To our knowledge this is the first use of a Chebyshev approximation for the softmax exponential in hardware.
- **Build a four-stage pipelined dataflow that overlaps all attention steps** and sustains continuous operand streaming (Section 2), reaching 95–98% utilization at sequence lengths $S \geq 2048$ while reducing SRAM round-trips of intermediate matrices. This cuts per-layer attention energy by up to ~2.8× and power by ~2.9× vs. our baseline (Section 3).

2. StreamAttention Design

2.1. Background: Systolic Array

A 2D systolic array (Kung & Leiserson, 1979; Kung, 1982) is a regular grid of processing elements (PEs) that compute the matrix product $C = AB$ by distributing the elementwise sum $c_{ij} = \sum_k a_{ik} b_{kj}$ across the grid. Operands enter from the west and north edges, and each PE forwards values to its east and south neighbors. Per cycle, every PE evaluates one step of the recurrence

$$c_{ij}^{(k+1)} = c_{ij}^{(k)} + a_{ik}\, b_{kj} \tag{1}$$

on its local multiply-accumulate (MAC) unit. The array is configured by choosing which term of Equation 1 stays resident in the PE. In the *weight-stationary* (WS) variant used in the TPU (Jouppi et al., 2017), b_{kj} is preloaded into the PE while a_{ik} streams west-to-east and the partial sum $c_{ij}^{(k)}$ accumulates south, draining at the bottom edge (Figure 2c). In the *output-stationary* (OS) variant, $c_{ij}^{(k)}$ stays in a local accumulator and is incrementally updated by having both operands stream through (Figure 2a). In both variants, operands enter the array skewed by one cycle per row and column, so each tile pays $O(N)$ fill-and-drain cycles around the N^2 MAC cycles of useful work. This overhead vanishes if successive tiles are streamed back-to-back (Section 2.3).

2.2. Online attention as MAC recurrences

Mapping any computation onto a systolic array requires expressing every step as a MAC recurrence following the form of Equation 1. We use i for output rows, j for output columns, and k for the inner reduction. Standard scaled-dot-product attention computes

$$S = \frac{QK^\top}{\sqrt{d}}, \quad P = \exp\big(S - \mathrm{rowmax}(S)\big),$$
$$O = PV, \quad Attention = \mathrm{diag}\big(\mathrm{rowsum}(P)\big)^{-1} O,$$

with $S, P \in \mathbb{R}^{n\times n}$ and the final output in $\mathbb{R}^{n\times d}$. Each entry of the unnormalized output O is a weighted sum

$$O_{ij} = \sum_k \exp(S_{ik} - m_i)\, V_{kj}, \qquad m_i = \max_k S_{ik},$$

which on a systolic array we want to express as an accumulator update of the form $c^{(k+1)} = c^{(k)} + a^{(k)} b^{(k)}$. The obstacle is that m_i is a global row maximum, but at step k a PE will only have information from the first k elements.

This is essentially the same problem FlashAttention solves on a *block level*, so we adopt their well-known solution (Milakov & Gimelshein, 2018; Dao et al., 2022; Alexandridis et al., 2025). We maintain a running maximum $m_i^{(k)}$ over elements seen so far and rescale the accumulator each time the max grows. Having first computed $m_i^{(k+1)} = \max(m_i^{(k)}, S_{ik})$, the update becomes

$$O_{ij}^{(k+1)} = O_{ij}^{(k)} \exp\big(m_i^{(k)} - m_i^{(k+1)}\big) + V_{kj} \exp\big(S_{ik} - m_i^{(k+1)}\big). \tag{2}$$

By noting that exactly one of the two arguments to exp is zero while the other is the negative difference between the current element (S_{ik}) and the maximum at the previous step ($m_i^{(k)}$), we can define

$$\tilde{S}_{ik} = -\big|S_{ik} - m_i^{(k)}\big|, \qquad \tilde{P}_{ik} = \exp(\tilde{S}_{ik}),$$

so that Equation 2 collapses to a single (conditional) MAC operation per element:

$$O_{ij}^{(k+1)} = \begin{cases} O_{ij}^{(k)} \cdot \tilde{P}_{ik} + V_{kj}, & S_{ik} \geq m_i^{(k)} \\ O_{ij}^{(k)} + V_{kj} \cdot \tilde{P}_{ik}, & S_{ik} < m_i^{(k)}. \end{cases} \tag{3}$$

Thus, if the current element updates the running max, we rescale the accumulator. Otherwise we scale the current element before adding. The only bookkeeping is the sign-bit

