Systolic Arrays for Efficient Attention Computation

Ulrik Unneberg, Olav Førland, Denny Cao, Lavik Jain
John A. Paulson School of Engineering and Applied Sciences
Harvard University
Cambridge, United States
{ulrikunneberg, olavfoerland}@g.harvard.edu
{dennycao, lavikjain}@college.harvard.edu

: Systolic Arrays for Efficient Attention Computation

***Abstract*—This paper proposes a novel systolic array architecture to enhance the efficiency of attention computation in Transformer models. Traditional attention mechanisms impose high computational costs due to intensive matrix multiplications and non-linear softmax operations. Our design pipelines these computations using systolic arrays and introduces an intermediate systolic array to approximate the softmax function via Taylor series expansion. We implement and verify the design in System Verilog, and analyze the error from the Taylor approximation. Comparisons with similar work highlight the potential of our design to speed up attention computation in hardware accelerators.**

I. Introduction

The Transformer architecture, introduced in 2017 [7], has become a cornerstone of modern machine learning, excelling in tasks across natural language processing (NLP), computer vision, and speech recognition[1][6][2]. It outperforms traditional architectures like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) by addressing limitations in sequential data processing and long-range dependency modeling. The key innovation in Transformers is the attention mechanism, which enables models to capture dependencies between input elements by evaluating pairwise interactions and assigning importance weights. Attention is defined as

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V,$$

where $Q, K,$ and $V \in \mathbb{R}^{N \times d_k}$ represent query, key, and value matrices, with sequence length N and embedding dimensionality d_k [7].

Despite its effectiveness, the attention mechanism imposes high computational costs due to its reliance on matrix multiplications and the sandwiched non-linear softmax operations. These computations demand substantial data movement and processing overhead, limiting throughput on conventional hardware like CPUs and GPUs [8].

The goal of this project is to explore whether hardware based on the systolic array architecture can enhance the efficiency of attention computation. Systolic arrays are specialized hardware architectures that achieve high throughput and energy efficiency in matrix operations by coordinating a network of processing elements (PEs) that operate in parallel with minimal I/O overhead [4, 3]. Data flows rhythmically through the array, allowing operations to overlap and reducing the need for frequent memory access.

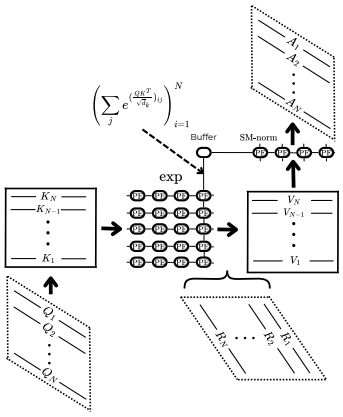

Fig. 1. Overview of the systolic architecture for computing attention. Queries Q and keys K are first computed using a weight-stationary systolic array. Then, each computed element flows into an intermediate systolic array that exponentiates each element using Taylor approximation. The resulting scores R are then pipelined into another weight-stationary systolic array that holds values V, simultaneously as the row-wise normalization term in the softmax is computed by accumulating the terms upwards in the diagram. As the results of RV become available, the normalization is applied in a final systolic array.

We propose a novel systolic architecture for attention computation, shown in Figure 1. The two matrix multiplications are pipelined using systolic arrays, with an intermediate systolic array estimating the softmax operation using a Taylor approximation. The architecture proposed (i) avoids materialization of intermediate results as each result is used immediately as it becomes available, and (ii) effectively hides the computation time of the normalizing term in the softmax.

II. Approach

In this section, we present our proposed approach for improving the performance of attention computation. We start by describing the architecture and the dataflow of the systolic arrays performing the attention computation, before detailing the contributions of our work.

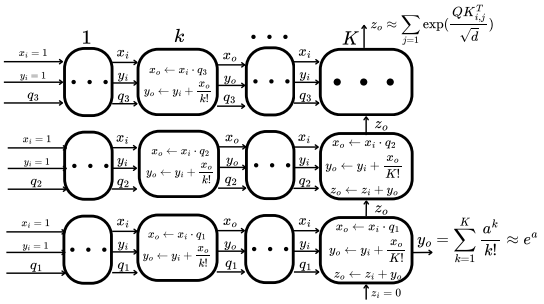

Fig. 2. Overview of the delayed softmax systolic array, where we exponentiate each element in the matrix $QK^\top$, and collect the cumulative row sum upwards in the final column.

