Efficient Implementation of Transformer Inference via a Tiled-Based Architecture on an FPGA*

1st Ling-Chi Yang
Institute of Electronics
National Yang Ming Chiao Tung University
HsinChu, Taiwan
hisky1256.ee11@nycu.edu.tw

2nd Chi-Jui Chen
Graduate Degree Program of College of Electrical and Computer Engineering
National Yang Ming Chiao Tung University
HsinChu, Taiwan
silencekugel.ee11@nycu.edu.tw

3rd Trung Le
Electrical and Computer Engineering
University of Washington
Washington, USA
tle45@uw.edu

4th Bo-Cheng Lai
Institute of Electronics
National Yang Ming Chiao Tung University
HsinChu, Taiwan
bclai@nycu.edu.tw

4th Scott Hauck
Electrical and Computer Engineering
University of Washington
Washington, USA
hauck@uw.edu

4th Shih-Chieh Hsu
Physics
University of Washington
Washington, USA
schsu@uw.edu

***Abstract*—This paper presents an ultra-low-latency implementation of a machine learning inference algorithm called a "Transformer". In this research, we utilized the FlashAttention-2 algorithm on an FPGA, which is a device that has greater on-chip memory resources compared to a GPU. This involved transitioning from row-wise to tile-wise data accesses and using smaller tiles to create a more fine-grained pipeline. To address the challenge of low efficiency on dataflow architecture due to limited memory ports and data conflicts, we implemented a set of ping-pong buffers that allow interleaved access without stalling the computation of the attention mechanism. Our proposed Dataflow architecture demonstrates a significant increase in power efficiency, achieving improvements of 61% to 321% over existing FPGA-based transformer accelerators.**

***Index Terms*—FPGA, transformer, machine learning, inference, flash attention, energy-efficient, dataflow**

I. Introduction

In recent years, due to significant data movement and low arithmetic intensity, general-purpose platforms such as GPUs and CPUs have encountered challenges in efficiently executing Transformer inference. Consequently, domain-specific accelerators implemented on FPGA or ASIC platforms, which have energy constraints, have been developed to enhance Transformer performance.

To address these challenges, the algorithms FlashAttention-2 [1] change the access from row-wise to tile-wise in the softmax without any accuracy drop, reducing bandwidth and the lifetime of attention weights, thus decreasing on-chip memory usage.

Despite the significant reduction in bandwidth and space in FlashAttention-2 when implemented on GPUs, the following issues still exist:

- The delay caused by High-Bandwidth Memory (HBM) transfers is still much greater than the time taken for K and V matrix multiplications. FPGAs, which have more on-chip memory, may not require data transfers to HBM.
- Even though the number of non-linear operations is reduced, their computational time cannot be ignored. FPGAs offer more flexibility with bit-level operations, allowing fast lookup tables to bypass this computation time for fixed-point operations.
- Although the FlashAttention-2 algorithm on GPUs provides adjustability in parallelism along the sequence length direction, it does not offer block partitioning in the head (hidden) dimension to fit cache sizes. We have utilized the reconfigurability of Block RAM on FPGAs and fine-tuned the Flash Attention algorithm to provide better memory utilization.

However, even though FPGAs can better utilize the characteristics of FlashAttention-2 to significantly reduce the memory space needed for attention calculations to be independent of N (though other variables still require O(N) storage), there are still many challenges to be addressed:

- Each tiled variable is dependent, which prevents Xilinx Vivado from using dataflow optimization to further reduce latency, resulting in low DSP and BlockRAM usage efficiency.
- Most of the FPGA optimizing tools currently use fixed-

point data types. However, for Transformers, each layer's output often contains a few very large values in unpredictable channels within the same token, causing quantization to require precision sacrifices to avoid numerical overflow.
- Due to the limited Bonded IO (IOB) on FPGAs, it is not possible to transmit all channel data at once, necessitating data reordering to implement tile-based streaming transmission to reduce kernel IO transfer rates.

Based on the above analysis and optimizations, we have developed a more efficient Transformer on FPGAs. The main contributions are as follows:

- We propose using a ping-pong buffer to interleave access to attention tiles to eliminate this dependence, enabling higher granularity of pipelining between different tiles and further improving hardware utilization.
- We used post-training quantization with a small amount of calibration data to determine the precision of each layer and variable to achieve high quality mixed precision inference.
- We provide all directions of tiling factors for users to adjust the transfer rate and computation rate between layers according to specific hardware specifications or latency requirements.