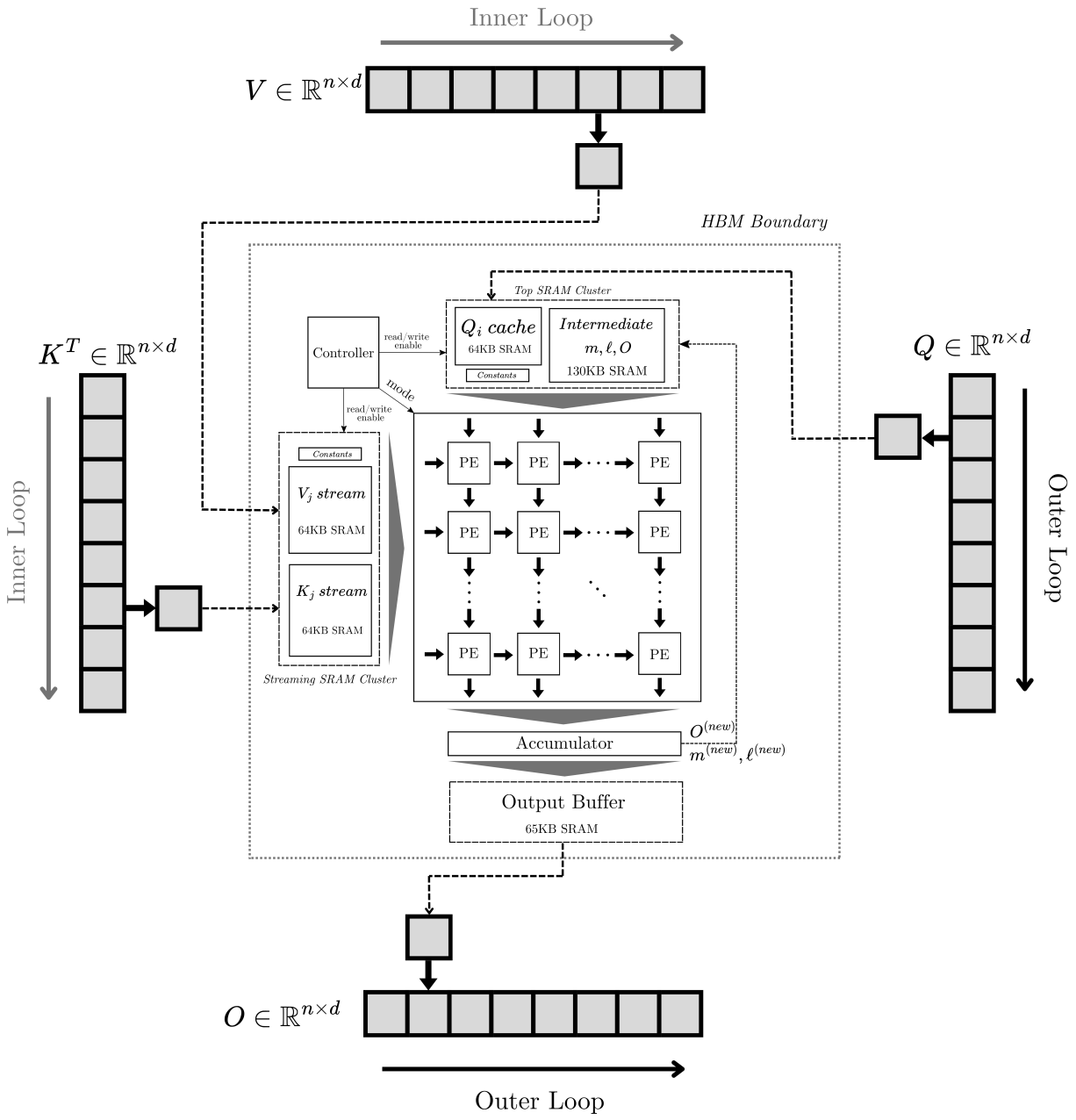


Figure 1. System-level architecture of the StreamAttention accelerator and co-design with FlashAttention. Every step of attention is run on a single $N \times N$ systolic array of multiply-accumulate units. The array is surrounded by double-buffered SRAM clusters along the west edge (K_j, V_j, constants) and the north edge (Q_i, intermediate m, ℓ, O, constants), and the south edge (m, ℓ, O). These clusters support continuous streaming of operands into and out of the compute unit. A controller broadcasts a per-cycle mode signal propagated along the data through the array, sequencing the four pipeline stages without idle cycles between tiles. Data multiplied in FP16, accumulated in FP32.

of $S_{ik} - m_i^{(k)}$, used to select which operand to scale. The row-normalizer ℓ_i follows the same recurrence with $V_{kj} = 1$. The final attention output $\mathrm{diag}(\ell)^{-1}O$ is computed in the downstream accumulator. Since this is only performed once per outer loop, its cost is negligible against the $O(S^2d)$ on-array work.

2.3. Four-stage systolic dataflow

We decompose the computation of each attention tile into four stages executed within the array: (i) output-stationary evaluation of the score matrix $S = QK^\top$, (ii) online row-wise max normalization to obtain $\tilde{S}$ while updating the running rowmax, (iii) in-place scaling and exponentiation to

produce $\tilde{P}$, and (iv) weight-stationary online accumulation of O and ℓ from $\tilde{P}$ and V. Figure 2 shows the dataflow on the array. All four stages run on the same $N \times N$ array of PEs, with the microarchitecture of each PE detailed in Appendix A. We now describe each stage in turn.

Stage 1: $S = QK^\top$ in output-stationary mode (os). Rows of K are streamed horizontally from the west and rows of Q enter vertically from the north. Each PE accumulates $S_{ij} = \sum_p Q_{ip}K_{jp}$ in place. After $3N$ cycles the score tile S is resident across the array, with rows aligned vertically.

Stage 2: in-place max (MAX). To form $\tilde{S}_{ij} = -|S_{ij} - m_i^{(j)}|$ we stream the running maxes $m_i^{(k)}$ from the previous tile from the top (so they propagate along the rows) and the constant -1 from the west. Then, the PE at position (i, j) computes $S_{ij} + (-1) \cdot m_i^{(j)}$ in place, and forcing the sign-bit of the FP result high yields $-|\cdot|$ in a single cycle. The 1-bit sign-of-difference is latched in each PE for use in stage 4. The new maxes, $m_i^{(k+N)}$, exit the bottom of the array.

Stage 3: in-place scale and exponentiation (SCALE+EXP). We first scale by $\gamma = 1/(\sqrt{d}\ln 2)$ in one cycle to convert from natural to base-2, i.e., $e^{\tilde{S}/\sqrt{d}} = 2^{\tilde{S}/(\sqrt{d}\ln 2)}$. The exponential is then evaluated on the same MAC datapath in p additional cycles via a Chebyshev polynomial of degree p (Section 2.4). The result $\tilde{P}_{ij}$ overwrites $\tilde{S}_{ij}$ in the local PE register.

Stage 4: $O = PV$ in weight-stationary mode (ws). With $\tilde{P}$ resident in the array, we stream rows of V from the west and the running output $O^{(k)}$ from the top. The 1-bit flag from stage 2 selects which side of the MAC gets rescaled, implementing Equation 3 per cycle. The row normalizers ℓ_i are produced by streaming a row of ones first.

Pipelining. The four stages run back-to-back without idle cycles between tiles. Each stage can begin once the first PE of the previous stage has produced its result — e.g., MAX starts N cycles into $QK^\top$ — so the stages overlap as shown in Figure 3. Single-tile latency is $4N + p + 3$ cycles, compared to $5N + 10$ in SystolicAttention (Lin et al., 2025). Pipelining successive tiles by starting the next $QK^\top$ during $O = PV$ yields an amortized period of $2N + p + 3$ per tile.

2.4. Approximating the exponential

After the MAX stage we have $\tilde{S}_{ij} \le 0$ resident in the PE and want to compute $\tilde{P}_{ij} = \exp(\tilde{S}_{ij}/\sqrt{d})$. Note that the score matrix S has not been scaled by $1/\sqrt{d}$, so we combine it with this step. We first convert to base 2, which is more efficient to implement in hardware (Schraudolph, 1999): $e^{\tilde{S}_{ij}/\sqrt{d}} = 2^{\tilde{S}_{ij}/(\sqrt{d}\ln 2)}$. The constant $\gamma = 1/(\sqrt{d}\ln 2)$ is streamed into the array during the SCALE step in one MAC cycle.

Letting $x = \tilde{S}_{ij} \cdot \gamma$, we range-reduce $x = x_i + x_f$ with $x_i = \mathrm{trunc}(x) \in \mathbb{Z}_{\le 0}$ and $x_f \in (-1, 0]$, so that $2^x = 2^{x_i} \cdot 2^{x_f}$. The integer part x_i is applied cheaply at the PE output by adding x_i to the FP exponent of the result. Only 2^{x_f} on the bounded interval $(-1, 0]$ requires numerical approximation. Following (Lin et al., 2025), a small split unit at the PE input separates x into x_i and x_f by shifting the mantissa.

