High-Performance Systolic Array Architecture for Accelerating Convolutional Neural Networks and Multi-Head Attention Mechanisms

Ruikai Zhai

***Abstract*—In the field of target detection and image classification, although hardware accelerators for CNN and Transformer are widely available, accelerators specifically designed for Transformer-CNN hybrid networks are relatively rare. Due to the high computational complexity, storage requirements, memory bandwidth limitations, and parallel computation difficulties of hybrid networks, the implementation of the model on hardware is challenging. To address these issues, this study proposes a dedicated hardware accelerator with a configurable Systolic Array computational architecture, on the one hand, designing an Img2col module for converting 3D feature maps into 2D matrices, and utilising the Systolic Array to implement convolution operations with different sizes as well as matrix multiplications, which is specifically designed to accelerate the inference of Transformer-CNN hybrid networks. On the other hand the accelerator is extremely flexible and configurable, parametrically configuring the Systolic Array according to computational needs ensures that its performance is fully exploited, while the use of on-chip buffers for storing maps, weight data, and intermediate results reduces off-chip memory accesses and power consumption, and improves data reusability.**

Our well-designed accelerator was tested on Xilinx Zynq, and the experimental results show that the accelerator exhibits excellent performance in both CNN and Attention Mechanism computation, with an output of up to 608.6 GOPS/W. The aim of this study is to build efficient hardware accelerators that utilise efficient computational units and memory structures for Transformer-CNN hybrid network to accelerate the processing and improve the performance of CNN and Transformer deployed on hardware.

***Index Terms*—FPGA, Transformer, CNN, Systolic Array.**

I. INTRODUCTION

Over the past few years, Transformer has achieved remarkable results in the field of Natural Language Processing (NLP). As an extensible framework, Transformer provides effective solutions for many complex NLP tasks, such as machine translation, text summarisation, and question and answer systems. However, the application of Transformer is not limited to the field of NLP, but in the field of computer vision, Transformer is also becoming an important new tool for solving various image processing tasks[1].

CNN[2] and ResNet[3], which have been dominant in computer vision tasks, focus on the extraction of local features of an image, which makes it difficult to capture global information with their limited receptive fields[4]. While Transformer's Multi-Head Attention (MHA) captures the relationship between global features through Q, K, and V matrices to obtain richer feature information [5], it is relatively weak in processing local information. Therefore, more and more researchers began to explore how to combine CNN and Transformer, and Transformer-CNN hybrid neural networks can complement each other's advantages[6], and the emergence of these hybrid models brings new ideas and methods to the field of computer vision, and also provides a new way to realize more efficient and accurate image processing tasks[7]. However, due to the significant difference between the two computational approaches[8], most of the accelerators accelerate CNN or Transformer alone, and relatively little research has been done on hardware accelerators specifically designed for CNN-Transformer hybrid networks. Therefore, in this paper, we design an efficient hardware accelerator for DETR, a hybrid network, using the img2col method to implement convolution-to-matrix multiplication mapping, and the Systolic Array to implement different types of convolution and matrix multiplication operations, which significantly improves the flexibility of convolution computation[9] and speed.

The main work of this paper is as follows：

1）We propose an innovative Systolic Array architecture that contains two different computational modes designed to provide acceleration for Transformer-CNN hybrid networks. Through the designed mapping method, we successfully transform the two computational processes, Convolution and Transformer, into matrix multiplication on Systolic Array for efficient computational processing.

2）The accelerator we designed is highly flexible and configurable, dynamically adjusting the number and arrangement of the processing elements (PEs) in the Systolic Array according to the size of the input matrix, ensuring that it achieves its maximum performance when performing parallel computing tasks. It also implements operators commonly used in neural networks such as Add, Maxpooling, and Concat.

3）This accelerator achieves a peak throughput of 608.6 GOP/S using Int8 computing. The accelerator is 5.4 times more energy efficient compared to CPUs, 1.45 times more efficient compared to GPUs, and 5.22 times more efficient in terms of speed and energy efficiency compared to existing accelerators.

Manuscript received March 16, 2025
Ruikai Zhai, School of computer science and technology, Tiangong University, Tianjin, China.

II. Related Work

Systolic Array usually has two forms: weight-fixed and output-fixed, as shown in Fig. 1. For weight-fixed SA, the weight data is preloaded into the PE, and then the input data is fed into the Systolic Array from the left side and propagated to the right so that the intermediate result of the operation is propagated from the top to the bottom[10]. For the output-fixed type SA, the input data is shifted in a consistent manner, one position per clock cycle. Unlike the weight-fixed SA, the weight data of the output-fixed SA is not preloaded and stored statically, but flows through the PE like the input data, and the intermediate results of the computation are accumulated inside the PE to output the final result.