II. Attention algorithm

The formulas for standard attention can be referenced in equations 1, 2, and 3. As one can see, softmax is accessed row-wise, which requires creating a buffer of at least the sequence length to store the attention. By using the formula for flash attention mentioned in [2], we can apply correction factors by calculating the exponential of the differences between two local maximums and local sums (such as alpha and beta in the equations) to split the softmax processing, shown in equations 4, 5 and 6. This approach reduces the bandwidth and buffer size needed for attention storage, while also shortening the lifetime of the attention variables to allow concurrent computation of the matrix multiplications Q_iK^T and P_iV.

$$S_i = \frac{Q_iK^T}{\sqrt{N}} \tag{1}$$

$$P_i = Soft(S_i) \tag{2}$$

$$O_i = P_iV \tag{3}$$

$$S_i = [S_{(i,0)}, S_{(i,1)}] \in \mathbb{R}^{1\times T} \tag{4}$$

$$P_i = [\alpha \cdot Soft(S_{i,0}), \quad \beta \cdot Soft(S_{i,1})] \in \mathbb{R}^{1\times T} \tag{5}$$

$$O_i = [\alpha \cdot Soft(S_{i,0})V, \quad \beta \cdot Soft(S_{i,1})V] \in \mathbb{R}^{1\times T} \tag{6}$$

III. Tile-based architecture

A. Layer Normalization

$$\text{LayerNorm}(x) = \frac{x-\mu}{\sqrt{\sigma^2+\epsilon}} \cdot \gamma + \beta \tag{7}$$

The formula 7 for Layer Normalization uses an inverse of the square root table to obtain the reciprocal of the standard deviation. Additionally, since the data needs to be accessed again after calculating the variance, the row buffer in Figure 1 is used to temporarily store a row of data, avoiding the need to access the entire matrix. Also, since we use fixed-point arithmetic, we can disregard the calculation of epsilon.

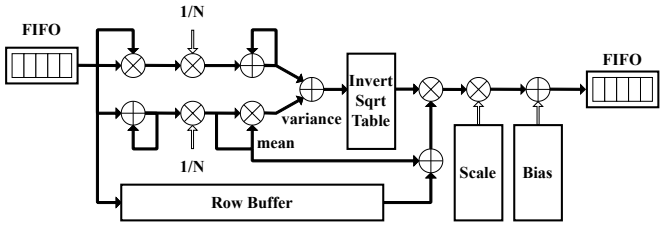

Fig. 1: Architecture of Layer Normalization

B. Multi-Head Self-Attention (MHSA)

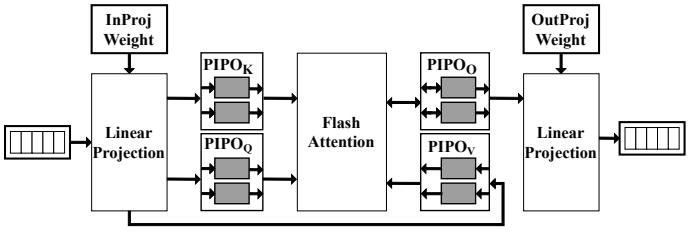

Fig. 2: Architecture of Multi-Head Self-Attention

The overall architecture uses Output Stationary Matrix Multiplication for Input Linear Projection, significantly shortening the lifetime of input data and eliminating the need for an input buffer. Conversely, Output Linear Projection employs Input Stationary Matrix Multiplication, which greatly reduces output data transmission latency and eliminates the need for an output buffer. Additionally, to ensure different batches can run simultaneously in MHSA as shown in Figure 2, double on-chip memory (block RAM) is used in an interleaved read-write manner as a ping-pong buffer to eliminate storage conflicts between different batches.

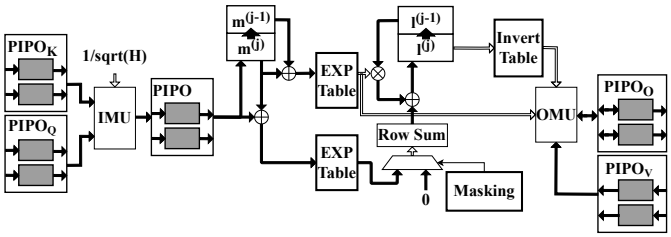

Fig. 3: Architecture of Tiled Flash Attention

C. Feed Forward Network

FFN, similar to MHSA, significantly reduces storage space and increases hardware utilization through the properties of Output Stationary, Input Stationary, and PIPO.