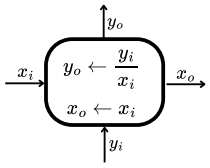

Fig. 3. Scaling processing element used for final normalization of the attention matrix. The elements of RV comes in from the bottom, while the normalization term flows in from the left. The normalization term is then propagated to the processing element to the right, while the final attention score is sent to the output buffer.

A. Architecture & Dataflow

Figure 1 shows our proposed systolic architecture for computing attention. Keys K and values V are first loaded into their respective weight stationary systolic arrays. Then, we compute the attention score matrix QK^T by skewing the query matrix Q and feeding it into this array in a traditional weight-stationary systolic manner [4]. The resulting elements are pipelined into a special-purpose systolic array that performs element-wise exponentiation using Taylor series approximation, shown in Figure 2. We define the resulting matrix R as

$$R_{i,j} = \exp((QK^\top)_{i,j}/\sqrt{d_k}), \quad \forall \;\; i,j \in \{1,\dots N\}$$

The final column of the exponentiation array does two things. (i) It pipelines the resulting exponentiated elements into the systolic array holding the elements of the V matrix, and (ii) it accumulates the normalizing terms of the softmax function upwards.

The application of the normalizing terms are delayed until the results of RV become available. To perform the normalization we introduce a 1-dimensional systolic array where each processing element takes as input an element $(RV)_{i,j}$ from below and the corresponding normalization term from the left, applies the normalization, and sends the resulting attention entry to the output buffer while pushing the normalization term to the next processing element. Figure 3 shows the structure of these processing elements.

B. Contributions

This approach is compelling as the full attention computation is fused together without the need to store any intermediate results. The architecture utilizes parallelism to a great extent, using systolic arrays for both matrix multiplications in addition to the systolic exponentiation array. Finally, delaying the softmax normalization ensures this sum can be computed while the final matrix multiplication takes place. This effectively hides the complexity of the softmax computation. The foundation of our work is laid by Kung on systolic arrays [4]. Similar approaches have been done by Ye et al. (2023) [10] and Wang et al. (2023) [8], which we will elaborate on in V.

III. Implementation

We verified and implemented the design using the Hardware Description language System Verilog. The implementation was simulated using Verilator and C++. We conducted two experiments:

(i) *Correctness of implementation*: We tested the implementation in System Verilog against a correct attention computation implemented in C++. This was done to verify that we obtained the correct results in the expected number of clock cycles.

(ii) *Error analysis*: We ran several simulations varying the scale of the input matrices and the number of terms used in the Taylor approximation. These will be discussed in IV.

Both when verifying correctness and when analyzing error from the Taylor approximation, the entries of Q, K, and V matrices where drawn i.i.d. from the uniform distribution:

$$Q_{ij}, K_{ij}, V_{ij} \sim \mathcal{U}(0, \overline{\mathrm{U}})$$

where $\overline{\mathrm{U}} \in \{0.01, 0.1, 1.0, 10.0\}$ are the maximum values we used. Throughout the implementation we use a fixed systolic size of 16×16. All code is available in this Github Repository.

IV. Results & Discussion

Without the resources or time to access FPGA and obtain empirical results, we instead present a theoretical analysis of the proposed systolic architecture. We present the clock cycle count needed to compute the attention mechanism and error introduced by the Taylor approximation. We use a fixed systolic array size of 16×16, and leave it to future work to implement tiling techniques to make this work for arbitrary matrix sizes.

A. Clock Cycle Count

For the proposed systolic architecture, the clock cycle count to compute the attention mechanism can be broken down into four main parts: (1) computing the matrix product $QK^\top$, (2) exponentiating the elements to obtain R, (3) multiplying R by V, and (4) normalizing RV to obtain the final attention scores.

