Revisiting Reconfigurable Acceleration of Vision Transformer with Patch Pruning

Hanning Chen*, Yang Ni*, Wenjun Huang*, Hyunwoo Oh*, Tamoghno Das*, Fei Wen†, and Mohsen Imani*
*University of California, Irvine, Irvine, CA 92697, USA
†Texas A&M University, College Station, TX 77843, USA
Email: *{hanningc, yni3, wenjunh3, hyunwooo, tamoghnd, m.imani}@uci.edu, †fei8wen@gmail.com

***Abstract*—Vision Transformers (ViTs) have become the backbone of numerous cutting-edge vision applications. The attention modules within ViTs play a crucial role in modeling spatial relationships between pixels. Although these attention modules enhance the accuracy of ViT models, they also increase computational demands, limiting the deployment of ViTs in edge computing environments. To address this issue, prior research has focused on optimizing ViTs from both software and hardware perspectives. A notable software optimization technique is reducing the image patches involved in attention computations. Two common methods to achieve this are window attention and patch pruning. However, they introduce new challenges for existing hardware platforms regarding attention computation. Therefore, it is essential to develop new hardware modules to simultaneously support pruned attention computations and efficient window shifts. In this study, we introduce an FPGA-based token reduction vision transformer accelerator called TRFPA. Experiments conducted on the Xilinx ZCU104 and Alveo U50 demonstrate that TRFPA outperforms previous FPGA-based ViT accelerators, achieving a 7× speedup and a 3× improvement in energy efficiency.**

I. Introduction

Vision transformers (ViTs) [1] have become popular as foundational models across a range of applications [2]. Unlike earlier convolutional neural networks (CNNs) [3], ViT models excel at global reasoning using image token-wise multi-head self-attention (MHSA). Nonetheless, ViT models are computationally demanding compared to CNNs, restricting their deployment in environments with limited resources. A further complication is that existing general-purpose computing platforms like CPUs and GPUs struggle to deliver both acceleration and energy efficiency for ViT applications [4]. As a result, various hardware and software co-design strategies have been developed to accelerate ViT, specifically for MHSA computation [5].

There are several strategies to optimize ViT MHSA from a model compression perspective, including quantization [6], pruning (QKV weights and attention heads) [7], and knowledge distillation [8]. These methods focus on reducing the ViT model size itself, thereby lowering MHSA memory consumption. Further architectural optimizations include using AI compilers like TensorRT [9] or Triton [10], and customized low-bit precision, high-speed computing hardware like NVIDIA tensor cores [11] to enhance computation efficiency. However, even with these optimizations, the computation overhead of ViTs remains a significant challenge. Recently, a new approach to compressing ViT MHSA involves reducing image tokens [12]. This is based on the intuition that MHSA computation overhead is approximately **quadratic** relative to the number of image tokens. A common strategy uses smaller machine learning models to drop a portion of image tokens (known as **token pruning**) before MHSA computation [13] or to perform local attention (known as **window attention**) [14]. Both token pruning and window attention aim to reduce the number of image tokens participating in MHSA computation.

979-8-3315-2710-5/25/$31.00 ©2025 IEEE

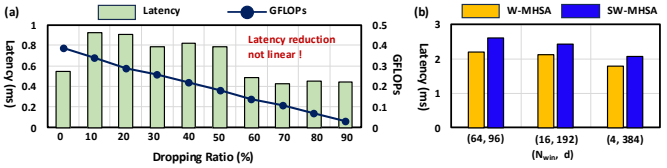


Fig. 1. (a) Impact of token reduction ratio on the computation and latency of ViT multi-head self-attention (MHSA). (b) Latency comparison between window MHSA (W-MHSA) and shifted-window MHSA (SW-MHSA) in the Swin Transformer. All results are measured on an NVIDIA RTX 4090.

While window attention and token pruning present promising methods for accelerating ViT models, implementing these optimizations on actual hardware platforms remains challenging. There are three primary challenges. The first is that, following window attention, a window shift operation is required, leading to non-continuous memory access during the next attention phase [14]. The second challenge arises from token pruning, which makes the feature maps of image embeddings and attention scores sparse [15], [16], necessitating an effective sparse attention acceleration engine to support this structure. The third challenge is that existing domain-specific accelerators support either window attention [17] or token pruning [15], but lack the capability to unify both within a single accelerator. As shown in Figure 1, neither image token pruning nor window attention achieves optimal performance on the GPU.

To address these challenges, this paper proposes an FPGA-based token reduction vision transformer accelerator called **TRFPA**. We selected FPGA as the implementation platform due to its high parallelism and low power consumption compared to CPU and GPU alternatives [18]. To the best of our knowledge, **TRFPA** is the first FPGA-based accelerator that unifies window attention and pruned global self-attention. The main contributions of this paper are as follows:

- We design a network-on-chip IP to support on-the-fly window shift operations, ensuring continuous memory access for each window attention.
- To unify window attention and global attention on the same hardware platform, we apply a tiling strategy for loading image token embeddings.
- We also design a pipelined window attention engine to support both window attention and tiling-style global attention.

The results show that the FPGA platform provides an average speedup of 7x and a 3x improvement in energy efficiency compared to previous FPGA-based ViT accelerators.