We approximate 2^{x_f} by a Chebyshev polynomial of degree p,

$$P_p(x_f) = \bar{c}_0 + \bar{c}_1 x_f + \cdots + \bar{c}_p x_f^p, \tag{4}$$

chosen because Chebyshev minimizes the worst-case error over the interval, in contrast to a Taylor expansion which minimizes error around a single point (Muller, 2016). To reuse the MAC datapath, we evaluate P_p via Horner's method. For $p = 3$ it becomes

$$P_3(x_f) = c_0 + x_f\big(c_1 + x_f(c_2 + x_f \cdot c_3)\big). \tag{5}$$

The first cycle streams c_2 and c_3 together, the remaining $p - 1$ cycles each stream one coefficient, for a total of p MAC cycles through the same FP16×FP32 datapath as the matmul stages. The integer part x_i is reapplied at the PE output by a combine unit. We choose $p = 3$, sufficient for end-to-end accuracy on real workloads (Section 3).

3. Evaluation

3.1. Utilization

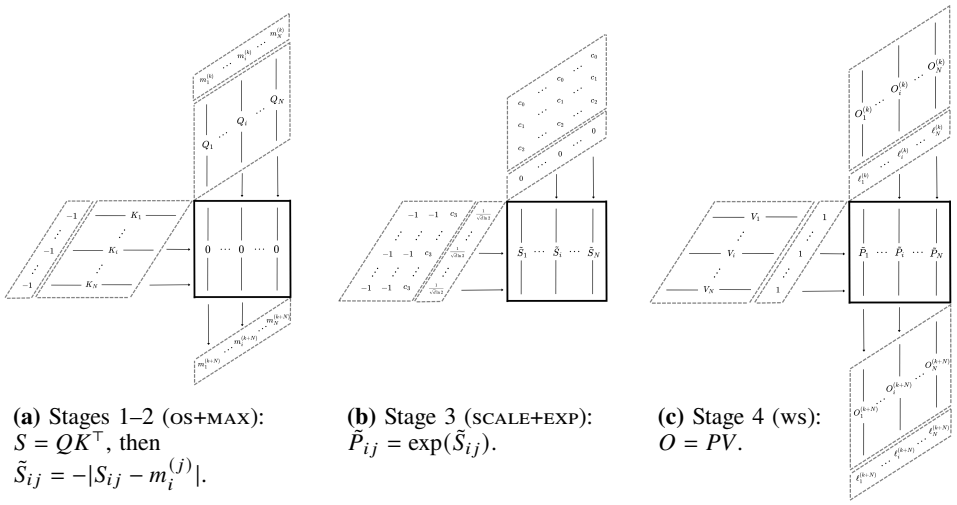
We measure utilization as the ratio of useful FLOPs to peak FLOPs over the full tile schedule, following (Dao et al., 2022):

$$\text{Utilization} = \frac{\text{Total FLOPs}}{\text{Peak FLOPs/cycle} \times \text{Total cycles}}, \tag{6}$$

where total FLOPs $= 4S^2d$ (Dao, 2024) (the two matmuls $QK^\top$ and PV) and peak throughput is $2N^2$ FLOPs per cycle (one addition and one multiplication per PE). From Section 2.3, each tile takes $C_{\text{tile}} = 2N + 3 + p$ cycles — $2N$ for the two GEMMs, three for max, scale, and row-sum, and p for the Chebyshev evaluation. With tiles fully pipelined and no gaps between them, the only overhead is $4N$ fill-and-drain cycles at the boundaries, giving total cycles $T = (S/N)^2 C_{\text{tile}} + 4N$. Substituting into Equation 6 with $d = N$,

$$\text{Utilization} = \frac{2N}{C_{\text{tile}} + 4N^3/S^2} \xrightarrow{S\to\infty} \frac{2N}{2N + 3 + p},$$

which for $N = 128$ and $p = 3$ gives an asymptotic ceiling of 97.7%. The fill/drain overhead $4N^3/S^2$ shrinks quadratically in S.



(a) Stages 1–2 (OS+MAX): $S = QK^\top$, then $\tilde{S}_{ij} = -|S_{ij} - m_i^{(j)}|$.

(b) Stage 3 (SCALE+EXP): $\tilde{P}_{ij} = \exp(\tilde{S}_{ij})$.

(c) Stage 4 (WS): $O = PV$.

Figure 2. Systolic dataflow across the four stages of one attention tile, all running on the same $N \times N$ array. (a) Output-stationary $QK^\top$ followed by in-place max via -1 broadcast from the west and the running max from the top; the new running max is piped out the bottom. (b) In-place γ-scaling followed by Horner evaluation of the Chebyshev polynomial (Section 2.4), producing $\tilde{P}$ in-place. (c) Weight-stationary $O = PV$ with conditional rescaling on a new running max using the in-place $\tilde{P}$; this stage also produces the row-normalizers ℓ.

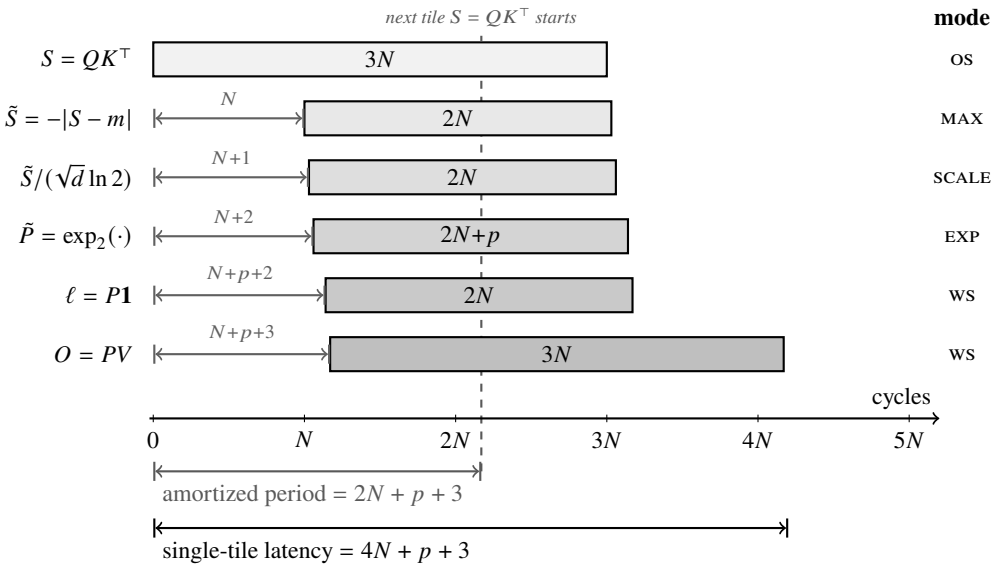


Figure 3. Pipeline schedule for a single attention tile. Bar widths are proportional to array-level duration. The dashed line marks when the next tile's $S = QK^\top$ overlaps the current tile, giving the amortized period $2N + p + 3$.