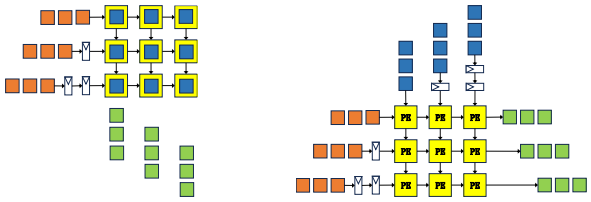

Fig. 1. Weight-stational SA (left), Output-stational SA(right).

In this paper, the output fixed type SA is used, for the multiply-accumulate operation of convolution, the 3D feature map and weights can be converted to a 2D matrix by the Img2col method, and then input into the Systolic Array for matrix multiplication respectively, and then the result can be rearranged to get the common channel-first data format. And for the matrix multiplication between Q K V matrices in MHA can also be achieved by Systolic Array. Therefore, we unify these two very different computation methods by using Systolic Array, which not only reduces the consumption of hardware resources such as DSP, BRAM, etc., but also the matrix multiplication itself is suitable for parallel computation by FPGA, and the different phases of the matrix multiplication (e.g., multiplication and accumulation) can be carried out in each PE at the same time. This processing can significantly increase throughput and reduce latency.

III. Architecture

A. *Architecture of SA*

This section presents a scalable Systolic Array architecture. It can be configured into different dimensions and sizes, such as 1*8*8, 4*8*8, 8*16*16 (Slice, Height, Width), by increasing the number of computational units in the array, thus easily expanding the computational power to meet the growing computational demands. The internal structure of some of the PEs, as shown in Fig. 2a, uses double buffers for data caching for matrix A. One buffer is used for reading and the other buffer is used for writing. While one buffer is being read, the other is being written. In this way, read and write operations can be performed simultaneously without conflicts. First, data from the front Kernel Size rows are written to Buffer A1, followed by data from the back Kernel Size rows being written to Buffer A2 when data from Buffer A1 is fed to the array.This overlapping of data transfer and computation significantly improves performance. Matrix B is also cached by double buffers and then input to the array in column order, and the array can process the data of Slice * Height columns at one time.The PE internally uses multiplier, adder, and registers to perform multiply-add operations, and two registers to cache the input data to flow into the next PE in the next clock cycle.

The 2D structure of a Slice in a 3D array is shown in Fig. 2b. A1, A2... B1, B2... denote the data streams of matrix A and matrix B, which flow and participate in the computation between the PEs in the column direction and row direction, respectively. O1, O2 indicates each row of output in the Systolic Array, only one PE in each row will output a valid result in a cycle, whether the output result is valid or not is controlled by the counter.

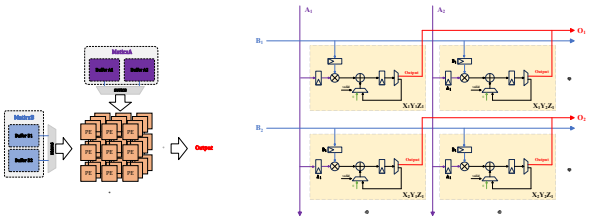

Fig. 2. 3D Architecture of systolic array(left),2D Architecture of systolic array(right).

Take two 4*4 pixel matrices as an example, matrix multiplication operation is performed on matrix A and matrix B. Matrix A and matrix B enter the Systolic Array in the order of columns and rows respectively, and at the beginning of the computation in the first clock cycle pixel a enters the PE with pixel 1 to do the multiplication operation and saves the result 1a in the register. From the column direction, the second clock cycle pixel a continues to flow down into PE (2,1), at which time pixel point 5 also enters this PE to do the multiplication operation with pixel point a and saves the result 5a. At the same time pixel b and pixel 2 enter PE(1,1) to do multiplication and addition with the previously saved result 1a and then the result is saved. Similarly in the row direction, the second clock cycle pixel 1 flows into PE(1,2) and does multiplication with pixel e to save the result. And so on until pixel p and pixel 16 do the multiply-accumulate operation at PE(4,4), representing the end of matrix multiplication. When the calculation results are output, the multiply-accumulate results of the first column of matrix A and the first row of matrix B are output at the moment of CLK1, and the calculation results of the second column of matrix A and the first row of matrix B are output at the moment of CLK2. And so on, each row of Systolic Array has an output for each clock cycle, and then the output results can be data rearranged, as shown in Fig 3.