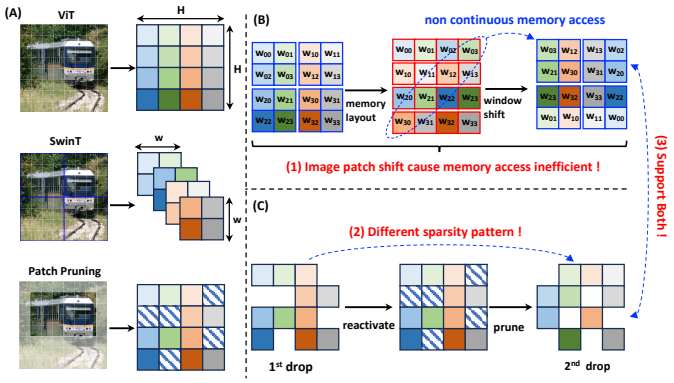


Fig. 2. (A) A comparison is made among normal vision transformer self-attention, Swin transformer window attention, and patch-pruned sparse self-attention. (B) Computing challenges faced by existing hardware platforms when executing window attention. (C) Challenges in computing pruned sparse attention.

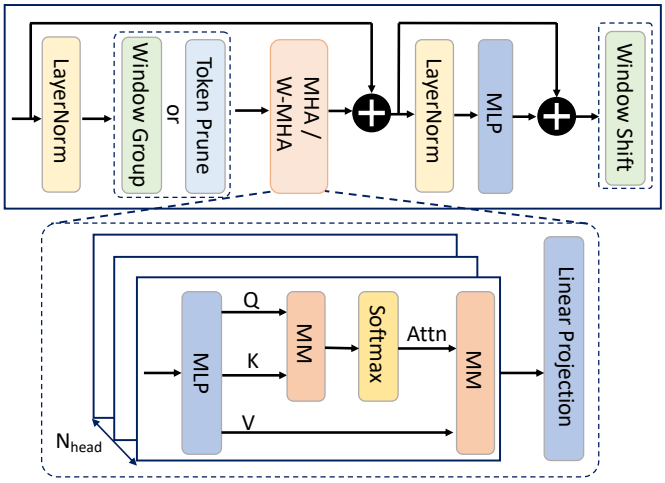


Fig. 3. Model architecture of vision transformer (ViT).

II. Related Works

A. Model compression of ViT

Vision transformers (ViT) have achieved impressive outcomes in vision tasks, but their significant computational demands present a challenge. The main computational load in ViT stems from the self-attention blocks [1]. Various model compression strategies have been proposed to address this challenge with ViT [19]. These strategies include weight pruning [7], knowledge distillation [8], and attention matrix specification [4], all targeting the reduction of self-attention computational demands by decreasing model size. Recently, a new method called activation pruning has gained popularity. This technique works by reducing the number of image patches involved in self-attention, thus decreasing latency and memory usage. There are two main methods. The first is known as **window attention**, employed by swin transformer (SwinT) [14] and segment anything (SAM) [20]. The basic principle is that, rather than performing global attention on all image patches, the model should focus on local attentions and only consider patches within a local window. Another approach is **patch pruning** [21], which aims to eliminate unimportant image patches during self-attention processing. For example, in segmentation tasks, a smaller model can be used to discard easily-classified image tokens [22] and megre tokens with the same semantic meaning [23].

B. FPGA acceleration of ViT

FPGA-based ViT accelerators have been extensively researched due to their energy efficiency and reconfigurability [24]. These unique attributes make FPGAs ideal candidates for hardware and software co-design, often surpassing GPU capabilities. Previous research has primarily targeted FPGA accelerators optimized for aspects such as ViT weight quantization [25], [26], head pruning [27], and mixtures of experts (MoE) [28]. Although FPGA-based accelerators for token reduction exist, challenges remain. Notably, HPTA [29] and SWAT [17] focus on the Swin Transformer model (SwinT), emphasizing sparse mask support during window attention but overlooking the memory read overhead associated with the window shift operation. For image token pruning, HeatViT [15] and ViT-ToGo [30] leverage systolic arrays to support both the main ViT model and pruning models; however, they do not optimize the sparsity of image tokens post-pruning. Moreover, all previous efforts are limited to supporting either window attention or global attention with token pruning, without integrating both optimization methods on a single hardware platform. We will resolve all these challenges at this work.

III. Backgrounds

A. Vision Transformer

Figure 3 illustrates the typical design of a vision transformer (ViT). The ViT model can be divided into three sections: pre-attention, multi-head self-attention (MHSA), and post-attention. The pre-attention steps primarily deal with image pre-processing operations, like padding and resizing, as well as creating patch embeddings. The post-attention phase is largely concerned with linear projections. Both the pre-attention and post-attention stages primarily perform dense matrix-to-matrix multiplications (GEMM), which can be efficiently executed using a systolic array or multiplier adder-tree architecture, as detailed in [15].