In isolation, (1) uses $2N + d_k$ cycles, (2) uses T cycles, (3) uses $2N + d_k$ cycles, and (4) uses 1 clock cycle. Performing these operations without pipelining leads to $4N + 2d_k + T + 1$ clock cycles. As our design avoids materializing intermediate

results, we finish in $2N + 2d_k + T + 1$ clock cycles, a clear improvement. Additionally, the scheme presented hides the $\mathcal{O}(N)$ complexity of the Softmax computation, replacing it with $\mathcal{O}(T)$, where T is the number of terms in the Taylor approximation, a significant improvement as $T \ll N$ for not too large inputs.

B. Error Analysis

The Taylor approximation leads to an error in the computed attention matrix. From Taylor's theorem, the error will depend on the size of the input and the number of terms in the Taylor approximation [9]:

$$R_T(x) = \left| e^x - \sum_{t=0}^{T} \frac{x^t}{t!} \right| \leq \frac{\max_{y \in (0,x)} |e^y|}{(T+1)!} |x|^{T+1}.$$

When running the error analysis described in III the results are in accordance with the theory. Figure 4 shows how the final attention error is reduced by additional terms, for four different input scales for Q, K, and V matrices. We can see that as the input size reaches a certain threshold, adding more terms provides minimal benefit in reducing error. Therefore, focusing on limiting the size of the input should be a priority for future work. A more detailed analysis is performed in Appendix A.

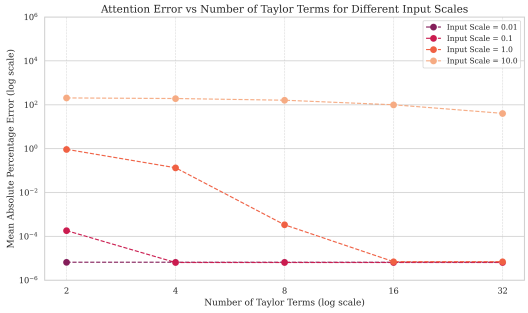

Fig. 4. Attention error (in terms of mean absolute percentage error) as a function of the number of terms in the Taylor approximation for different scales of the Q, K, and V matrices.

An interesting strategy to explore is to model $e^x = 2^i e^y = e^{i \log 2} e^y$. See Appendix B for more details.

Another point of consideration is the potential large values occuring from the exponentiation and delaying the normalization. This might consume large memory, and is something that should be investigated experimentally for further research.

V. Related Works

As our results are simulation-based and work as a proof of concept, it is hard to provide direct comparisons with state-of-the-art. However, we want to highlight some similar implementations that have come further in the development phase than ours, and have proved to work well.

Ye et al. (2023) [10] present a hardware accelerator for multi-head attention on FPGAs using a reconfigurable systolic array architecture. The proposed design introduces dynamic reconfiguration to adapt to different bit-width inputs and implement a radix-2 softmax to replace the exponential computation in traditional softmax. Their architecture improves latency between $24.5\times$ and $51.3\times$ compared to CPU and between $3.3\times$ and $17.3\times$ compared to GPU.

Wang et al. (2023) [8] proposes Co-Operative Systolic Arrays (COSA) that are dynamically configured into weight- and output-stationary systolic arrays to compute attention. They implement it for multi-head attention, and use fusion methodologies to hide the softmax computation. They report a $2.95 - 28.82\times$ speedup compared to SOTA.

VI. Conclusion

This research introduces a novel application of systolic arrays for attention computation in Transformer models. Our contributions include a systolic architecture that pipelines the computation of attention and allows delaying softmax normalization for parallelization benefits.

As we all had little to no prior experience within hardware design, we have learned a lot about the theoretical and practical challenges in designing these systems. We find this a valuable experience that one rarely get exposed to on the high level.

While we successfully implemented the forward pass of attention computation in HDL Verilog and validated it experimentally, the lack of empirical testing on FPGA hardware limits its practical applications at this point. This is something we would be very interested in exploring in the future.

A. Future Work

Future research can expand on our findings in several directions. The first and most important one is, as mentioned, to actually implement and test the method on an FPGA, to experience where the practical bottlenecks appears, and obtain experimental performance metrics. Another natural extension is to extend the design to include the matrix multiplications with the weight matrices W^Q, W^K, W^V, as described in [7], and possibly parallelize / reuse computations across more attention heads in the multi-head attention. Another easy extension is to explore the aforementioned binary exponentiation methods is also something that could enhance the enhance the accuracy for larger inputs, in addition to being just as fast.