Table 1 compares utilization against SystolicAttention (Lin et al., 2025), the only prior work that runs the entire attention layer on a single systolic array. StreamAttention reaches 95% at $S = 2048$ and 97% at $S = 4096$, approximately 2.5× SystolicAttention's ∼39%. The gap arises because SystolicAttention does not overlap consecutive tiles, paying $\sim 5N$ cycles per tile of which only $2N$ are GEMM. StreamAttention pipelines all four phases on the same array and overlaps them with the next tile's $S = QK^\top$, so the only non-GEMM overhead per tile is the $3 + p$ cycles for max, scale, exp, and rowsum. The Chebyshev approximation incurs a bit-exact MRE of $4\text{–}6 \times 10^{-3}$ vs an FP64 reference at $S \in \{2048, 4096, 8192, 16384\}$, dominated by FP16 quantization rather than the polynomial itself (full sweep in Appendix D).

Table 1. FLOPs/s utilization against sequence length S at $N = d = 128$, measured from Verilator RTL simulation. SystolicAttention numbers from (Lin et al., 2025).

S	2048	4096	8192	16384
StreamAttention	95.1%	97.0%	97.5%	97.6%
SystolicAttention	39%	39%	39%	39%

3.2. Task-level accuracy

We measure whether the Chebyshev approximation affects downstream task accuracy. We replace the attention module of three pretrained transformers with a bit-exact emulator of the StreamAttention datapath. We report Llama-3.2-1B perplexity on the WikiText-103 validation set (~218K tokens) with a 1024-token causal window, ViT-Base/16 top-1 on the ImageNet-1k validation set (50K images), and BERT-Large (whole-word-masking, SQuAD-finetuned) F1 on the SQuAD v1.1 validation set (10,570 examples). For each model we sweep four kernels: an FP32 reference and three hardware-equivalent variants (Chebyshev degree-3, degree-4, and 8-segment PWL).

Table 2 reports the result. All three approximations stay within 1×10^{-4} of the FP32 reference perplexity, shift ImageNet top-1 by at most -0.002 pp (one flipped prediction in 50K images), and shift SQuAD F1 by at most +0.017. The cheapest approximation we tested (Chebyshev degree-3 at 3 MAC cycles) is indistinguishable from the FP32 reference across all three workloads.

The choice of approximation method at the kernel level is further detailed in Appendix B.

3.3. Power and energy

We synthesize the StreamAttention array and a plain weight-stationary (WS) array of FP16×FP32 MAC units against the SkyWater 130 nm open process design kit (SkyWater Technology and Google, 2020), using Yosys 0.62 (Wolf, 2013) mapped to the high-density standard-cell library (`sky130_fd_sc_hd`). OpenSTA (Cherry, 2018) reports dynamic power at the nominal corner under a 10% default-activity assumption. Off-array SRAM access energies come from CACTI 7 (Balasubramonian et al., 2017), configured as a 32 KB scratchpad at 90 nm (the closest CACTI-supported node). Both arrays go through the identical flow at the same 100 ns clock period, so the difference isolates the cost of fusing softmax onto the array.

A WS-only array cannot fuse softmax. Per layer per head, it writes the per-tile S and P matrices to SRAM and reads them back, paying $4S^2$ extra word accesses. Multiplying by the CACTI access energies gives an avoided energy of $51.8\,S^2$ pJ per layer per head. Both this energy and the StreamAttention layer runtime scale as S^2, so the ratio gives an S-independent average spill power of 32.4 mW (see Appendix C for full derivation). Figure 4(a) reports the per-layer energy breakdown across S: the spill dominates the WS budget already at $S = 1024$, and on-array softmax fusion cuts total energy by ~2.8× at $S = 4096$. Figure 4(b) reports the corresponding average power: 48.9 mW for WS (7.5 mW array-dynamic + 9.0 mW Q, K, V, O traffic + 32.4 mW spill) against StreamAttention's 17.0 mW (8.0 mW array-dynamic + the same 9.0 mW Q, K, V, O traffic), a ~2.9× reduction.

3.4. Hardware cost

Table 3 reports per-PE cell counts for three configurations: a plain WS array of FP16×FP32 MAC units, the hybrid OS+WS array with online max comparison and γ-scaling, and the full StreamAttention array with the in-place $\exp_2$ pipeline. The hybrid array adds +10.7% over the WS baseline — mainly the MAC input muxes and the FP16↔FP32 conversion paths needed by the WS-with-correction stage. Adding the $\exp_2$ pipeline contributes a further +23.8%, dominated by the split unit, fractional encoder, and combine unit (510 cells, 15.3% of the WS baseline). On pure GEMM, the StreamAttention PE also pays +6.3% dynamic power from the unused mode logic. The total +34.4% area is the price for the ~2.9× attention-power reduction and the near-optimal utilization.

4. Related Work

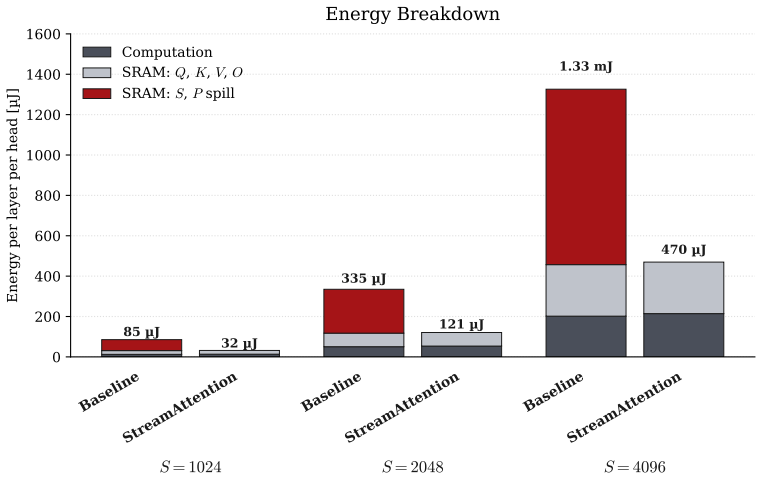
FlashAttention lineage. The online softmax recurrence (Milakov & Gimelshein, 2018; Rabe & Staats, 2021) is the algorithmic foundation FlashAttention builds on. FlashAttention (Dao et al., 2022) fuses tiled $QK^\top$, online softmax, and PV into a single IO-aware GPU kernel. FlashAttention-2 (Dao, 2024) restructures the loop nest for parallelism and defers softmax normalization; FlashAttention-3 (Shah et al., 2024) and FlashAttention-4 (Zadouri et al., 2026) specialize to the Hopper and Blackwell architectures, respectively. StreamAttention is co-designed around FlashAttention-2.

Systolic attention accelerators. COSA (Wang et al., 2023) and COSA Plus (Wang et al., 2025) couple two cooperating systolic arrays through a dedicated softmax unit between $QK^\top$ and PV, reaching ~95% utilization across the full multi-head attention (MHA) layer; Agile Systolic MHA (Chen et al., 2026) runs MHA on a single array but still relies on external softmax and layer-norm units. SystolicAttention (Lin et al., 2025) is the closest peer: it removes the dedicated softmax unit by streaming row-max and PWL exponential coefficients through a single array, but does not pipeline successive tiles and saturates at ~40% utilization. FuseMax (Nayak et al., 2024) extends the operator-fusion idea of FLAT (Kao et al., 2023) from GPUs to a 2D PE array. Other accelerators target sparsity (Lu et al., 2021; Wang et al., 2021; Ham et al., 2020; 2021) or low-precision (Islamoglu et al., 2023), which are orthogonal to our work.