The majority of the computation for ViT inference is occupied by multi-head self-attention (MHSA) [16]. As depicted in Figure 3, assume the output of the patch embedding layer is of size $N \times d$, where N represents the total number of image tokens and d is the embedding dimension. Initially, the input patch embedding is processed by a fully connected (MLP) layer with dimensions $d \times 3d$, producing the query ($\mathbf{Q}$), key ($\mathbf{K}$), and value ($\mathbf{V}$) embedding matrices. Each of these matrices has dimensions $N \times d$. Subsequently, these $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ matrices are divided into N_{head} parts, resulting in $\mathbf{Q}_i$, $\mathbf{K}_i$, and $\mathbf{V}_i$, where i is the index of the head. Within each head, the computation of attention can be represented as:

$$A_i = Softmax(\frac{\mathbf{Q_i}\mathbf{K_i}^T}{\sqrt{d_i}}) \quad (1)$$

$$Attention(\mathbf{Q_i}, \mathbf{K_i}, \mathbf{V_i}) = A_i \times \mathbf{V_i} \quad (2)$$

For head i, let A_i represent the attention score matrix with dimensions $N \times N$. The softmax function is applied along the last axis of $\frac{\mathbf{Q_i}\mathbf{K_i}^T}{\sqrt{d_i}}$. As mentioned in section IV, the main challenge in performing window-attention and tiling optimization lies in the softmax computation. Subsequently, the attention outputs are concatenated into an $N \times d$ matrix. This is followed by the projection and MLP layers, which map the image embeddings into a higher-dimensional feature space. The total multiplication and accumulation (MAC) operations required for self-attention (T_{atten}) is:

$$T_{atten} = 4Nd^2 + 2N^2d \quad (3)$$

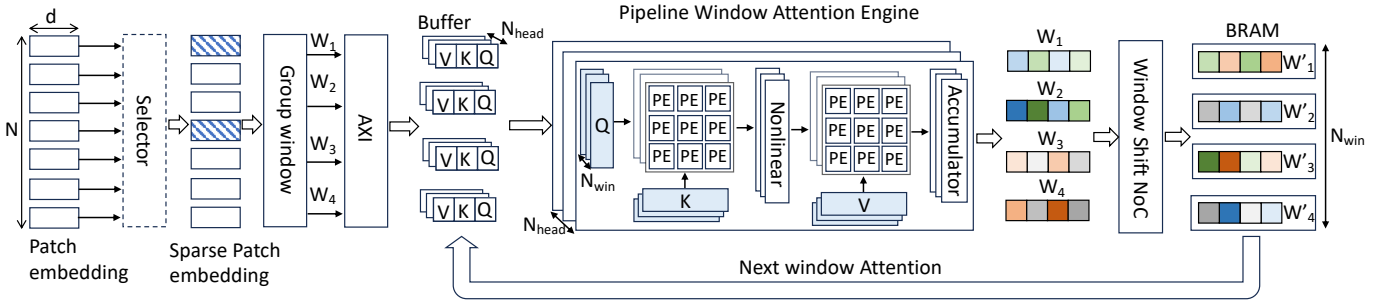


Fig. 4. Top architecture of TRFPA

As demonstrated in equation 3, a direct approach to lower the MHSA computation involves decreasing the number of image tokens involved in attention computation.

B. Window Attention

Given that the computational expense of MHSA scales roughly with the square of the number of image tokens (N), it is typically more feasible to execute local attention within a reduced window rather than using global attention, which we define as window multihead self-attention (**W-MHSA**). With a fixed window dimension of $M \times M$, the total MAC operations are:

$$T_{win} = 4Nd^2 + 2M^2Nd \tag{4}$$

Performing attention solely on local windows can limit the Vision Transformer (ViT)'s comprehension of global context. As illustrated in Figure 2.(B), to address this, we shift window boundaries and merge tokens from various windows. For instance, the initial patches w_{03}, w_{12}, w_{21}, and w_{30} originate from distinct windows but will be grouped into a single new window following a window shift operation. Subsequently, we apply multi-head self-attention to this shifted window, a process we call shift window multi-head self-attention (**SW-MHSA**). In the Swin Transformer [14], each fundamental block incorporates both an W-MHSA and an S-MHSA.

C. Image token pruning

A different method to reduce the number of image tokens used involves utilizing a compact machine learning model to predict the relevance of each image patch for the specific task, keeping only the significant tokens. Figure 2.(c) demonstrates that after discarding the unnecessary image patches, the remaining patches participate in the global self-attention mechanism. To reliably predict the importance of each image patch, various studies [13], [31] perform multiple predictions, resulting in a varying sparsity map of image tokens, as illustrated in Figure 2.(C).

IV. Architecture Design

A. Top Architecture

Figure 4 illustrates the principal architecture of **TRFPA**. To support both window attention (SwinT [14]) and pruned global attention (SViT [13]), we have designed two distinct hardware IPs. The first IP, called the Pipeline Window Attention Engine (**P-WAE**), is designed to perform global attention computations using multiple window attention. The second IP is a network on chip (NoC), which is responsible for supporting the window shift operations for image embeddings. Suppose the original image tokens number is **N**, with embedding dimension of **d**. After token selector (**Selector**), only **N'** image tokens embeddings will be write into buffer (implemented