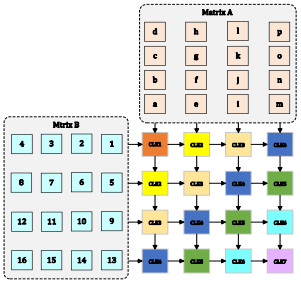

Fig. 3. Clock period of the output result.

B. *Mapping Convolution to Matrix Multiplication*

Tensor, as the basic building block of neural network data

storage, plays a crucial role in network computation tasks. Then how to implement convolution operation by matrix multiplication for such data format as tensor, a large amount of academic research is devoted to Img2Col[11], which has the core idea of converting the pixel values of input feature maps into column vectors and storing them in a large matrix. This approach can greatly simplify the computational process of convolutional neural networks, because each convolution operation can be regarded as a matrix multiplication operation, and can support convolution operations of arbitrary size, which only requires dynamic configuration of the Img2Col module. The implementation of the Img2Col algorithm requires the use of a number of techniques, such as the use of stride and padding, to ensure that the output of the convolution operation is of the correct result and size. The Img2Col algorithm is a very effective image processing algorithm that can greatly accelerate the training and inference process of neural networks, and a large number of academic studies have proved the effectiveness of this algorithm and it has been widely used[12].

Traditional convolutional accelerators tend to focus on providing hardware support for convolutions of specific sizes, such as 1×1 and 3×3 convolutions. While this design improves the efficiency of these common operations, it also limits the flexibility and scalability of the accelerator to adapt to different network architectures. These accelerators may not provide optimal performance when confronted with uncommon convolution sizes, such as 7×7 convolution.The DETR model is a good example of a 7×7 convolution used in Backbone to extract features from input feature maps.Convolution kernels of this size may not run efficiently on conventional accelerators, which in turn can affect the performance of the entire network.

The Img2col module allows for the equivalent mapping of convolution operations to matrix multiplication. This mapping converts the convolution operation to matrix multiplication by converting the convolution kernel to a matrix, converting the input feature maps to another matrix, and then inputting the column vectors in the matrix sequentially to the Systolic Array[13]. Various sizes of convolution operations such as 1*1, 3*3, 5*5, 7*7, 16*16 can be achieved by mapping.

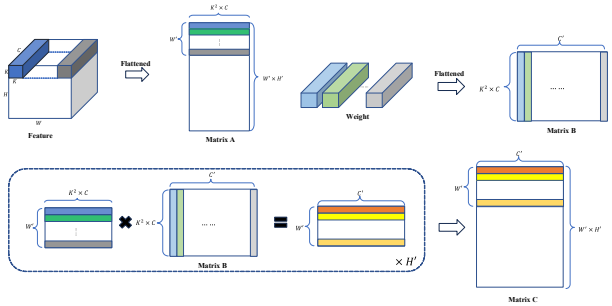

Fig. 4. CNN mapped to matrix multiplication.

Assume that the dimension of the input feature map is [B, H,W,C] (assuming a batch size of 1 here), the dimension of t he convolution kernel is $[C',H',W',C]$, and the dimension of the output feature map is $[B',H',W',C']$. The input feature ma p and convolution kernel can then be spread into matrices A $[(W' \times H') \times (K^2 \times C)]$ and B$[(K^2 \times C) \times C']$, respectivel y. Due to the image data storage format, the weight values of the convolution kernel with the pixel values of each sliding window of the input feature maps will be unfolded along the dimensions of the channel (the order of unfolding is [C,W,H]) as shown in Fig. 4. Further, by matrix multiplication operat ion A× B, we can get the matrix representation C$[(W' \times H') \times C']$ of the final output feature map.

We can clearly observe that: each row of matrix A corresponds to the spreading of each sliding window in the input feature map; each column of matrix B corresponds to the spreading of the weight values of the convolution kernel on each output channel; and each row of matrix C represents the eigenvalues of all the channels for each pixel point in the output feature map. So what we are trying to do is to get the result we want by performing an additional memory rearrangement of the inputs and outputs, converting the 3D data into a 2D matrix and then performing a matrix multiplication operation. By corresponding the 3D convolution operation to this matrix multiplication, the tensor cells are expanded in the channel dimension as shown in Fig 5.

