

A Reduced-Scale RTL Implementation of a Runtime-Configurable Systolic Array for Transformer Acceleration

Eugene Lee

*Dept. of Electrical and Computer Engineering
University of Waterloo
Waterloo, Canada
e33lee@uwaterloo.ca*

Nicholas Rathfelder

*Dept. of Electrical and Computer Engineering
University of Waterloo
Waterloo, Canada
nhrathfe@uwaterloo.ca*

Akash Jayathirtha

*Dept. of Electrical and Computer Engineering
University of Waterloo
Waterloo, Canada
a2jayath@uwaterloo.ca*

Ryan Eggens

*Dept. of Electrical and Computer Engineering
University of Waterloo
Waterloo, Canada
reggens@uwaterloo.ca*

Simra Khan

*Dept. of Electrical and Computer Engineering
University of Waterloo
Waterloo, Canada
s77khan@uwaterloo.ca*

Abstract—Transformer inference is dominated by data movement, not arithmetic. We reproduce, at reduced register-transfer-level scale, the core ideas of a runtime-configurable systolic-array accelerator: processing elements that switch dataflow mode at runtime, integrated side units for the nonlinear operations, and operand reuse that keeps intermediates on chip. We implement a four-by-four array with four dataflow modes, a divider-free fixed-point softmax and layer-normalization vector unit, and an autonomous attention controller, all verified bit-exact against an independent software model. Keeping the query resident in the array removes a memory round trip and saves thirty cycles per attention operation, and a memory-traffic model projects the on-chip saving growing with sequence length. The fixed-point vector unit stays faithful to floating point, with a maximum softmax probability error of eight thousandths and a similarly small layer-normalization error, while eliminating the divider and square-root hardware.

Index Terms—systolic array, transformer accelerator, dataflow, softmax, layer normalization, fixed-point arithmetic, RTL, hardware verification

I. INTRODUCTION

Transformers underpin modern language, vision, and recommendation systems [2], but moving their intermediate matrices can cost more time and energy than computing the multiply-accumulates themselves. Attention is especially demanding: it creates the query, key, and value matrices, materializes a score matrix whose size grows quadratically with sequence length, and alternates matrix multiplication with nonlinear operations such as softmax and LayerNorm. A fixed-dataflow array must

repeatedly spill these intermediates or sit idle while a separate unit processes them.

DESA addresses this mismatch by changing the systolic dataflow at runtime, placing nonlinear side units next to the array, and fusing adjacent attention tiles so that reusable values remain on chip [1]. This project asks a more focused question than the full published accelerator: can those mechanisms be implemented and validated clearly at RTL scale? We therefore build a reduced four-by-four realization and compare every stage against an independent, bit-exact software model rather than attempting to reproduce DESA’s full chip.

The main contributions of this work are: (1) a configurable processing element and diagonally propagated mode wave supporting output-stationary, weight-stationary, quantize/output, and reload behavior; (2) integrated IPU, IOU, and fixed-point softmax paths, plus a standalone divider- and square-root-free LayerNorm VPU mode; (3) an autonomous controller for a reduced one-head attention workload and direct on-chip query reuse; and (4) an end-to-end verification and evaluation flow reporting cycle savings, numerical fidelity, memory traffic, FPGA resource use, and timing.

II. BACKGROUND

DESA combines three architectural ideas [1]. First, its processing elements adapt between stationary dataflows at runtime so each transformer matrix multiplication can use an appropriate mapping. Second, immediate processing, vector,

and output units are integrated at the array boundary to perform quantization, reductions, normalization, and transposition without a round trip through external memory. Third, tile fusion keeps intermediate attention values on chip and overlaps neighboring operations, reducing both buffer traffic and array bubbles.

Our implementation preserves these mechanisms but deliberately reduces their scale. The included scope is a parameterized RTL datapath demonstrated as a four-by-four INT8 array, the four runtime modes, adjacent side units, fixed-point softmax and LayerNorm, synchronous on-chip buffers, an autonomous single-head attention schedule, bit-exact verification, and FPGA implementation results. The workload uses four-by-four matrices and evaluates one- and two-tile query reuse. A production multi-head scheduler, DMA and external-memory interface, general large-model tiling, nonlinear-latency overlap, integrated encoder-block LayerNorm, and silicon power or energy measurements remain outside the project scope.

