FPGA-Based Deep-Pipelined Architecture for Vision Transformer's Multi-Head Attention

Hasitha Muthumala Waidyasooriya
and Masanori Hariyama

Graduate School of Information Sciences
Tohoku University
6-3-09, Aramaki-Aza-Aoba, Aoba, Sendai,
Miyagi 980-8579, Japan
{hasitha, hariyama}@tohoku.ac.jp

Daisuke Tanaka

Department of Mechanical Engineering
National Institute of Technology, Niihama College
7-1, Yagumo, Niihama,
Ehime 792-8580, Japan
d.tanaka@niihama-nct.ac.jp

Abstract— Multihead attention is a crucial component within the Vision Transformer architecture, which plays a significant role in the overall processing. While multihead attention contains a substantial degree of parallelism, it comes with a considerable demand for memory access. This paper proposes an FPGA-based deep pipelined architecture to increase the processing speed while reducing the external memory access. According to the experimental results, proposed accelerator is faster than the multicore CPU implementation. We also discuss the potential to increase the processing speed further.

I. Introduction

The Transformer architecture [1] represents a groundbreaking development in the field of deep learning. Its versatility has led to its widespread application in various machine learning domains, spanning natural language processing and computer vision. Notably, the Vision Transformer, as introduced in Dosovitskiy et al.'s work [2], has gained prominence in image analysis, challenging the conventional dominance of Convolutional Neural Networks (CNNs). A crucial component within the Vision Transformer is multihead attention, which plays a significant role in the overall processing. Given its substantial computational requirements, power-efficient implementation of multihead attention is extremely important.

While multihead attention computation inherently boasts substantial parallelism, it comes with a considerable demand for memory access. This intense memory access requirement can substantially elevate power consumption. FPGAs (Field-Programmable Gate Arrays) are reconfigurable VLSI (Very-Large-Scale Integration) devices that offer power-efficient processing through the creation of customized accelerators tailored to specific tasks. FPGAs provide a solution through pipeline parallel processing, a technique that segments the entire computation into multiple stages connected by storage elements such as buffers. Intermediate data from the preceding stage is stored within these buffers and serves as inputs for the next stage. Consequently, several stages of the computation can operate in parallel while sharing intermediate results. This approach unlocks a significant degree of parallelism while effectively minimizing external memory access, resulting in notably lower power consumption.

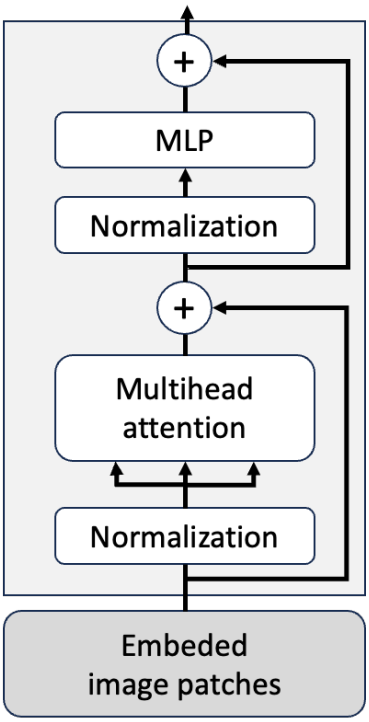

Fig. 1. Transformer encoder architecture.

In this paper, we introduce a deeply pipelined accelerator architecture designed for multihead attention tasks. Proposed architecture is characterized by a substantial number of stages, each responsible for processing a small segment of the overall computation. Importantly, all of these stages operate in parallel, accommodating different input data streams while

delivering a high degree of parallelism with minimal memory access requirements. We discuss the details of our proposed accelerator architecture and explore strategies for scaling it further to increase parallel processing. To validate our approach, we have implemented the proposed architecture on the BittWare IA-840F FPGA board, equipped with an Intel Agilex 7 FPGA. Our experimental results demonstrate that the proposed accelerator outperforms an optimized 20-core CPU implementation while consuming less than 20% of the available floating-point computing resources. This suggests that there is significant untapped potential for boosting processing speed by harnessing the rest of the FPGA's resources.

II. FPGA architecture for multihead attention

Multihead attention consists of multiple "heads", each of which independently learns to attend to different aspects of the input data. Therefore, this model can pay attention to various parts of the input sequence and weigh them differently. In a multihead attention mechanism, the input sequence is transformed into multiple sets of query, key, and value vectors for each head. As shown in Fig.2, each head calculates attention scores between the queries and keys, and these scores are used to weight the values, resulting in multiple context vectors. These context vectors from different heads are then concatenated and linearly transformed to produce the final output.