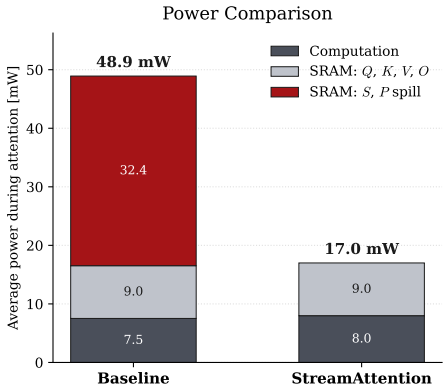
Positioning. StreamAttention combines three threads. The FlashAttention algorithm (Dao et al., 2022) supplies the tiled, online-softmax structure that our hardware is co-

Table 2. Task-level accuracy of three pretrained transformers with the StreamAttention kernel against an FP32 reference. PPL on WikiText-103, Top-1 on ImageNet-1k, F1 on SQuAD v1.1.

Kernel	Llama PPL	Δ	ViT Top-1	Δ	BERT F1	Δ
FP32 reference	11.2549	—	80.326	—	93.159	—
Chebyshev deg-3	11.2548	−0.0001	80.324	−0.002	93.176	+0.017
Chebyshev deg-4	11.2548	−0.0001	80.324	−0.002	93.162	+0.003
PWL 8-segment	11.2549	0.0000	80.324	−0.002	93.165	+0.006



(a) Energy per layer per head, decomposed into computation, I/O traffic, and the spill of S and P to SRAM.



(b) Average power during one attention layer.

Figure 4. Energy and power during one attention layer. SkyWater 130 nm, $N = d = 128$, $p = 3$, 100 ns clock. SRAM access energies from CACTI 7 at 90 nm.

Table 3. Per-PE cell count for three $N \times N$ array configurations, normalized to the WS baseline. Synthesized with Yosys 0.62 against SkyWater 130 nm, hierarchy preserved.

Component	Cells	Δ%	Cumul.%
WS array (MAC + accumulator + passthrough)	3337	—	100.0
Hybrid array (OS+WS + attention control flow)			
Max comparison + sign-bit forcing OS dataflow + γ-scaling + FP conv.	+356	+10.7	110.7
+ *in-place* $\exp_2$ *pipeline*			
Split unit (integer/fractional decomposition)	+288	+8.6	119.3
Fractional encoder (priority encoder)	+137	+4.1	123.4
Combine unit (exponent adjustment)	+85	+2.5	126.0
Horner accumulator register	+32	+1.0	126.9
Extra FP32→FP16 truncations + mux overhead	+251	+7.5	134.4
StreamAttention array (total)	**4486**	+34.4	134.4

designed around. We improve on SystolicAttention (Lin et al., 2025)'s in-array attention dataflow to reach near-perfect utilization, while retaining the general matmul capability of a standard weight-stationary engine (Jouppi et al., 2017; 2023). To achieve this, we draw on Low-Cost FlashAttention (Alexandridis et al., 2025)'s fused $\exp \cdot V$ insight, replacing their exponential approximation with a Chebyshev polynomial that we evaluate using our in-array MAC units.

5. Conclusion and Limitations

We presented **StreamAttention**, an accelerator that runs every step of the attention layer on a single systolic array of MAC units. The two algorithmic ingredients — mapping online attention to MAC recurrences and evaluating the softmax exponential as a Chebyshev polynomial through the same FP16×FP32 datapath — collapse the entire attention pipeline onto one hybrid processing element. A four-stage pipeline runs the four phases back-to-back without inter-tile stalls, sustaining 95–98% utilization and cutting per-layer attention energy by ∼2.8× vs. a weight-stationary baseline at the cost of +34% area. On edge devices, where energy is the binding constraint, this translates directly into longer battery life and lower latency for local LLM inference.

Broader Impact. The same on-array softmax fusion that helps edge devices also reduces datacenter electricity demand: the SRAM-roundtrip energy scales quadratically per request with sequence length, and cloud LLM endpoints pay this cost in their power bill. As LLM inference grows into a measurable share of regional grid load (International Energy Agency, 2024), per-request reductions compound across the trillions of tokens served daily. A ∼2.9× attention-power cut in the dominant prefill phase narrows both the operational cost of cloud inference and the upward pressure on local electricity prices that AI workloads increasingly exert.

Limitations. The synthesis flow is pre-layout (no measured wire capacitance) on the SkyWater 130 nm open PDK, with CACTI 7 SRAM access energies modeled at 90 nm; absolute energy figures at 7–16 nm production nodes would scale, but the relative comparison between StreamAttention and the WS baseline is robust to this. The dynamic-power numbers assume uniform 10% switching activity rather than workload-driven SAIF back-annotation. Taking the design to fabrication would also require optimizing the critical paths inside the hybrid PE, potentially with additional pipeline stages, to hit a competitive target frequency. We have evaluated only forward-pass inference at FP16×FP32; backward pass and lower-precision inference (INT8, FP8) are left to future work. Decode-phase attention — where the sequence dimension is one — is dominated by KV-cache traffic rather than matrix-unit utilization and is not addressed here.

Reproducibility. We release the SystemVerilog RTL of the StreamAttention array and PE, the bit-exact C++ hardware model used for the accuracy sweep, the OpenSTA synthesis flow against the open SkyWater 130 nm PDK, and the plotting scripts for every figure in this paper. The repository `README` documents the toolchain and re-running the experiments end-to-end.

References

Alexandridis, K., Titopoulos, V., and Dimitrakopoulos, G. Low-cost FlashAttention with fused exponential and multiplication hardware operators. arXiv:2505.14314, 2025.

Amazon Web Services. NeuronCore-v3 Architecture. https://awsdocs-neuron.readthedocs-hosted.com/en/latest/about-neuron/arch/neuron-hardware/neuron-core-v3.html, 2024. 128×128 BF16 systolic tensor engine in Trainium2.

Anthropic. Claude Opus 4.7. https://www.anthropic.com/claude/opus, 2026. 1M token context window.

Balasubramonian, R., Kahng, A. B., Muralimanohar, N., Shafiee, A., and Srinivas, V. CACTI 7: New tools for interconnect exploration in innovative off-chip memories. *ACM Transactions on Architecture and Code Optimization (TACO)*, 14(2):14:1–14:25, 2017.

Chen, X., Li, Y., Zhao, B., Liu, Y., Cao, S., and Jiang, Z. An agile systolic array-based hardware accelerator for scalable multi-head self-attention. *IEEE Embedded Systems Letters*, 18(2):90–93, 2026.