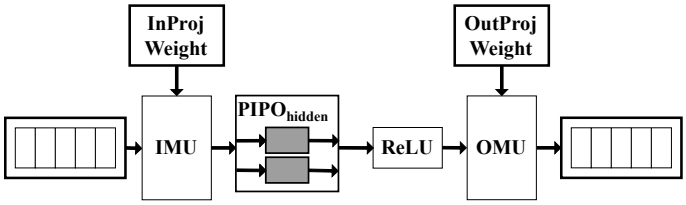

Fig. 4: Architecture of Feed Forward Network

D. Overall Architecture

The overall structure shown in Figure 5, which connects the three layers mentioned above via FIFOs. Notably, to support long life-cycle data for the residual connection, a deeper FIFO is needed to store multiple batches of data, preventing deadlock and stalling. The entire model's input and output are connected to High Bandwidth Memory (HBM) via AXI-streams. Since most activations and weights are stored in on-chip memory, access to HBM is infrequent.

E. Task Scheduling

From the Xilinx Vivado waveforms, we can observe how can these tasks are scheduled by HLS PRAGMA DATAFLOW, shown in Figure 7. This design offers four parameters for users to adjust parallelism: B_T, B_N, B_H, and B_F, which can be used to modify the degree of parallelism for each stage of computation. The required number of blocks can be derived as $T = seq\_len/B_T$, $N = embedded\_dim/B_N$, $H = hidden\_dim/B_H$, and $F = ff\_dim/B_T$. To maximize hardware efficiency and avoid stalls, it is essential to ensure that the number of cycles required for each stage is as close as possible, ideally satisfying $T \simeq N \simeq F$.

IV. Experiments

A. Benchmark and Accuracy of Quantization Model

Neural Data Transformer (NDT) [9] and its extensions [3], [8] are prominent transformer-based architectures in neuroscience domain. The models aim to infer the underlying neural firing rates of animals performing behavioral tasks. In this paper we implemented the NDT model and evaluated its performance on the Neural Latents Benchmark, using the MC_MAZE dataset with co-bps as the evaluation metric [5]. In our experiments, we used 182 neural channels and 180 tokens as model inputs. Using floating-point NDT, we achieved 0.3634 co-bps, whereas mixed-precision fixed-point NDT reached 0.3613, an approximately 0.57% accuracy drop (Figure 6).

B. Power, Resource Utilization and Latency

The power and resource utilization after using the 2023 Vitis HLS and Vivado place and route is shown in Table I. Additionally, we recorded the simulation results on the Xilinx Alveo U55C with a 200MHz clock. It is evident that storing all activations and weights in on-chip memory, along with the use of FIFOs to avoid stalls and deadlocks, has nearly maxed out the BRAM capacity of the board. Therefore, using a small amount of HBM to reduce BRAM usage could be a potential improvement for the future.

BRAM	DSP	FF	LUT	Interval	Power
96.16%	21.25%	2.39%	56.23%	1.146 ms	24.079 W

TABLE I: Power, Resource Utilization and Latency

V. Related Work

FTRANS [4] uses FFT/IFFT to speed up the model. FT [6] compresses the model using mixed block and vector-wise pruning methods. ViA [7] accelerates using a coarse-grained pipeline and multiple DRAM accesses. Compared to these three methods in Table II, this paper employs fewer external memory accesses, instead utilizing more on-chip memory to provide higher bandwidth, achieving better hardware efficiency and throughput. We achieved a 61% to 321% efficiency improvement in GOPS per watt efficiency.

Related Works	FTRANS [4]	FT [6]	ViA [7]	Ours
Year	2020	2021	2022	2024
Model	RoBERT	TinyBert	Swin-T	NDT
Board	VCU 118	Alveo U200	Alveo U50	Alveo U55C
Frequency (MHz)	-	-	300	200
Power (W)	25.13	25	39	24
Throughput (GOPS)	170	75.94	309.6	307.68
Efficiency (GOPS/W)	6.76	3.04	7.94	12.82

TABLE II: Related Work

VI. Conclusion

In this paper, we primarily address (1) reducing External memory access times to increase hardware efficiency, (2) enhancing throughput with a large number of PIPO and FIFO, and (3) providing Dataflow architecture and mixed-precision optimization to finely adjust the bitwidth of variables in each layer, resulting in lower accuracy drop and reduced resource utilization. Additionally, we achieved a 61% to 321% efficiency improvement compared to the previous transformer-based model on FPGA using Xilinx Alveo U55C with a 200M frequency. Moreover, implementing mixed-precision computation with NDT only incurred a 0.57% decrease in co-bps.

References

[1] T. Dao, *Flashattention-2: Faster attention with better parallelism and work partitioning*, 2023. arXiv: 2307.08691 [cs.LG].

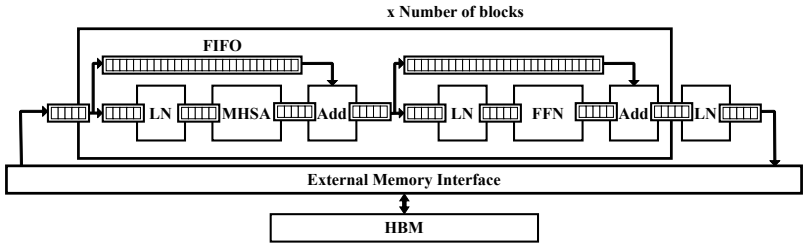

Fig. 5: Overall Architecture. The FIFO depth in the residual connection is approximately 4x the input array.