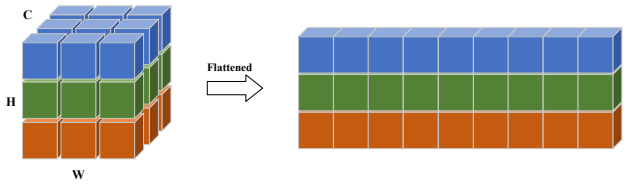

Fig. 5. 3-D tensor is flattened along the channel dimension.

C. *Calculation of multi-head attention mechanism*

The essence of the Transformer model lies in its multi-head attention mechanism. This mechanism is able to capture complex relationships in a sequence by splitting the input sequence into multiple heads and computing the attention of different parts in parallel. At the underlying computation, the process of multi-head attention can be clearly represented as a matrix operation, which is easy to implement in hardware, allowing the distribution of matrix multiplication computations across multiple computational units for parallel processing[14].

In DETR, a 1x1 convolution is used to reduce the channel dimension of the activation f from C to a smaller dimension d, thus creating a new feature map $Z_0 \in R^{d*H*W}$. The encoder expects a sequence as input, so it compresses the spatial dimension of Z_0 into one dimension to obtain a feature map of d*HW. Positional encoding, as the other input to the encoder, also compresses the spatial dimension into one dimension, obtaining a sequence of dimension d*HW, which is used to provide positional information.

The three matrices Q (Query), K (Key), and V (Value) are obtained by summing the spatial dimensionally compressed activation f and Positional encoding. These three matrices are the basis of the multi-head attention mechanism, and represent different roles in the attention computation: the Query matrix is used to determine the focus of attention, the Key matrix is used to compare the relative importance of the input elements, and the Value matrix contains the information that is ultimately to be aggregated. Q, K, and V are generated in parallel, and their weights are concatenated to form a larger matrix. This design allows each head to process the information independently, thus enabling rich features to be captured in different representation subspaces.

In this way, the multiple attention mechanism is able to effectively mimic selective attention in the human visual system, being able to focus on important parts while ignoring irrelevant information when processing complex scenes[15]. In DETR, this mechanism enables the model to learn complex relationships between image blocks, leading to excellent performance in target detection and other visual tasks.

IV. Evaluation And Analysis

A. *Analysis of resources and power consumption*

To achieve efficient and less resource-intensive neural network deployments, we adopt a strategy that minimises the consumption of DSP resources in the FPGA while ensuring efficient processing of critical computational tasks. We use DSP multiplexing to reduce resource consumption by implementing the computation of two sets of int8 type data in a single DSP. Especially in Systolic Array, which is used for computationally intensive tasks, this resource-optimised strategy not only reduces the hardware cost, but also reserves valuable resource space for subsequent expansion of the whole system. In addition, the configurable compute logic units enhance the versatility of the accelerator for networks of different sizes, and the design dynamically adjusts the use of DSPs according to the computational demands of the neural network model, thus improving data throughput and ensuring real-time performance. In this way, even neural network models with large computational volumes can achieve efficient operation on limited hardware resources, ensuring performance and avoiding waste of resources. As shown in Fig. 6, we configured Systolic Array with different sizes. The power consumption increases from 3.5W for the (1,8,8) configuration to 13.8W for the (8,16,16) configuration. however, the throughput increases from 25.6 GOP/s to 608.6 GOP/s, resulting in a 20-fold improvement. In addition, the energy efficiency increased from 7.17 GOPs/W to 41.4 GOPs/W.

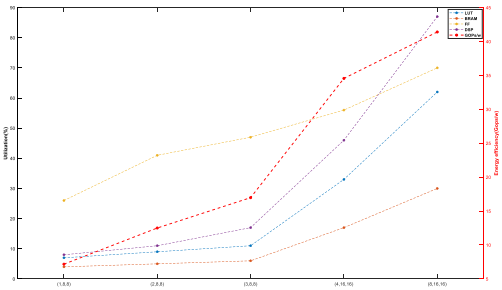

Fig.6.Comparison of Characteristics and Resource Utilization among
Accelerators With Different Sizes.