Cherry, J. OpenSTA: A gate-level static timing analyzer. https://github.com/parallaxsw/OpenSTA, 2018.

Choquette, J., Gandhi, W., Giroux, O., Stam, N., and Krashinsky, R. NVIDIA A100 tensor core GPU: Performance and innovation. *IEEE Micro*, 41(2):29–35, 2021.

Dao, T. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024.

Dao, T., Fu, D. Y., Ermon, S., Rudra, A., and Ré, C. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Gemini Team, Google. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

Google. TPU v5p. https://docs.cloud.google.com/tpu/docs/v5p, 2023. TPU v5p product documentation. 8960-chip pod, reconfigurable optical interconnect.

Ham, T. J., Jung, S. J., Kim, S., Oh, Y. H., Park, Y., Song, Y., Park, J.-H., Lee, S., Park, K., Lee, J. W., and Jeong, D.-K. A^3: Accelerating attention mechanisms in neural networks with approximation. In *IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2020.

Ham, T. J., Lee, Y., Seo, S. H., Kim, S., Choi, H., Jung, S. J., and Lee, J. W. ELSA: Hardware-software co-design for efficient, lightweight self-attention mechanism in neural networks. In *ACM/IEEE International Symposium on Computer Architecture (ISCA)*, 2021.

International Energy Agency. Electricity 2024 – analysis and forecast to 2026. IEA, Paris, 2024. https://www.iea.org/reports/electricity-2024.

Islamoglu, G., Scherer, M., Paulin, G., Fischer, T., Jung, V. J. B., Garofalo, A., and Benini, L. ITA: An energy-efficient attention and softmax accelerator for quantized transformers. In *IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, 2023.

Jouppi, N. P., Young, C., Patil, N., Patterson, D., et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA)*, pp. 1–12, 2017.

Jouppi, N. P., Kurian, G., Li, S., Ma, P., Nagarajan, R., Nai, L., Patil, N., Subramanian, S., Swing, A., Towles, B., Young, C., Zhou, X., Zhou, Z., and Patterson, D. A. TPU v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings. In *ACM/IEEE International Symposium on Computer Architecture (ISCA)*, 2023.

Kao, S.-C., Subramanian, S., Agrawal, G., Yazdanbakhsh, A., and Krishna, T. FLAT: An optimized dataflow for mitigating attention bottlenecks. In *ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2023.

Kung, H. T. Why systolic architectures? *Computer*, 15(1): 37–46, 1982.

Kung, H. T. and Leiserson, C. E. Systolic arrays (for vlsi). In Duff, I. S. and Stewart, G. W. (eds.), *Sparse Matrix Proceedings 1978*, pp. 256–282. SIAM, 1979.

Lin, J., Li, Y., Chen, G., and Bourgeat, T. Systolicattention: Fusing flashattention within a single systolic array. *arXiv preprint arXiv:2507.11331*, 2025.

Lu, L., Jin, Y., Bi, H., Luo, Z., Li, P., Wang, T., and Liang, Y. Sanger: A co-design framework for enabling sparse attention using reconfigurable architecture. In *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2021.

Milakov, M. and Gimelshein, N. Online normalizer calculation for softmax. arXiv preprint arXiv:1805.02867, 2018.

Muller, J.-M. *Elementary Functions: Algorithms and Implementation*. Birkhäuser, 3 edition, 2016.

Nayak, N. et al. FuseMax: Leveraging extended einsums to optimize attention accelerator design. In *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2024. arXiv:2406.10491. TODO: complete author list.

Patel, P., Choukse, E., Zhang, C., Shah, A., Goiri, Í., Maleki, S., and Bianchini, R. Splitwise: Efficient generative LLM inference using phase splitting. In *Proceedings of the 51st Annual International Symposium on Computer Architecture (ISCA)*, 2024.

Rabe, M. N. and Staats, C. Self-attention does not need $O(n^2)$ memory. arXiv preprint arXiv:2112.05682, 2021.

Schraudolph, N. N. A fast, compact approximation of the exponential function. *Neural Computation*, 11(4):853–862, 1999.

Shah, J., Bikshandi, G., Zhang, Y., Thakkar, V., Ramani, P., and Dao, T. Flashattention-3: Fast and accurate attention with asynchrony and low-precision. *arXiv preprint arXiv:2407.08691*, 2024.

SkyWater Technology and Google. SkyWater Open Source PDK. https://github.com/google/skywater-pdk, 2020.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Wang et al. COSA: Co-operative systolic arrays for multi-head attention mechanism in neural network using hybrid data reuse and fusion methodologies. In *ACM/IEEE Design Automation Conference (DAC)*, 2023. TODO: complete author list.

Wang et al. COSA Plus: Enhanced co-operative systolic arrays for attention mechanism in transformers. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2025. TODO: complete author list.

Wang, H., Zhang, Z., and Han, S. SpAtten: Efficient sparse attention architecture with cascade token and head pruning. In *IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2021.

Wolf, C. Yosys – a free Verilog synthesis suite. In *Proceedings of the 21st Austrian Workshop on Microelectronics (Austrochip)*, 2013.

Zadouri, T., Hoehnerbach, M., Shah, J., Liu, T., Thakkar, V., and Dao, T. Flashattention-4: Algorithm and kernel pipelining co-design for asymmetric hardware scaling. *arXiv preprint arXiv:2603.05451*, 2026.

A. Hybrid Processing Element

The four-stage dataflow of Section 2.3 requires a single PE that can run in five logical modes: OS for the $S = QK^\top$ accumulation, MAX for the in-place subtraction and sign-bit force, SCALE for the γ scaling, EXP for the Horner steps that produce $\tilde{P}$, and WS for the rescale-or-add update during $O = PV$. The PE is the same physical hardware across all five.

Figure 5 shows the internal datapath to support these modes. A multiplier ($\otimes$) and an adder ($\oplus$) form the multiply-accumulate (MAC) core. Three mode-driven multiplexers select the MAC operands. The multiplier's left input chooses between the input data x_{in} (default), the local accumulator register Acc (for rescaling accumulated $O_{ij}^{(k)}$), or the exponential accumulator register Exp Acc (for exp computation). A mux on the multiplier's other input chooses between y_{in} (OS), Acc (WS), or the fractional part x_{frac} from the split unit when computing exponential. The mux on the adder's input chooses between Acc (OS and MAX), y_{in} (EXP, SCALE, WS), x_{in} (WS), or zero (init). A fourth mux at the bottom selects which value is stored in the local accumulation buffer and is either the result of the MAC operation or the computed exponential. A fifth mux selects which value is sent south and chooses between y_{in} and the local accumulated value (used for max comparison). All mux selectors and register write enables are driven by the central *Controller* block, which receives the per-cycle mode_{in} signal and propagates it one cycle later through the *Mode* register as mode_{out}, so the mode signal traverses the array in lockstep with the data. The controller also uses the MSB of the output of the MAC to determine during mode MAX if a new maximum value was encountered. Some details related to converting between FP16 and FP32 are hidden for readability.