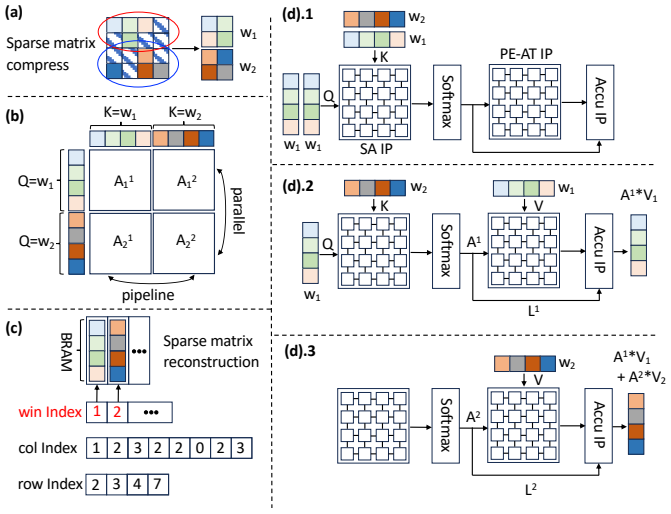


Fig. 5. Tiling sparse attention computation. (a) Token pruning and sparsity compression. (b) Tiling global attention with sub-window attention. (c) Sparse matrix reconstruction. (d) Sparse attention tiling computation on the systolic array (SA) IP and processing elements adder tree (PE-AT) IP.

using BRAM or UltraRAM). Given that several methods for token selection have been introduced [13], [15], [31], this work opts not to concentrate on the Selector's hardware architecture in order to maintain generality. Instead, it emphasizes how to accommodate various sparse patterns of image embedding for attention computation. We assume there are $\mathbf{N}_{head}$ heads and $\mathbf{N}_{win}$ for ViT models. Following the window attention, each window's token embedding will be shifted while preserving the window size at $\mathbf{N}_{win}$.

B. Sparse Attention

To unify window attention and pruned global attention, we tile and load sub-windows of token embeddings within the Pipeline Window Attention Engine to perform MHSA computations. Figure 5.(a) illustrates the compression process for pruned image embeddings. Predicted critical image tokens are grouped into windows, each containing win^2 tokens. In the example shown in Figure 5.(a), we assume that after pruning, there are 8 non-zero token embeddings, resulting in 2 sub-windows after grouping. In Figure 5.(c), we modify the classic compressed sparse row (CSR) format for sparse matrix representation by adding a window index array (win index). The i^{th} element of the win index stores the address of the on-chip BRAM or UltraRAM that holds the i^{th} window's token embeddings. The

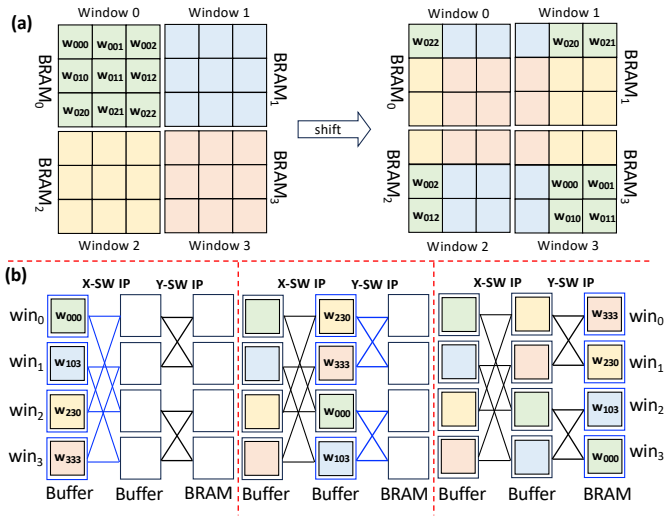


Fig. 6. Here w_{ixy} represents the image token at i^{th} window with global feature coordinate of x (row) and y (column). (a) Partitioning of image feature maps into windows prior to and following the window shift operation. (b) Support for window shift operation utilizing butterfly network on the chip.

win^2 consecutive elements of the column index array represent the original feature map column coordinates of a window's tokens.

Figure 5.(b) presents the sparse-attention tiling process. Each window needs to perform QKV operations with all sub-windows [32]. Suppose there are a total of $\frac{H\times W}{win^2} \times \beta$ tokens, where H and W represent the original feature map's height and width, and β is the pruning rate. The pipeline stage for each window's attention computation is $\frac{H\times W}{win^2} \times \beta$. In the example shown in Figure 5.(b), each sub-window computation has a pipeline stage of 2.

Figure 5.(d) shows the dataflow of the tiling process, using two GEMM engines to support pipeline computation between the attention score A and the sub-attention result $A \cdot V$. The first GEMM engine is a standard output-stationary systolic array (SA IP) [33], and the second is a processing element adder tree architecture (PE-AT IP) [34]. We chose the PE-AT IP for the second stage, considering that in handling the Swin Transformer, the attention score **A** will be masked [14]. The PE-AT IP provides more flexibility to manage sparsity when calculating $A \times V$.

Figures 5.(d).1 to 5.(d).3 present the pipeline process for the tiling computation of sub-window W_1. In Figure 5.(d).1, we perform matrix multiplication (MM) between W_1 and itself. The output result is passed through softmax units [35] to obtain the sub-attention score matrix A_1. In Figure 5.(d).2, we perform MM operations between A_1 and W_1 using the PE-AT IP and between W_1 and W_2 using the SA IP. In Figure 5.d.(3), we perform an MM operation between A_2 and W_2, adding the two sub-attention output embeddings to obtain the final result. Compared to previous works, our tiling attention mechanism improves the throughput and parallelism of sparse attention computation, as demonstrated in the results section.