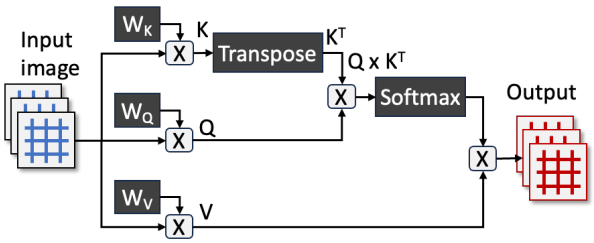

Fig. 2. Flow-chart of the self attention process belonging a single head.

A. Spatial and temporal parallelism of multihead attention computation

The following equations show the computations of a single head of the self attention mechanism.

$$A_{head_i} = \text{softmax}\left(\frac{QK^T}{\sqrt{(d_K)}}\right) \times V$$
$$Q = X \times W_Q$$
$$K = X \times W_K$$
$$V = X \times W_V$$

Input data X is obtained after the positional encoding of the image data. W_Q, W_K, and W_V represent the weights for the query, key, and value data, respectively. These computations are carried out for all the images in a batch and are repeated for all heads. The computations for all heads can be executed in parallel. For each head, all images within a batch can be simultaneously processed. Additionally, the three matrix multiplications required to generate Q, K, and V can also be processed in parallel. Each of these matrix multiplications can be further optimized for a massive degree of parallelism. Consequently, the degree of spatial parallelism in this process is exceptionally high.

As shown in Fig.2, the attention mechanism involves multiple stages. Therefore, it is possible to compute multiple input images, belonging to the computation of multiple heads, in different stages simultaneously. We call this temporal parallelism. Fig.3 shows a timeline of pipeline processing of multiple images. If we can divide the processing into more stages, it is possible to increase the temporal parallelism further.

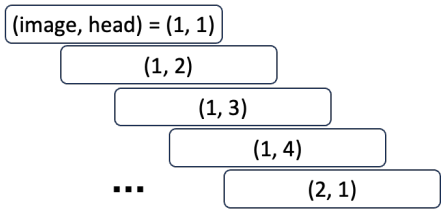

Fig. 3. Temporal parallelism of multiple images processed in multiple heads.

B. Deep-pipelined architecture

Fig.4 illustrates the FPGA architecture we propose. This architecture is composed of several stages, with each stage containing a buffer. Notably, the intermediate results from the preceding stage are stored within these buffers. In the final stage, the output buffer concatenates the data from all heads, delivering the final output.

The number of stages can be expanded by subdividing the processing within each stage, allowing for a finer-grained approach to computation. The matrix multiplication is executed using a systolic array architecture, as shown in Fig.5. This approach enables concurrent processing within each of these stages, resulting in a substantial increase in processing speed, while effectively reusing intermediate data among different stages.

Further memory access reduction can be achieved through the temporal storage of reusable data. Considering that the input image data are reused across all heads, they are initially accessed from external memory and subsequently utilized while being stored within the FPGA. Furthermore, the weight matrices,

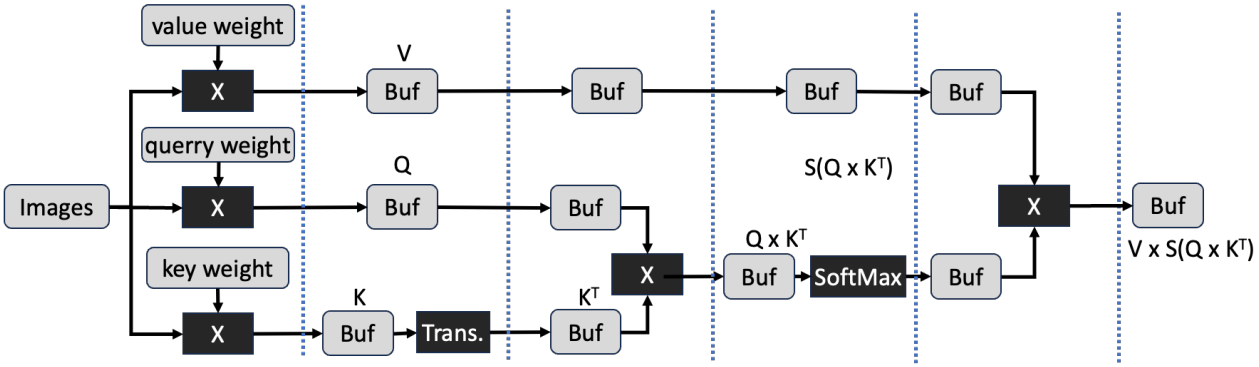

Fig. 4. Architecture of self attention computation of a single head.