Furthermore, the PE holds four registers and one passthrough. *Acc* is the PE-local accumulation register. During the OS stage it accumulates S_{ij}, during MAX it is overwritten with $\tilde{S}_{ij}$, which it holds until the end of EXP, at which point it is replaced with $\tilde{P}_{ij}$. During WS it participates as the rescale-term in the rescale-or-add update. *Exp Acc* is the Horner accumulator used only during EXP, holding the partial polynomial value r_k between successive Horner steps. *x reg* and *y reg* are the systolic passthrough registers that hold the outputs x_{out}, y_{out} for one cycle so that the array meets timing along the east–west and north–south wires. The 1-bit sign-of-difference flag captured during MAX is held inside the Controller and used to select the operand routing in the subsequent WS stage.

The split and combine units handle the in-place base-2 exponentiation. The split unit decomposes Acc into a signed 8-bit integer part x_{int} and a 13-bit fractional part, and a small priority encoder downstream of split rebuilds the fractional part as a normalized FP value $x_{\text{frac}} \in [-1, 0]$ that is fed to the multiplier during EXP. The combine unit takes the final Horner result and adds x_{int} directly to its FP exponent field, producing $2^{x_{\text{int}}} \cdot P_p(x_{\text{frac}})$. The MAX stage uses the same MAC but forces the sign bit of the result high before the value is written into Acc, yielding $-|S_{ij} - m_i^{(j)}|$ in a single cycle.

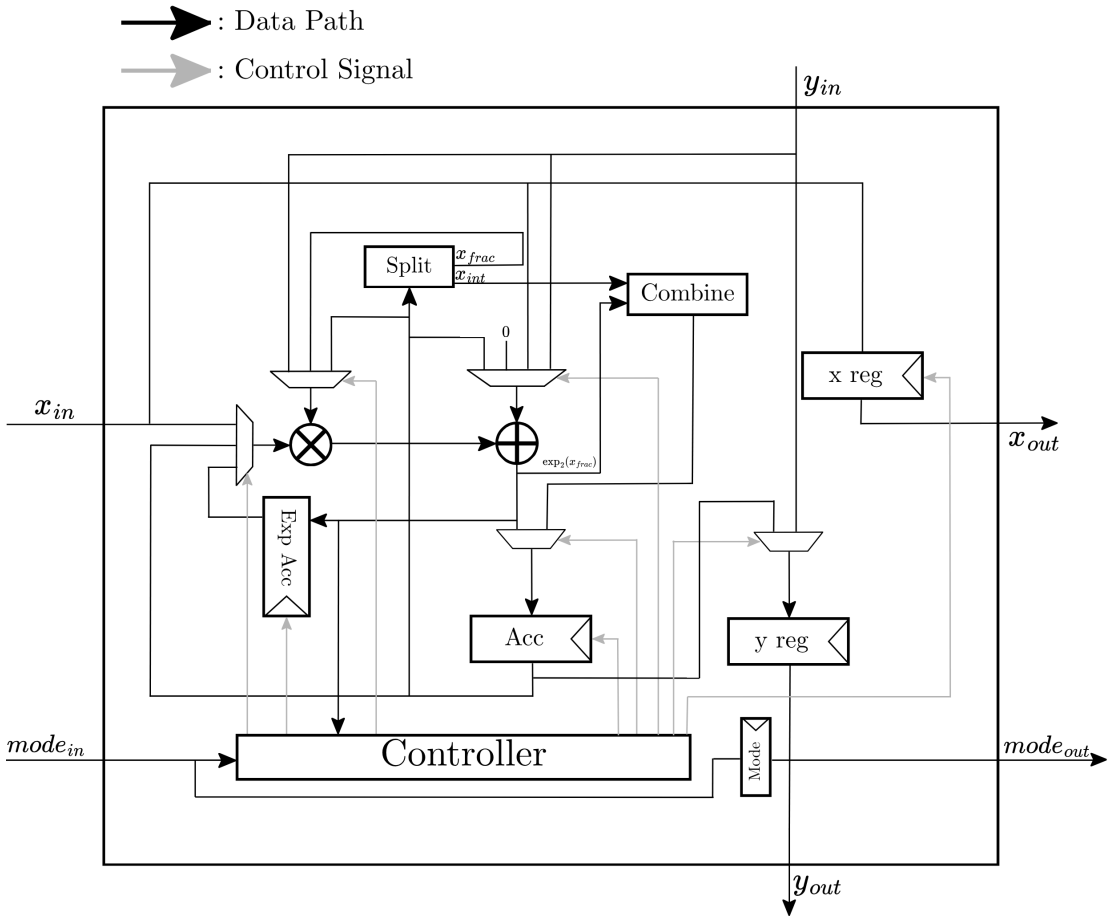


Figure 5. StreamAttention hybrid PE microarchitecture. Black arrows are data; grey arrows are control. The MAC core ($\otimes + \oplus$) is shared across all five modes; mode-driven muxes (driven by the Controller) select the MAC operands and the register paths each cycle. *Acc* is the resident PE register and holds S_{ij}, then $\tilde{S}_{ij}$, then $\tilde{P}_{ij}$ across the stages; *Exp Acc* is the Horner accumulator used during EXP; *x reg* and *y reg* are the systolic passthrough registers; *Mode* forwards the per-cycle mode signal. The split unit decomposes Acc into x_{int} and x_{frac}, and the combine unit re-applies x_{int} to the FP exponent of the polynomial result during EXP.

B. Choice of Approximation

Recall from Section 2.4 that each PE must evaluate 2^{x_f} for $x_f \in [-1, 0]$ as part of the in-place exponentiation stage. We consider three families of approximation: uniform piecewise-linear (PWL), Chebyshev polynomial via Horner's method, and Chebyshev via Clenshaw's algorithm. All three methods reuse the existing MAC unit, and Figure 6 compares their mean relative error (MRE) and maximum relative error (MaxRE) over $x_f \in [-1, 0]$ as a function of allocated MAC cycles.

Horner reaches its floor of MRE $\sim 1.1 \times 10^{-4}$ and MaxRE $\sim 6.9 \times 10^{-4}$ after only 3 MAC cycles (degree 3). Beyond that, additional terms yield little to no benefit as the FP32→FP16 truncation applied to the running accumulator at each multiply step limits precision. Clenshaw reaches a noticeably lower floor of MRE $\sim 4.0 \times 10^{-5}$ and MaxRE $\sim 2.0 \times 10^{-4}$ at 8 MAC cycles (degree 4). Each Clenshaw step consumes two MAC operations, so the last two terms are accumulated in FP32 before the next FP16 truncation, which lets Clenshaw retain more precision than Horner under the same MAC datapath. PWL converges much more slowly. At 16 MAC cycles (16 linear segments) its error matches the FP16 exact 2^x baseline (MRE $\sim 1.8 \times 10^{-4}$, MaxRE $\sim 4.9 \times 10^{-4}$, dashed) but does not go meaningfully below it.