C. Window shift support

The window shift operation is crucial for the Swin Transformer because local window attention causes the Vision Transformer (ViT) to struggle with capturing global information. Thus, each layer of the Swin Transformer consists of one block for standard window attention (W-MHSA) alongside another for shifted window attention (SW-MHSA). As illustrated in Figure 4, following window attention, the image embeddings proceed through the network-on-chip (NoC) IP. The NoC IP shifts the image token embeddings according to the window shift rule. After shifting, these token embeddings are stored in on-chip BRAM or UltraRAM, ready for the subsequent window attention. Before moving into the NoC architecture, first we list the window shift rule. Suppose for a token with coordinate of (**i**, **x**, **y**) where **i** is the window index, **x** and **y** are the image tokens global coordinate inside the whole feature maps. Suppose the feature map have dimension **H** and **W** and the height and with of the window is **win**. The range of **x** and **y** therefore is [0, **H**-1] and [0, **W**-1]. we will have the correlation between window index and token coordinate:

$$i = \lfloor \frac{x}{win} \rfloor \cdot \lfloor \frac{W}{win} \rfloor + \lfloor \frac{y}{win} \rfloor \tag{5}$$

Following the original swin transformer shift rule [14], we will have after shift operation the new coordinate x_{new} and y_{new} are:

$$x_{new} = (x + H - \lceil \frac{win}{2} \rceil) \% H \tag{6}$$

$$y_{new} = (y + W - \lceil \frac{win}{2} \rceil) \% W \tag{7}$$

Figure 6.(a) illustrates an example of image token shifts occurring within the first window (window 0). There are two findings based on Equation 6 and Equation 7. The first is that for image tokens with coordinates satisfying the condition $x < \lceil \frac{\text{win}}{2} \rceil$ or $y < \lceil \frac{\text{win}}{2} \rceil$, after the shift operation, they will be regrouped with tokens that are not originally connected to them in the feature map. For example, in Figure 6.(a), the token w_{000} will be shifted from window 1 to window 3, and its left and top image tokens do not belong to the original window 0. Therefore, when performing the shifted window attention operation, we need to apply a mask during the attention computation [14]. This is also the primary focus of previous works on FPGA acceleration of the Swin Transformer [17]. The second finding is that any image tokens that has distance non-less than $\lceil \frac{\text{win}}{2} \rceil$ in the original feature maps will be shifted into different windows. Based on these two findings, we propose a butterfly network-on-chip (NoC) IP to support the window shift operation on the fly, as shown in Figure 6.(b). Suppose, in Figure 4, the Pipeline Window Attention Engine supports N_{window} parallelism across different windows. In this case, the NoC IP will have $\lceil \log_2 N_{win} \rceil$ levels. In Figure 6.(b), we assume N_{win} is 4. We first switch tokens w_{000}, w_{103}, w_{230}, and w_{233} since they share the same local coordinate (0,0). Given that each token's distance in the original feature map is at least $\lceil \frac{\text{win}}{2} \rceil$, these four image tokens will shift into different windows. To ensure that during window attention we perform only continuous memory reading, we store each window's token embeddings in independent memory channels (distinct BRAM or UltraRAM). Therefore, these four image tokens will not experience any write or read conflicts.

V. Evaluation

A. Experiment setup

For the machine learning side experiments, we explored four different image token reduction methods, following previous works on image token reduction [13], [14], [22], [23]. These methods include window attention, decoder-based, CNN-based, and multilayer perceptron (MLP)-based approaches. The decoder, CNN, and MLP modules are significantly smaller than the ViT backbone. Specifically, the decoder consists of a single cross-attention layer between the feature map embeddings and the query embeddings. The CNN is a compact policy network [36], while the MLP comprises two fully connected layers. For datasets, we experimented with three object detection and image segmentation datasets: ADE20K [37], Pascal

Context [38], and COCO [39]. The baseline vision transformer model is DeiT-S [1], and for the window attention-based ViT, we used Swin-T [14]. All models traning and inference are implemented with PyTorch [40]. To balance pruning accuracy and acceleration efficiency, we retain all image tokens in the first four layers of DeiT-S and begin dropping tokens starting from the 5^{th} layer.

For the hardware experiments, we synthesized and implemented **TRFPA** on two different FPGA boards using Xilinx Vivado 2020.2. For the edge-level FPGA board, we selected the Xilinx ZCU104, and for the desktop-level FPGA board, we chose the Xilinx Alveo U50. The Alveo U50 board was paired with an Intel Core i9-12900 CPU running at 3.2 GHz as the host. Communication between the FPGA kernel and the host CPU is managed by the Vitis runtime [41]. When accelerating DeiT-S and Swin-T on FPGA, we also apply post-training quantization (PTQ) following PTQ4ViT [42]. Both model weights and activations are quantized to 8-bit fixed-point numbers. Unlike previous works [15], [26] that quantize models to 4 bits, we use 8-bit quantization due to the higher complexity of image segmentation tasks. Applying 4-bit post-training quantization directly to vision transformers leads to a significant drop in segmentation accuracy.