III. RTL IMPLEMENTATION DETAILS

Our implementation uses INT8 operands throughout the datapath, with 24-bit (INT8 $\times$ 3) signed accumulators inside each processing element to avoid overflow across a tile’s worth of multiply-accumulate operations. This precision choice is carried consistently through the array, side units, and vector unit described below. At the top level, `desa_top` composes the array with input, weight, temporary, and output buffers; the IPU, IOU, and VPU; and a controller that can execute either a generic programmed tile or the reduced attention schedule through a single start/done command. The controller coordinates buffer addresses, skewed operand streams, diagonal mode changes, result capture, softmax rows, and output requantization.

A. Configurable Processing Element and the Mode Wave

The Configurable Processing Element (CPE) sits at the heart of the main systolic array implemented for our reduced-scale reproduction of this architecture, enabling the three previously aforementioned features of the original design. The CPE features four different modes that enable the configurable systolic array behaviors required for fusing matrix multiplication tiles and localizing all non-linear computations.

The four CPE modes are as follows:

- **OS (Output Stationary):** The partial sum accumulates in place at the PE while operands stream through; used for standard matrix multiplication.
- **WS (Weight Stationary):** An operand is held locally while partial sums propagate downstream through the array.
- **QO (Quantize-and-Output):** The resident value is quantized and drained out through the IPU/IOU rather than continuing to accumulate.
- **RE (Reload):** A previously computed, quantized result is reloaded directly into the PE as the operand for the next operation, avoiding an explicit offload/setup cycle.

Our testing suite individually tests each of these PE modes, while our final workload focuses on performing OS-WS tile fusion, utilizing all four modes in the process.

To fully support tile fusion through our systolic array, our design required additional logic to ensure that each CPE transitions smoothly and efficiently into the computation of the next tile after the completion of the last. A naive broadcast implementation fails at this task due to the diagonal nature of result propagation in a systolic array. Following the insertion of the final round of operands into the array, results travel along a diagonal wavefront that shifts by one position every cycle. Without staggering the mode transition to match this wavefront, a PE downstream that has not yet finished consuming the previous tile’s operand would prematurely switch modes, either overwriting a partial sum still in flight or attempting to reload a result that has not yet arrived at that PE.

As such, our architecture features a 2D shift register implementing the “Mode Wave” that mimics this diagonal propagation pattern for our processing elements. The mode wave is seeded at PE(0, 0) on transition and shifts outward by one PE per cycle along both the row and column directions, reproducing the diagonal timing below without requiring global mode routing to every PE. By doing this, each processing element on the same diagonal shares the same time-multiplexed mode, allowing for the uninterrupted transition from the completion of one matrix multiplication (OS/WS) to the reloading and setup for the next (QO/RE). For a given row r and column c of the systolic array, PE(r, c) commits to the new mode at

$$t_{\text{switch}}(r, c) = r + c + 2, \quad (1)$$

guaranteeing that a PE only changes behavior once the last operand belonging to its previous mode has already been consumed. This mechanism is what allows Q^i to remain resident in the array immediately after generation and be reused as the WS operand for $Q^i K^{iT}$ without an intervening buffer round trip, as discussed further in Section VI.

B. Systolic Array, Buffers, and Control

The main compute fabric is a 4×4 grid of the CPEs described in Section IIIA, reduced in scale from DESA’s 64×64 array to keep the design tractable for RTL implementation and bit-exact verification within the scope of this project. Weight operands enter the array along the top edge and propagate downward through each column, while activation operands enter along the left edge and propagate rightward through each row, following the standard systolic interconnect pattern. Each CPE forwards its operand to its downstream neighbor on every cycle it does not consume it locally, so that data reaches PE(r, c) after r vertical and c horizontal hops.