Simpler extensions could be to explore with different number representation, for instance fixed-points. This loses some accuracy but is more efficient on edge devices. There naturally also comes new challenges when working with huge input matrices, so efficient tiling strategies here could be of importance. Finally, we need to test the method end-to-end in for instance a transformer, train the model to test both backward and forward pass in action, and validate how well the model performs.

The architecture proposed also fits very well with grouped-query attention, as the key and value matrices only needs to be read once per group. Then, each query in the group can be pipelined into the architecture shown in Figure 1 as a long Q matrix.

Group Contribution Statement

Olav and Ulrik designed the architecture and implemented it in System Verilog. Olav and Ulrik also created the diagrams (Figure 1, Figure 2, Figure 3). Lavik conducted the initial literature review and comparison of results with the state of the art. Lavik also wrote the initial draft of the report, which the other team members proofread and revised. Denny conducted a theoretical analysis of the results in comparison with the state of the art and worked on writing the introduction and literature review.

Acknowledgment

The repository structure was influenced by the work from the repository 2D-Systolic-Array-Multiplier. His System Verilog code for output-stationary systolic arrays was a great starting point when we had no background in any Hardware Description Language. Also, we adapted his testbench in C++. Thank you!

We would also like to thank Professor H.T. Kung and the CS2420 teaching staff for providing us guidance and challenging our design throughout this project. Thank you!

References

[1] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *arXiv preprint arXiv:1810.04805* (2019). URL: https://arxiv.org/abs/1810.04805.

[2] Alexey Dosovitskiy et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. ICLR. 2021. URL: https://github.com/google-research/vision_transformer.

[3] Norman P Jouppi et al. "In-datacenter performance analysis of a tensor processing unit". In: *Proceedings of the 44th annual international symposium on computer architecture*. 2017, pp. 1–12.

[4] H. T. Kung. "Why Systolic Architectures?" In: *Computer* 15.1 (1982), pp. 37–46. DOI: 10.1109/MC.1982.1653825.

[5] A. Cristiano I. Malossi et al. "Fast Exponential Computation on SIMD Architectures". In: *Workshop on Applications of Petri Nets and Other Concurrency Models (WAPCO)*. Accessed: 2024-12-11. IBM Research - Zurich, Switzerland. 2015. URL: https://wapco.e-ce.uth.gr/2015/papers/SESSION3/WAPCO_3_5.pdf.

[6] Alec Radford et al. "Improving Language Understanding by Generative Pre-Training". In: *OpenAI preprint* (2018). URL: https://www.openai.com/research/language-unsupervised.

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

[8] Zhican Wang et al. "COSA: Co-Operative Systolic Arrays for Multi-head Attention Mechanism in Neural Network using Hybrid Data Reuse and Fusion Methodologies". In: *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE. 2023, pp. 1–6.

[9] Wikipedia contributors. *Taylor's theorem — Wikipedia, The Free Encyclopedia*. [Online; accessed 2024]. 2024. URL: https://en.wikipedia.org/wiki/Taylor%27s_theorem.

[10] Wenhua Ye et al. "Accelerating Attention Mechanism on FPGAs based on Efficient Reconfigurable Systolic Array". In: *ACM Transactions on Embedded Computing Systems* 22.6 (Nov. 2023), 93:1–93:22. DOI: 10.1145/3549937.

Appendix

A. Error Analysis

With additional time, we would strive to obtain an error bound for the model. We first perform a theoretical analysis of the error introduced by the Taylor approximation and how it affects the attention terms. We apply Lagrange Error Bound to the Taylor approximation of the exponential function of order T to get an upper bound on the error for each entry in the intermediate matrix R:

$$|\varepsilon(x)| = \left| e^x - \sum_{t=0}^{T} \frac{x^t}{t!} \right| \leq \frac{e^c}{(T+1)!}|x|^{T+1} \tag{1}$$

where e^c is the maximum value of the exponential function on the interval $[0, x]$. For readability, let $D(x)$ be the Taylor approximation of the exponential function with T terms, $A = \sum_{j=1}^{N} D(z_j)$, and $B = \sum_{j=1}^{N} \varepsilon(z_j)$. We can then express the error in the softmax computation as:

$$\Delta S_i = \left| \frac{D(z_i) + \varepsilon(z_i)}{A+B} - \frac{D(z_i)}{A} \right| = \left| \frac{A \cdot \varepsilon(z_i) - B \cdot D(z_i)}{A(A+B)} \right|$$