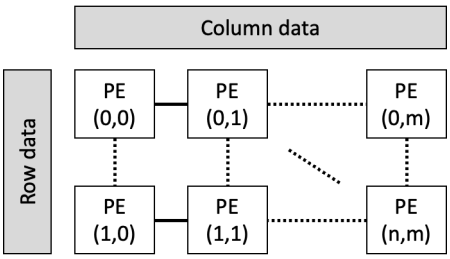

Fig. 5. Systolic array architecture for matrix multiplication.

TABLE I
PROCESSING TIME OF 12 HEADS.

Image size	32
Number of channels	3
Patch size	8
MLP size	3072
Hidden size	768
Number of heads	12
Mini batch size	200

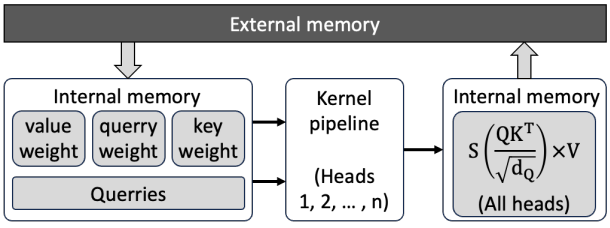

Fig. 6. FPGA accelerator architecture

which can be shared among all images in a batch, are also stored within the FPGA, minimizing the need for repeated external memory access and enhancing overall computational efficiency.

Fig.6 shows the overall architecture. The input image data and weights are initially stored in the external memory. Then a portion of the data are transferred to the internal memory of the FPGA. Those data are repeatedly accessed by the kernel pipeline. The output data of all heads are stored in the internal memory. After the outputs of all heads are received, those data are transferred back to the external memory and accessed by the CPU.

III. EVALUATION

For the evaluation, we use BittWare IA-840F FPGA board that contain Intel Agilex-7 AGF027 FPGA [3]. The FPGA kernels are compiled using Intel FPGA SDK for OpenCL version 2023.1 with Quartus prime pro 21.4. The CPU used for the comparison is Intel Xeon Silver 4316 with 20 cores. The CPU version of the multihead attention computation is compiled using Intel OnelAPI DPC++/C++ compiler 2023.1. We use Intel mkl (math kernel library) to accelerate matrix multiplications. CIFAR-10 dataset is used for the evaluation [4]. Table I shows the encoder parameters.

Compared to the optimized CPU implementation, the FPGA processing is 1.17 times faster as shown in Table II. The resource utilization is shown in Table III. The maximum resource usage is for the RAM blocks. RAM blocks are used to store a portion of inputs, outputs and intermediate results. Since only 19% of the DSP resources are used for the computation, there is a huge potential to increase the processing speed significantly. Note that increasing parallelism does not result in any increase of the RAM blocks, since we do not need to increase the storage any further. The output data are exactly the same as the result of CPU that is done using single-precision floating-point computation.

TABLE II
COMPARISON OF THE PROCESSING TIME FOR 12 HEADS.

Intel Xeon 4316 (20 core)	163 (ms)
Intel Agilex IA-840F FPGA board	139 (ms)

TABLE III
RESOURCE UTILIZATION.

Resource	Utilization	Percentage
Registers	1,675,326	
Logic	505,531	55%
DSP	1,644	19%
RAM blocks	9,197	69%
Memory	19.3 MB	60%

IV. Conclusion

Our proposal introduces a deeply pipelined architecture for multihead attention mechanism. Notably, it outperforms a parallel CPU multicore implementation, demonstrating a significant potential for further speed improvements. Moreover, the proposed accelerator minimizes external memory access, offering substantial energy efficiency gains. Looking ahead, our future work aims to extend this approach to implement the entire Vision Transformer architecture on an FPGA.

Acknowledgment

This research is partly supported by MEXT KAKENHI, grant number 20H04197.

References

[1] **Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need.** ***Advances in neural information processing systems*****, 30, 2017.**

[2] **A Dosovitskiy, L Beyer, A Kolesnikov, D Weissenborn, X Zhai, and T Unterthiner. Transformers for image recognition at scale.** ***arXiv preprint arXiv:2010.11929*****, 2020.**

[3] **BittWare. IA-840F.** `https://www.bittware.com/products/ia-840f/`**, 2023. [Online; accessed 31-Oct.-2023].**

[4] **The CIFAR-10 dataset. 2023. [Online; accessed 31-Oct.-2023].**