The mode wave signal described in Section IIIA is wired independently of the operand datapath: it is driven directly from the 2D mode shift register to each CPE’s mode input, rather than being carried alongside weight or activation data. This separation keeps the diagonal timing of Equation 1 exact regardless of operand stalls elsewhere in the array.

Four instances of a common synchronous buffer module hold input activations, weights, temporary intermediates, and final outputs. Each buffer has independent read and write ports, one-cycle read latency, write-first behavior on an address collision, and an always-ready interface. The controller sequences configure/fill, load, compute, QO, store, fusion, softmax, and requantization states. For the autonomous workload it advances through K , V , Q , score, softmax, and output phases. In the DESA-style schedule, the Q phase enters the fusion state: the mode wave moves the array through QO and RE while K begins streaming, avoiding the separate query store and reload used by the matched naive schedule.

C. Side Units: IPU and IOU

The Immediate Processing Unit (IPU) and Immediate Output Unit (IOU) are responsible for all post processing performed on results as they leave the systolic array. These units retrieve results from the processing elements as they enter QO mode, with the IPU and IOU operating on the row and column sides of the array respectively.

The IPU performs three functions on each lane as results are captured. First, a per-tile bias or residual is added to every captured lane before any other processing. Second, the three tile statistics required by the VPU for non-linear processing – tile max, tile sum, and tile sum-of-squares – are computed via running reduction blocks over the bias-added lanes, updated incrementally so they are ready almost as soon as the last lane of a tile is captured. Separately, each bias-added lane is also independently re-quantized to INT8 by a scale-and-shift $S/2^b$, with saturation applied before truncation. The tile statistics are exposed to the VPU for non linear processing, while the quantized lane values can instead be routed directly back into the array as the operand for a subsequent RE-mode reload, allowing a result such as Q^i to be reused on-chip without a round trip through memory.

The IOU is a purely combinational drain path used for reading out already-computed values from the array’s PE accumulators. These raw accumulator values are read out one at a time as the controller sweeps a select index across the tile. A single transpose select line determines whether the unit drains in row-major order, matching the original matrix, or column-major order, matching the transposed matrix. This avoids the latency introduced by transposing through memory when a subsequent mapping requires it, most notably for K^T in the attention score multiplication. As per the original DESA specification, values leaving the IOU are left unquantized; when needed, they are routed through the IPU’s existing quantizer rather than being quantized locally.

D. Vector Unit: Divider-Free Fixed-Point Softmax

The Vector Processing Unit (VPU) computes the nonlinear attention operations that cannot be expressed as multiply-accumulate and therefore cannot run on the systolic array. The first is the softmax applied to each row of the score matrix $S = Q^i K^{iT}$. A direct evaluation of $\text{softmax}(x_j) = e^{x_j} / \sum_k e^{x_k}$ requires an exponential unit and a divider, and the divider is

by far the more expensive of the two in hardware. Following the DESA formulation [1], we remove it with the log-sum-exp identity,

$$\text{softmax}(x_j) = \exp\left(x_j - x_{\max} - \ln \sum_k e^{x_k - x_{\max}}\right), \quad (2)$$

which turns the division into a subtraction of the log-sum term and reuses the exponential unit a second time for the normalization. The row maximum $x_{\max}$ can be supplied by the IPU’s tile-max statistic (Section III); in the reduced top-level attention path it is recomputed combinatorially from the complete row held in the temporary buffer. Subtracting it keeps every exponent argument non-positive.

The datapath is entirely fixed point, using eight fractional bits in the exponential and probability domain. The exponential is evaluated as $2^{x \log_2 e}$, so the integer part of the exponent becomes a bit shift and the fractional part indexes a 64-entry table storing 2^f for $f \in [0, 1)$. The natural logarithm reuses a leading-one detector for its integer part and a second 64-entry table for the mantissa. Both tables are generated by the software model and emitted verbatim into the RTL, so the hardware and the golden model share identical constants. A short finite-state machine sequences the unit and emits one normalized INT8 row per invocation.