B. ViT accuracy and pruning effects

In Figure 7, we illustrate the balance between various token reduction strategies and computational efforts. We quantify the required multiplications and accumulations (MAC) for ViT models using Giga Floating Point Operations (GFLOPs). In the same figure, we set a pruning rate of 50%, indicating that only half of the image tokens in each ViT encoding layer are utilized in the computation. Consequently, both the MLP layer for QKV embedding generation and the self-attention calculation are reduced. Results in Figure 7 show that eliminating 50% of the image tokens leads to less than a 1% drop in accuracy compared to the original model. This implies that more than half of the image tokens do not significantly contribute to predicting the final bounding box (bbox) and segmentation masks.

In Figure 8.(a-c), we demonstrate the visualization of applying image token pruning at various ratios. Figure 8.(d) illustrates the decrease in accuracy as the token pruning rate increases. As depicted in Figure 8.(a-b) and Figure 8.(d), there is minimal accuracy loss when the token pruning rate is below 60%. However, at a 70% pruning rate, some essential details, like building pixels, are omitted.

C. Resource utilization and Performance

Table I presents the FPGA resource utilization of **TRFPA** on the Xilinx ZCU 104 and Alveo U50. Due to resource limitations on the ZCU 104, the window parallelism (N_{win}) supported by each pipeline window attention engine (P-WAE) is limited to 2. The self-attention and MLP projection computations share the same P-WAE. This requires additional read and write operations from on-chip BRAM for temporary token embedding results in each self-attention layer. On the Alveo U50, with its greater availability of lookup tables (LUTs) and DSPs, each P-WAE IP supports $N_{win} = 8$, allowing the implementation of a complete pipeline between QKV self-attention computations and MLP projection computations. The NoC IP has two levels (2 inputs) on the ZCU 104 and three levels (4 inputs) on the Alveo U50. To conserve DSP resources during the self-attention pipeline, we follow previous work [35], [43] by implementing non-linear layers without resource-intensive exp and div operations. These layers include Softmax, GeLU, and LayerNorm.

Figure 9 shows the acceleration performance of DeiT-S on the Alveo U50 using different pruning ratios and strategies. We report

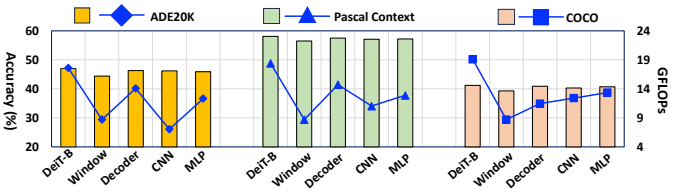


Fig. 7. ViT Models Accuracy and Computation (GFLOPs) Trade-Off with Various Image Token Reduction Techniques.

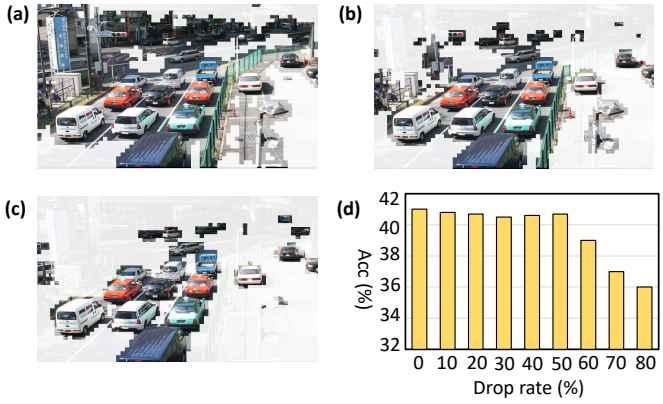


Fig. 8. (a) Image tokens when dropping 20%. (b) Image tokens when dropping 50%. (c) Image tokens when dropping 70%. (d) Model accuracy and drop rate trade off. Here the pruning model that we choose is a two-layer MLP.

the latency of a single ViT layer, taking into account the latency from ViT GEMM operations, data loading (both activations and model weights), and pruning network execution. As illustrated in Figure 9, the 2-layer MLP generally demonstrates higher hardware efficiency compared to CNNs and cross-attention decoders. Given that the 2-layer MLP also achieves decent accuracy, as shown in Figure 7, we adopt it as the pruning method for the cross-platform comparison in Section V-D.

In Figure 10(a), we show the average latency of a single layer across different Swin-T stages. Stage 0 corresponds to the patch generation stage, which primarily involves convolution operations and vector additions (e.g., adding positional embeddings to each image token). The "MHSA" in Figure 10(a) refers to the average latency of both W-MHSA and SW-MHSA. "FFN" represents the feed-forward network, while "Other" includes the latency for data loading (activations and model weights) and window shift operations. Figure 10(b) presents the model parameters for each Swin-T stage. In this setup, each attention head has a token embedding dimension of 32, and each window is of size 7×7. As the stage number increases, the number of windows (N_{win}) decreases, while the number of attention heads (N_{head}) increases.