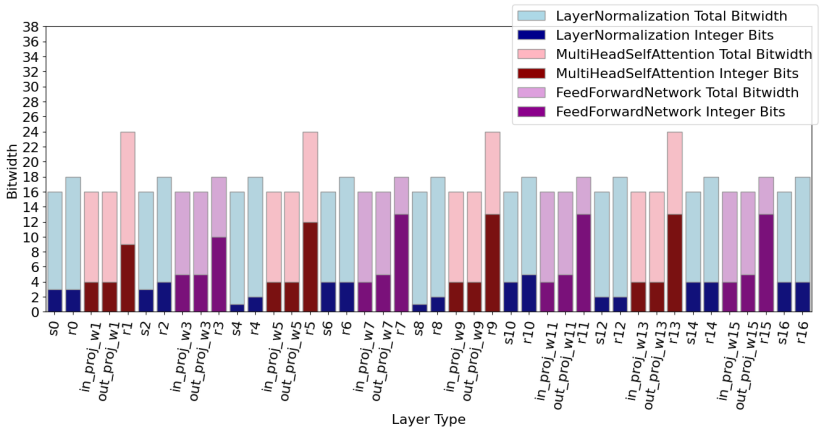

Fig. 6: Mixed precision configuration. s0 refers to the precision of the layer normalization scale. in_proj_w1 refers to Input Projection Weight for Input Linear Projection of MultiHeadSelfAttention. r0 refers to the precision of result(activation).

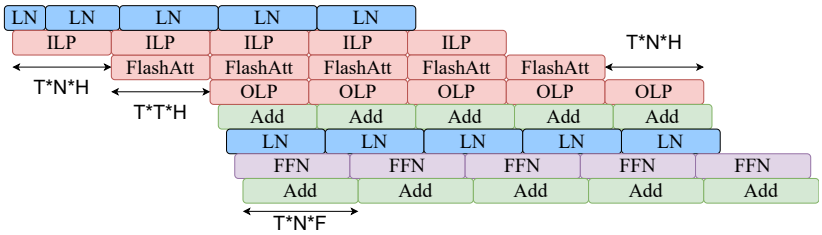

Fig. 7: Task Scheduling. LN, ILP, OLP, FFN refer to Layer Normalization, Input Linear Projection, Output Linear Projection, Feed Forward Network, respectively. T, N, H, F refer to the number of tiles in the sequence length direction, the embedded dimension, the hidden dimension of Multi-Head Self-Attention, the hidden dimension of Feed Forward Network.

[2] T. Dao, D. Fu, S. Ermon, A. Rudra, and C. Ré, "Flashattention: Fast and memory-efficient exact attention with io-awareness," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35, Curran Associates, Inc., 2022, pp. 16 344–16 359. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/67d57c32e20fd0a7a302cb81d36e40d5-Paper-Conference.pdf.

[3] T. Le and E. Shlizerman, "Stndt: Modeling neural population activity with spatiotemporal transformers," *Advances in Neural Information Processing Systems*, vol. 35, pp. 17 926–17 939, 2022.

[4] B. Li, S. Pandey, H. Fang, *et al.*, "Ftrans: Energy-efficient acceleration of transformers using fpga," vol. 7, 2020. DOI: 10.1145/3370748.3406567. [Online]. Available: https://arxiv.org/pdf/2007.08563.pdf (visited on 12/13/2022).

[5] F. Pei, J. Ye, D. Zoltowski, *et al.*, "Neural latents benchmark'21: Evaluating latent variable models of neural population activity," *arXiv preprint arXiv:2109.04463*, 2021.

[6] P. Qi, E. H.-M. Sha, Q. Zhuge, *et al.*, *Accelerating framework of transformer by hardware design and model compression co-optimization*, 2021. arXiv: 2110.10030 [cs.LG].

[7] T. Wang, L. Gong, C. Wang, *et al.*, "Via: A novel vision-transformer accelerator based on fpga," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 11, pp. 4088–4099, 2022. DOI: 10.1109/TCAD.2022.3197489.

[8] J. Ye, J. Collinger, L. Wehbe, and R. Gaunt, "Neural data transformer 2: Multi-context pretraining for neural spiking activity," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[9] J. Ye and C. Pandarinath, "Representation learning for neural population activity with neural data transformers," *arXiv preprint arXiv:2108.01210*, 2021.