Raw scores are on the order of a few thousand, so the gaps between them span hundreds; the exponential then underflows every entry except the maximum and the softmax collapses into a hard $\arg \max$. We therefore apply a pre-softmax requantization, an arithmetic right shift, that is the integer realization of the conventional $1/\sqrt{d_k}$ scaling sized for the operand magnitudes. Sweeping the shift from 6 to 14 against a floating-point reference, a value of 11 gives non-degenerate distributions (mean maximum probability near 0.6) while keeping the upstream matmuls at full INT8 range; because $\max(s \gg k) = (\max s) \gg k$, shifting the lanes and the tile maximum by the same amount is exact. Fig. 2 compares the fixed-point attention weights with the true floating-point softmax: the maximum per-row probability error is 0.008 and the final attention output correlates with the floating-point result at 0.9997.

E. Vector Unit: Divider- and Square-Root-Free LayerNorm

Layer normalization is the second nonlinear operation handled by the VPU. For a length- m lane vector it computes

$$y_i = \gamma_i \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta_i, \quad \frac{1}{\sqrt{v}} = \exp\left(-\frac{1}{2} \ln v\right), \quad (3)$$

which introduces both a division and a square root. We avoid both by evaluating the reciprocal square root in the log domain, as shown on the right of (3), so the same logarithm table and exponential unit built for the softmax are reused and no dedicated divider or square-root circuit is required.

The mean and variance are taken from the tile-sum and tile-sum-of-squares statistics the IPU already tracks (Section III). For a power-of-two lane count they reduce to exact integer shifts: $\mu = \text{sum} \gg \log_2 m$, and, writing $m^2 \sigma^2 = m \cdot \text{sumsq} -$

sum², the fixed-point variance is a shift of that quantity. A constant ϵ (chosen as 1.0 in the variance domain) keeps the logarithm argument at least unity, so constant rows normalize to zero rather than dividing by zero. The reciprocal square root is evaluated at additional fractional precision by adding a constant offset to the exponential argument; because the softmax exponential only ever receives non-positive arguments whereas the LayerNorm reciprocal can require a positive one, the RTL adds a bidirectional-shift variant of the exponential function. Each lane is then centered, scaled by the reciprocal standard deviation, and passed through the per-lane affine parameters γ_i and β_i .

Fidelity against a floating-point LayerNorm reference, shown in Fig. 4, is a mean absolute error of 0.004 and a maximum of 0.031 over fifty thousand random rows, limited by the shared 64-entry tables rather than by the reciprocal-square-root formulation. Beyond the reference comparison, the implementation is checked against the defining properties of normalization: the pre-affine output has zero mean and variance $\sigma^2/(\sigma^2 + \epsilon)$, approaching one when the row variance dominates ϵ . Translation invariance is checked directly, while scale invariance is checked in that same non-degenerate regime. These properties hold independently of any reference implementation, and shift invariance is bit-exact in the fixed-point datapath.

IV. TESTING METHODOLOGY

Verification is built around an independent software golden model (`sw/desa_model.py`) that reproduces the exact integer behavior of the hardware: two’s-complement wrap-around, saturating quantization, truncation, and the fixed-point softmax and LayerNorm datapaths, including the shared lookup tables. Because the model is bit-accurate rather than tolerance-based, every comparison against the RTL design is an exact-equality check, so a mismatch is a genuine bug and not a rounding artifact. The model is derived independently from the specification, which means agreement between the two is meaningful evidence of correctness rather than a shared implementation detail.

The RTL is exercised with cocotb testbenches driven over Verilator. Each unit, namely the configurable PE, the buffers, the systolic array, the softmax VPU, the LayerNorm VPU, and the integrated top level, is checked element-for-element against the golden model across directed corner cases and randomized soak tests. For the vector unit a randomized soak confirms that the fixed-point softmax is bit-exact against the model and that its `argmax` always agrees with the true floating-point softmax, so the reference itself is validated rather than merely self-consistent. The LayerNorm mode is validated both by an element-for-element comparison of the device under test and by a suite of property tests that assert the mathematical properties of normalization (zero mean, the expected ϵ -adjusted variance, scale invariance away from the ϵ -dominated regime, and shift invariance) alongside a fidelity bound against an independent NumPy reference. In total, 34 cocotb tests exercise eight RTL targets: the configurable

PE, the buffers, the systolic array, the softmax vector unit, the LayerNorm vector unit, the integrated top level, and the single- and two-tile end-to-end attention flows. A further seven software property tests validate the LayerNorm golden model against the mathematical invariants of normalization.