D. Cross-platforms comparison

In Figure 11, we present a cross-platform comparison of **TRFPA** with other AI computing platforms. To ensure fairness, we compare latency speedup and energy efficiency across desktop-level platforms in Fig 11.(a) and (c). Specifically, our baseline includes NVIDIA V100, A6000, and RTX 4090 GPUs. In Figure 11, we denote DeiT-S with pruning on GPU as GPU-p and Swin-T on GPU as GPU-w. Similarly, we denote DeiT-S with pruning on **TRFPA** as **TRFPA-p**, and Swin-T on **TRFPA** as **TRFPA-w**. We also compare the Alveo U50 version of **TRFPA** with previous FPGA-based ViT accelerators,

TABLE I
RESOURCE UTILIZATION OF TRFPA ON XILINX ZCU 104 AND ALVEO U50

	ZCU104					Alveo U50				
	LUT	FF	DSP	BRAM	UltraRAM	LUT	FF	DSP	BRAM	UltraRAM
P-WAE	129K	24K	1024	32	48	516K	128K	4096	96	480
NoC	41.3K	46.3K	0	4	0	82.3K	94.3K	0	12	0
Others	21.2K	28.4K	0	126	0	32.4K	48.4K	0	126	0
Total	191.5 K	98.7K	1024	162	48	630.7K	270.7K	4096	234	480
Available	230.4 K	460.8K	1728	312	96	872K	1743K	5952	1344	640
Frequency	200 MHz					200 MHz				
Power	15.4 W					27.5 W				

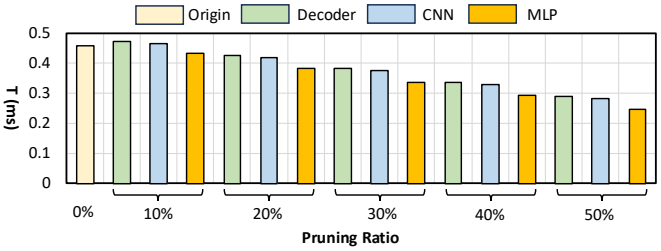
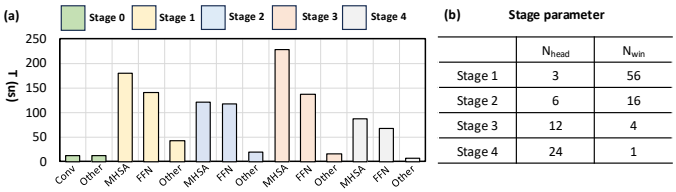


Fig. 9. Acceleration performance of DeiT-S on Alveo U50 across different pruning ratios and image token reduction strategies.



(b) Stage parameter

	N_{head}	N_{win}
Stage 1	3	56
Stage 2	6	16
Stage 3	12	4
Stage 4	24	1

Fig. 10. (a) Execution latency breakdown of Swin-T on Alveo U50. (b) Parameters of each Swin-T stage. All stages use a window size of 7×7, and each attention head has a dimension of 32.

including ViA [24] and SWAT [17]. As shown, **TRFPA** achieves, on average, a 7x speedup compared to ViA and SWAT, thanks to its tiling sparse attention engine and switch IP support for continuous memory reading. Note that for latency comparisons, we restrict the GPU batch size to 1. For example, the RTX 4090 can run Swin-T with a batch size of up to 4 with only a negligible increase in latency. As a result, TRFPA shows little to no throughput advantage. We attribute this to the GPU's high clock frequency and dense SIMD architecture. However, this high throughput comes at the cost of high power consumption, reducing overall energy efficiency. From an energy efficiency perspective, **TRFPA** provides a 5x improvement over GPU platforms, demonstrating its high energy efficiency.

In Figure 11.(b) and (d), we present a comparison between various edge-level accelerators, including the Jetson Nano (Jetson), HeatViT [15], AutoViTAcc [26], TCAS-I [44], and PIVOT [16]. Compared to the Alveo U50 version, the limited LUT and DSP resources on the ZCU 104 restrict the parallelism of **TRFPA**. Additionally, the limited on-chip UltraRAM increases memory communication overhead between the host ARM CPU and kernel FPGA via the AXI interconnect. Despite these constraints, we believe **TRFPA** achieves a good balance between throughput and power efficiency on edge platforms.

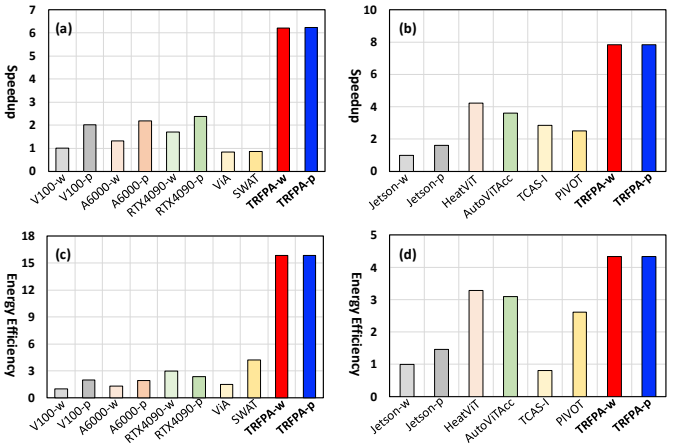


Fig. 11. (a) Speedup comparison at the desktop level. (b) Speedup comparison at the edge level. (c) Energy efficiency improvement at the desktop level. (d) Energy efficiency improvement at the edge level.