Recall that Horner evaluates $p(x) = c_p x^p + \cdots + c_0$ via the recurrence

$$r_k = r_{k-1} \cdot x_f + c_{p-k}, \quad r_0 = c_p, \tag{7}$$

while Clenshaw evaluates the same polynomial in its Chebyshev basis via

$$d_k = 2x_f \cdot d_{k-1} - d_{k-2} + a_{n-k}, \quad d_0 = d_{-1} = 0. \tag{8}$$

We choose Horner evaluation going forward because it maps directly onto our MAC datapath. Each cycle performs

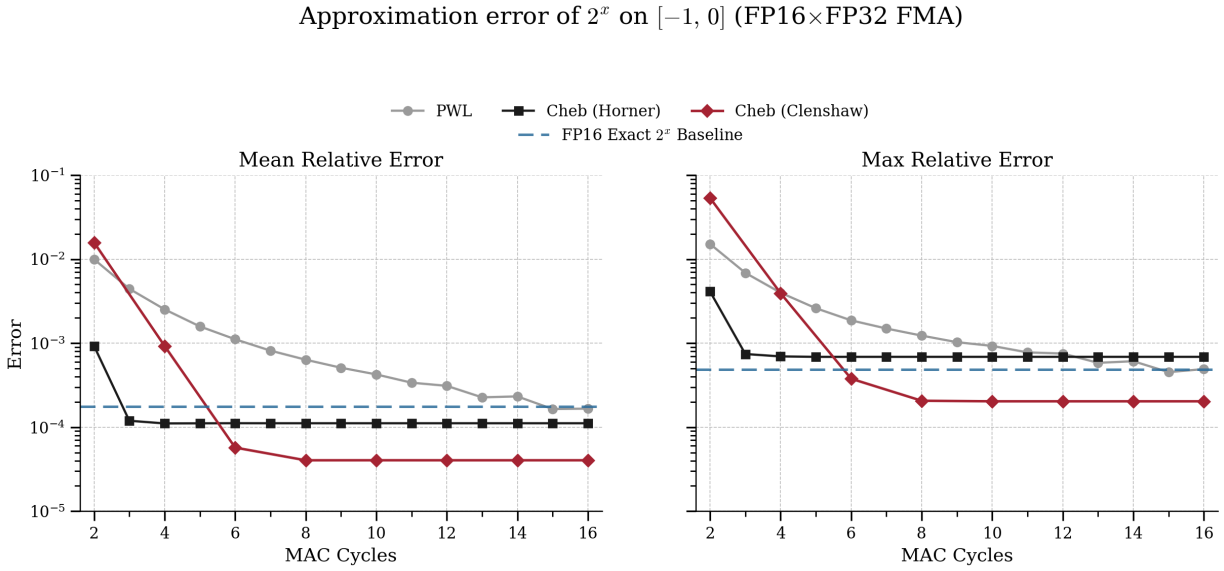


Figure 6. Approximation error of 2^{x_f} on $[-1, 0]$ under FP16×FP32 MAC precision, as a function of MAC cycles.

a single $acc \leftarrow acc \cdot x_f + c_k$, with coefficients streamed in order from top and left. Clenshaw requires maintaining two recurrence variables (d_{k-1} and d_{k-2}) and alternating between them each cycle, which would need either an extra register and multiplexer or a modified streaming schedule. Thus, we choose Horner for simplicity for now, and leave exploring Clenshaw to future work.

C. Attention Spill: Energy and Power Calculation

A WS-only array cannot fuse the softmax stage onto the MAC array, so it has to write the per-tile score and softmax matrices S and P ($N \times N$ each) to SRAM and read them back during the softmax stage. This costs $4N^2$ extra word accesses per tile (2 writes + 2 reads of N^2 elements each). Summed over the $(S/N)^2$ tiles of one layer per head, this is $4S^2$ extra word accesses, independent of N.

CACTI 7 (Balasubramonian et al., 2017), configured as a 32 KB single-port scratchpad at 90 nm (the closest CACTI-supported node to SkyWater 130 nm), reports a per-block (64-byte) read energy of 115.2 pJ and write energy of 299.9 pJ. Per 32-bit word these are $E_{\text{rd}} = 7.2$ pJ and $E_{\text{wr}} = 18.7$ pJ, so the avoided traffic energy is

$$E_{\text{spill}} = 2S^2(E_{\text{rd}} + E_{\text{wr}}) = 51.8\, S^2 \text{ pJ per layer per head.}$$

The StreamAttention runtime per layer is $T_{\text{layer}} = (S/N)^2 \cdot C_{\text{tile}} \cdot \tau_{\text{clk}}$, which at $N = d = 128$, $p = 3$, $\tau_{\text{clk}} = 100$ ns evaluates to $1.60\, S^2$ ns. Dividing E_{spill} by T_{layer} gives the asymptotic spill power $P_{\text{spill}} = 32.4$ mW, independent of S.

D. End-to-end Numerical Accuracy of the Chebyshev Pipeline

We measure the impact of the Chebyshev approximation on the full attention output using a bit-exact C++ model of the StreamAttention pipeline, validated against Verilator RTL simulation at small array sizes. The model reproduces every precision-limiting step of the hardware (FP16 truncation of MAC inputs, 13-bit fractional split, FP32→FP16 truncation between Horner steps) while executing at native speed, making the long-context sweep tractable. We sweep $S \in \{2048, 4096, 8192, 16384\}$ on a 128×128 array with $d = 128$, drawing Q, K, V from the heavy-tail distribution used in (Shah et al., 2024; Lin et al., 2025):

$$\mathcal{N}(0,1) + \mathcal{N}(0,100) \cdot \text{Bernoulli}(0.001).$$

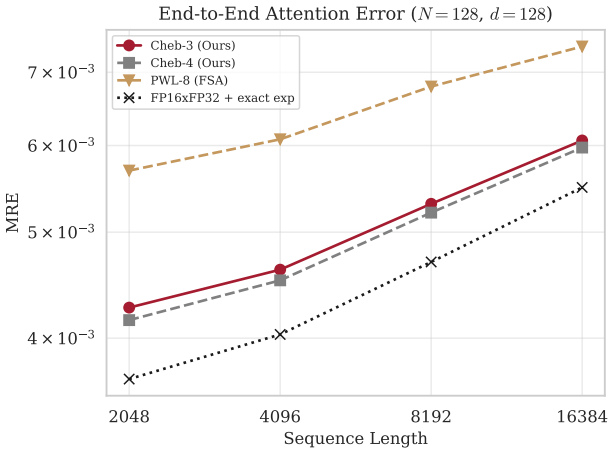


Figure 7. End-to-end attention MRE against an FP64 reference, measured with the bit-exact C++ hardware model. Chebyshev degree-3 sits within 10–17% of the FP16-exact-$\exp_2$ floor (dashed) at every sequence length, with error dominated by FP16 quantization rather than the polynomial approximation. Adding an additional Chebyshev degree yields little benefit.