Cycle counts are measured directly in the Verilator simulation through the controller’s start/done handshake and are deterministic across ten random seeds. The naive and DESA-style schedules are driven through the same controller and produce identical, golden-verified outputs, so any reported speedup reflects a difference in scheduling alone rather than in the computed result.

V. WORKLOADS AND SOFTWARE SIMULATOR

The primary workload is reduced single-head self-attention. Its input activations X and weight matrices W_Q , W_K , and W_V are 4×4 INT8 matrices; the systolic array accumulates in 24 bits, and the sequence length, head dimension, and array dimensions are all four. This baseline uses one reduction tile with no cross-tile accumulation. A separate two-tile evaluation extends the reduction depth to eight (X is 4×8) specifically to test query reuse across multiple reduction tiles. The baseline flow is $Q, K, V = XW$, then the scores $S = Q_q K_q^T$, a softmax producing S' , and the attention output $S'V_q$. Because the array multiplies INT8 by INT8, the 24-bit products of the projection matmuls are requantized to INT8 before the score and output matmuls; the reload and output shifts were swept against the floating-point reference (`reload_shift=8, output_shift=7`), as was the pre-softmax `score_shift=11`.

Alongside the RTL we developed a software simulator in the same repository. Its core is the bit-exact model described in Section IV, which additionally provides a cycle-accurate model of the configurable PE, an integer matmul with the accumulator’s wrap-around semantics, and the fixed-point softmax and LayerNorm. These components are composed end to end in `sw/attention_pipeline.py`, which runs the complete attention flow and returns every intermediate stage together with a floating-point reference for the softmax weights and the final output. The simulator serves three roles: it is the golden reference for RTL verification, it is the vehicle for the numerical studies (the requantization and score-shift sweeps and the softmax and LayerNorm fidelity measurements), and it generates reusable test vectors consumed by the end-to-end RTL tests.

VI. RESULTS

A. Software Simulator vs. RTL Simulation

At every stage of the workload the RTL design is bit-exact against the software simulator: the projection and score matmuls, the fixed-point softmax, the LayerNorm mode, and the integrated end-to-end attention operation all match the model element-for-element. This equivalence is what licenses using the software simulator for numerical studies that would

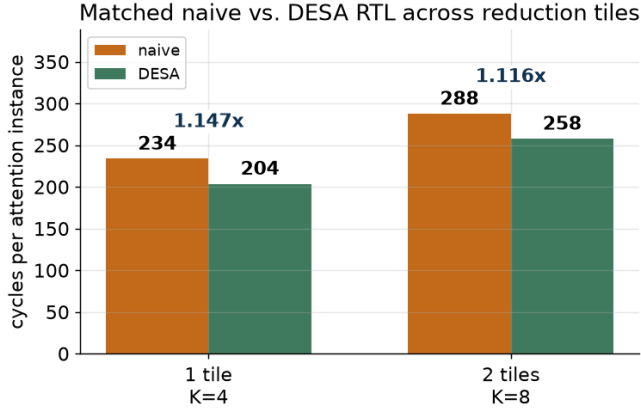


Fig. 1. Naive vs. DESA-style Q reuse. One tile: 234→204 cycles (1.147×); two tiles: 288→258 cycles (1.116×).

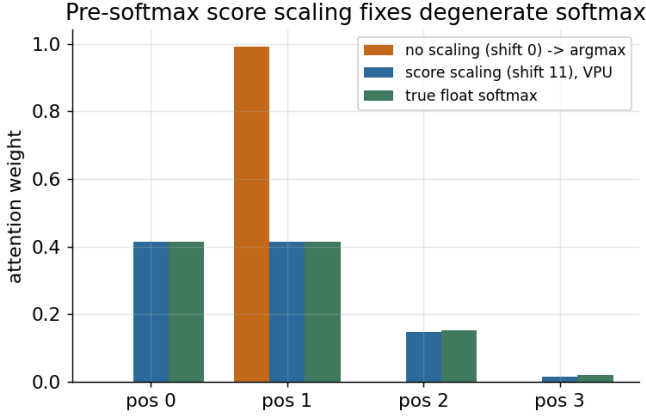


Fig. 2. Fixed-point softmax fidelity vs. floating point.