By the triangle inequality, $|a-b| \leq |a| + |b|$.Thus:

$$\leq \frac{A \cdot |\varepsilon(z_i)| + |B| \cdot D(z_i)}{|A(A+B)|}$$

For the numerator, we upper bound $A \leq Ne^c$, $|B| \leq N \cdot \frac{e^c c^{T+1}}{(T+1)!}$, and $D(z_i) \leq \frac{e^c |z_i|^{T+1}}{(T+1)!}$ and substitute the Lagrange error bound from Equation 1 for $|\varepsilon(z_i)|$. For the denominator, we lower bound $A \geq Ne^{-c}$ since, with high probability, $|z_i| \leq c$ by construction and lower bound $B \geq -N \cdot \frac{e^c c^{T+1}}{(T+1)!}$:

$$\begin{aligned}
&\leq \frac{Ne^c \cdot \frac{e^c |z_i|^{T+1}}{(T+1)!} + N \cdot \frac{e^c |z_i|^{T+1}}{(T+1)!} \cdot \frac{e^c |z_i|^{T+1}}{(T+1)!}}{Ne^{-c}(Ne^{-c} - N \cdot \frac{e^c |z_i|^{T+1}}{(T+1)!})} \\
&\leq \frac{Ne^{2c}|z_i|^{T+1}\left(\frac{1}{(T+1)!} + \frac{|z_i|^{T+1}}{(T+1)!^2}\right)}{N^2 e^{-2c} - N^2 \cdot \frac{|z_i|^{T+1}}{(T+1)!}} \\
\Delta S_i &= \mathcal{O}\left(\frac{(Ne^{2c}|z_i|^{T+1} \cdot \frac{1}{(T+1)!}}{N^2 e^{-2c}}\right) = \mathcal{O}\left(\frac{e^{4c}|z_i|^{T+1}}{N(T+1)!}\right) \\
&= \mathcal{O}\left(\frac{e^c}{(T+1)!}|z_i|^{T+1}\right)
\end{aligned} \tag{2}$$

We note that this is on the same order as the error of the Taylor Approximation. We run an experiment to evaluate the distribution of $\frac{QK^\top}{\sqrt{d_k}}$ entries by randomly generating token

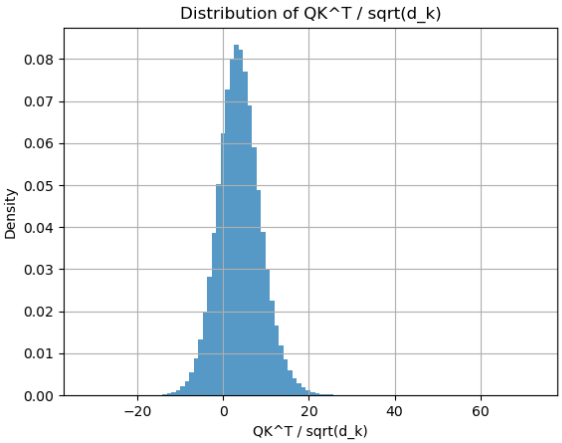

Fig. 5. Distribution of $\frac{QK^\top}{\sqrt{d_k}}$ values using GPT-2 with single head

IDs to create embeddings to produce Q, K using pretrained weights. From there, we plot the distribution of values. For instance, using GPT-2, we obtain the following distribution in Figure 5 which is approximately normal with mean 3.8648 and variance 25.879. We can effectively obtain a value c by getting the 99.7% confidence interval with $P(|X-\mu| < 3\sigma) = 0.997$.

B. Exponentiation

As mentioned, an interesting strategy to explore is to model $e^x = 2^i e^y = e^{i \log 2} e^y$. Let then $i = \lfloor x / \log 2 \rfloor$ be the integer part of the division. Here we only divide by the constant $\log 2$ and omit the fractional bits to obtain the floor function. 2^i is easily computed using a shift operation. $y = x - i \log 2$ is the fractional part. We then know that e^y is a small number that is approximated well using the Taylor approximation, and $e^x = 2^i e^y$ [5].