B. *Performance comparison*

To further demonstrate the superiority of our proposed computational architecture, Table 1 shows the performance comparison of our design with existing FPGA-based accelerators. For all evaluated networks, our accelerator achieves 38.01 - 41.4 GOP/s/W in terms of energy efficiency (GOP/s/W), which outperforms the accelerator proposed by [7][16], although the accelerator proposed by [17] achieves 1030.93 GOP/s/W in terms of energy efficiency, it only implements Transformer- base and cannot implement deep learning operators such as convolution, pooling, ADD, etc., whereas our proposed accelerator can implement the complete ResNet-50 as well as Transformer's Encoder and Decoder deployment. In addition, the accelerator proposed in [7] uses 16bit quantisation for computation, making the DSP utilisation inefficient, with much higher power consumption and much lower throughput than ours. Although the throughput of the accelerator proposed in [16] is slightly higher than ours, the power consumption is higher than ours, making the energy efficiency lower than ours.

Table 1 Comparison with related FPGA accelerators

Related work	[7]	[16]	[17]	This work
Platform	Alveo U50	XCVU37p	XCVU13p	ZCU102
DSP	2420	1024	129	2198
BRAM	1002	448	498	663
LUT	258k	141k	472k	212k
FF	257k	223k	218k	256k
Power	39	16.9	16.7	13.8
Throught	309.6	576.52	1030.93	524.6

V. Conclusion

In this paper, we propose a hardware accelerator that uniformly maps two types of large-scale computations onto a Systolic Array according to the computational characteristics of models based on the attention mechanism and convolution, which not only improves the computational efficiency, but also simplifies the hardware design, enabling the accelerator to support various network structures with higher flexibility and scalability. We have achieved accelerated processing of deep learning networks using an efficient Systolic Array and a reasonable memory structure, and achieved good results. Further future research could explore the scalability of hardware accelerators to handle more complex tasks and models.

References

[1] Liu Y, Zhang Y, Wang Y, et al. A survey of visual transformers[J]. IE EE Transactions on Neural Networks and Learning Systems, 2023.

[2] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[J]. Advances in neural information processing systems, 2012, 25.

[3] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.

[4] Xu R, Ma S, Wang Y, et al. Configurable multi-directional systolic array architecture for convolutional neural networks[J]. ACM Transactions on Architecture and Code Optimization (TACO), 2021, 18(4): 1-24.

[5] Wang H, Ma C. An optimization of im2col, an important method of CNNs, based on continuous address access[C]//2021 IEEE International Conference on Consumer Electronics and Computer Engineering (ICCECE). IEEE, 2021: 314-320.

[6] Chen Y, Li T, Chen X, et al. High-frequency systolic array-based transformer accelerator on field programmable gate arrays[J]. Electronics, 2023, 12(4): 822.

[7] Wang T, Gong L, Wang C, et al. Via: A novel vision-transformer accelerator based on fpga[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2022, 41(11): 4088-4099.
[8] Kyriakos A, Kitsakis V, Louropoulos A, et al. High performance accelerator for cnn applications[C]//2019 29th international symposium on power and timing modeling, optimization and simulation (PATMOS). IEEE, 2019: 135-140.
[9] Shawahna A, Sait S M, El-Maleh A. FPGA-based accelerators of deep learning networks for learning and classification: A review[J]. ieee Access, 2018, 7: 7823-7859.
[10] Wang B, Ma S, Zhu G, et al. A novel systolic array processor with dynamic dataflows[J]. Integration, 2022, 85: 42-47.
[11] Chellapilla K, Puri S, Simard P. High performance convolutional neural networks for document processing[C]//Tenth international workshop on frontiers in handwriting recognition. Suvisoft, 2006.
[12] Wang H, Ma C. An optimization of im2col, an important method of CNNs, based on continuous address access[C]//2021 IEEE International Conference on Consumer Electronics and Computer Engineering (ICCECE). IEEE, 2021: 314-320.
[13] Ahmad A, Pasha M A. Optimizing hardware accelerated general matrix-matrix multiplication for CNNs on FPGAs[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2020, 67(11): 2692-2696.
[14] Shao H, Lu J, Wang M, et al. An efficient training accelerator for transformers with hardware-algorithm co-optimization[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2023, 31(11): 1788-1801.
[15] Guo R, Chen X, Wang L, et al. CIMFormer: A systolic CIM-array-based transformer accelerator with token-pruning-aware attention reformulating and principal possibility gathering[J]. IEEE Journal of Solid-State Circuits, 2024.
[16] Li T, Zhang F, Fan X, et al. Unified accelerator for attention and convolution in inference based on FPGA[C]//2023 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 2023: 1-5.
[17] Lu S, Wang M, Liang S, et al. Hardware accelerator for multi-head attention and position-wise feed-forward in the transformer[C]//2020 IEEE 33rd International System-on-Chip Conference (SOCC). IEEE, 2020: 84-89.