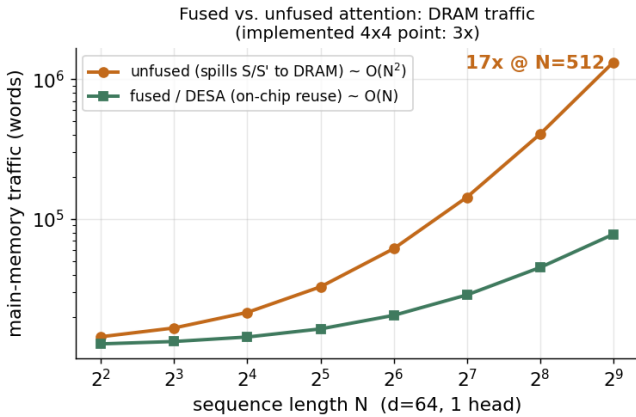


Fig. 3. Modelled DRAM traffic, fused vs. unfused, vs. sequence length.

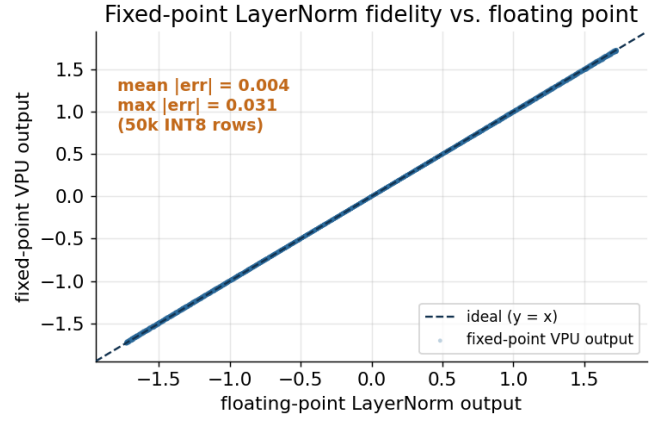


Fig. 4. Fixed-point LayerNorm output versus a floating-point reference over 50 000 random INT8 rows (identity affine). The divider- and square-root-free datapath tracks the reference to a mean absolute error of 0.004 and a maximum of 0.031.

be impractical to run entirely in RTL, including the 50,000-row LayerNorm fidelity sweep, attention-level softmax comparison, and requantization parameter searches.

B. Obtained vs. Published Results

Table I separates directly measured RTL results from published full-scale results and analytical projections. Our 1.147× figure measures only the cycle benefit of retaining Q across one fused boundary; it is not an attempt to reproduce DESA’s complete 11.6–15× system-level speedup. Likewise, the 17× memory-traffic reduction at sequence length 512 is produced by our traffic model, not by an attached DRAM controller. The four-by-four implementation instead provides direct evidence for mechanism-level correctness, fixed-point accuracy, FPGA cost, and timing, while the scale gap explains why fill, drain, and serialized nonlinear work occupy a much larger fraction of our execution.

TABLE I
OBTAINED VS. PUBLISHED (DESA) RESULTS

Metric	This work (4×4)	DESA [1] (64×64)
Array / precision	4×4, int8/int24	64×64, INT8/INT28
Q-reuse command speedup	1.147× (1 tile)	—
System speedup vs. SOTA	not measured	11.6–15×
Softmax max prob err	0.008	—
LayerNorm mean/max err	0.004 / 0.031	—
Memory traffic (N=512)	17× (modelled)	—
PE utilization	36.4% (K=4)	trend

C. Resource Utilization and Timing

Table II summarizes post-implementation resource utilization for the reduced-scale design on a Xilinx Spartan-7 device (xc7s25csga324-1). The design uses no on-chip Block RAM, since all buffering at this scale fits in distributed memory and registers.

TABLE II
RESOURCE UTILIZATION (xc7s25csga324-1)

Resource	Used	Available	Util.%
LUTs	8,874	14,600	60.78%
Registers	2,247	29,200	7.70%
DSP48E1	10	80	12.50%
Block RAM	0	45	0.00%

Under a clock-only constraint, post-route internal register-to-register timing closes at a period of 17.937 ns, corresponding to 55.75 MHz, with a worst negative slack (WNS) of +0.019 ns. This is an internal core-frequency result, not a board-level system limit: input and output delays were not constrained. The critical path runs from a mode-wave shift register through 26 logic levels, ten of them chained carry blocks, into the IPU’s running tile-maximum reduction register. This is consistent with the IPU’s reduction structure described in Section III: because tile max, sum, and sum-of-squares are updated incrementally every cycle rather than computed as a single-cycle tree reduction, the comparison and carry logic for the running maximum forms a long combinational chain that dominates the critical path at this array size and sets the achievable clock frequency. Register utilization remains low relative to LUTs, reflecting that the design is combinational bound rather than register-bound – consistent with a systolic array whose per-PE logic is dominated by the multiply-accumulate and mode-select datapath rather than deep pipelining.

D. Discussion

The measured $1.147\times$ matched-command speedup from direct query reuse is far smaller than the $11.6\text{--}15\times$ system-level speedup over prior accelerators reported by DESA [1], but the two values have different baselines and scope. Our result isolates and validates a single mechanism cleanly and at reduced scale, whereas the published figure combines full attention fusion, tile overlap, and nonlinear-latency hiding on a much larger array. Two of our own measurements predict the difference: the memory-traffic model (Fig. 3) shows the on-chip saving growing from roughly $3\times$ at our size to a modelled $17\times$ at a sequence length of 512, and the per-matmul utilization rises from 36.4% at a reduction depth of four toward its asymptote as the reduction depth grows. On the numerical side, the fixed-point vector unit stays faithful to floating point, with a maximum softmax probability error of 0.008 and a mean and maximum LayerNorm error of 0.004 and 0.031; in both cases the residual error is limited by the shared 64-entry lookup tables, which trade a small, quantified accuracy loss for the elimination of the divider and square-root hardware.

VII. CONCLUSION AND FUTURE WORK

We implemented the central DESA mechanisms at reduced RTL scale: a runtime- configurable systolic array, diagonal mode transitions, on-chip query reuse, integrated side units,

an integrated fixed-point softmax path, a standalone Layer-Norm mode, synchronous buffers, and an autonomous one-head attention controller. Thirty-four cocotb tests and seven software property tests establish bit-exact agreement with the golden model. The matched schedules show that direct query reuse saves 30 cycles for a single tile, while the numerical studies quantify the small error introduced by the shared lookup-table datapath. Implementation on a Spartan-7 further grounds the design at 8,874 LUTs, 2,247 registers, 10 DSPs, and 55.75 MHz.

The next step is to overlap softmax, drain, and neighboring tiles rather than serializing them, and to measure traffic with RTL counters connected to a realistic external-memory interface. General sequence lengths and tile shapes, multi-head scheduling, and DMA support would turn the demonstrator into a more complete accelerator. Integrating residual addition and the existing LayerNorm mode into `desa_top` would extend the workload from attention to an encoder block. Finally, larger lookup tables or interpolation could explore the accuracy/area tradeoff, while post-route power analysis or board measurements would enable an energy comparison with the published design.

Repository: <https://git.uwaterloo.ca/ece493-s26/projects/team-24/>

REFERENCES

- [1] Z. Wang, H. Fan, and G. He, “DESA: Dataflow efficient systolic array for acceleration of Transformers,” *IEEE Transactions on Computers*, vol. 74, no. 6, pp. 2058–2072, June 2025, doi: 10.1109/TC.2025.3549621.
- [2] A. Vaswani *et al.*, “Attention is all you need,” *Advances in Neural Information Processing Systems*, 2017.