VI. CONCLUSION

undIn this paper, we introduce **TRFPA**, a novel FPGA-based vision transformer accelerator that combines window attention and pruned global self-attention to address the computational challenges of ViT models, especially for vision tasks involving large feature maps. TRFPA leverages FPGA's high parallelism and energy efficiency, outperforming previous accelerators with a 7x speedup and 3x energy efficiency gain on desktop platforms. While resource limitations affect performance on edge devices, TRFPA achieves a balanced trade-off between throughput and power efficiency. This work underscores the potential of FPGA acceleration for deploying ViTs in resource-constrained environments, advancing the applicability of high-performance transformers in edge computing.

VII. ACKNOWLEDGEMENT

This work was supported in part by the DARPA Young Faculty Award, the National Science Foundation (NSF) under Grants #2127780, #2319198, #2321840, #2312517, #2235472, and #2431561, the Semiconductor Research Corporation (SRC), the Office of Naval Research through the Young Investigator Program Award, and Grants #N00014-21-1-2225 and #N00014-22-1-2067, Army Research Office Grant #W911NF2410360. Additionally, support was provided by the Air Force Office of Scientific Research under Award #FA9550-22-1-0253, along with generous gifts from Xilinx and Cisco.

REFERENCES

[1] A. Dosovitskiy, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[2] K. Han, , *et al.*, "A survey on vision transformer," *IEEE transactions on pattern analysis and machine intelligence*, vol. 45, no. 1, pp. 87–110, 2022.

[3] S. Ren *et al.*, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 6, pp. 1137–1149, 2016.

[4] H. You *et al.*, "Vitcod: Vision transformer acceleration via dedicated algorithm and accelerator co-design," in *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 273–286, IEEE, 2023.

[5] K. T. Chitty-Venkata *et al.*, "A survey of techniques for optimizing transformer inference," *Journal of Systems Architecture*, p. 102990, 2023.

[6] Y. Li *et al.*, "Q-vit: Accurate and fully quantized low-bit vision transformer," *Advances in neural information processing systems*, vol. 35, pp. 34451–34463, 2022.

[7] F. Yu *et al.*, "Width & depth pruning for vision transformers," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, pp. 3143–3151, 2022.

[8] Z. Yang *et al.*, "Vitkd: Practical guidelines for vit feature knowledge distillation," *arXiv preprint arXiv:2209.02432*, 2022.

[9] E. Jeong *et al.*, "Tensorrt-based framework and optimization methodology for deep learning inference on jetson boards," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 21, no. 5, pp. 1–26, 2022.

[10] P. Tillet *et al.*, "Triton: an intermediate language and compiler for tiled neural network computations," in *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pp. 10–19, 2019.

[11] S. Markidis *et al.*, "Nvidia tensor core programmability, performance & precision," in *2018 IEEE international parallel and distributed processing symposium workshops (IPDPSW)*, pp. 522–531, IEEE, 2018.

[12] S. Wei *et al.*, "Joint token pruning and squeezing towards more aggressive compression of vision transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2092–2101, 2023.

[13] Y. Liu *et al.*, "Revisiting token pruning for object detection and instance segmentation," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 2658–2668, 2024.

[14] Z. Liu *et al.*, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10012–10022, 2021.

[15] P. Dong, , *et al.*, "Heatvit: Hardware-efficient adaptive token pruning for vision transformers," in *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 442–455, IEEE, 2023.

[16] A. Moitra *et al.*, "Pivot-input-aware path selection for energy-efficient vit inference," *arXiv preprint arXiv:2404.15185*, 2024.

[17] Q. Dong *et al.*, "Swat: An efficient swin transformer accelerator based on fpga," in *2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 515–520, IEEE, 2024.

[18] J. Wang *et al.*, "Fpga implementation of object detection accelerator based on vitis-ai," in *2021 11th International Conference on Information Science and Technology (ICIST)*, pp. 571–577, IEEE, 2021.

[19] Y. Tang *et al.*, "A survey on transformer compression," *arXiv preprint arXiv:2402.05964*, 2024.

[20] A. Kirillov *et al.*, "Segment anything," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4015–4026, 2023.

[21] Y. Rao *et al.*, "Dynamicvit: Efficient vision transformers with dynamic token sparsification," *Advances in neural information processing systems*, vol. 34, pp. 13937–13949, 2021.

[22] Q. Tang *et al.*, "Dynamic token pruning in plain vision transformers for semantic segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 777–786, 2023.

[23] C. Lu *et al.*, "Content-aware token sharing for efficient semantic segmentation with vision transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 23631–23640, 2023.

[24] T. 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.

[25] Z. Li, , *et al.*, "Quasar-vit: Hardware-oriented quantization-aware architecture search for vision transformers," in *Proceedings of the 38th ACM International Conference on Supercomputing*, pp. 324–337, 2024.

[26] Z. Li, , *et al.*, "Auto-vit-acc: An fpga-aware automatic acceleration framework for vision transformer with mixed-scheme quantization," in *2022 32nd International Conference on Field-Programmable Logic and Applications (FPL)*, pp. 109–116, IEEE, 2022.

[27] D. Parikh *et al.*, "Accelerating vit inference on fpga through static and dynamic pruning," *arXiv preprint arXiv:2403.14047*, 2024.

[28] R. Sarkar *et al.*, "Edge-moe: Memory-efficient multi-task vision transformer architecture with task-level sparsity via mixture-of-experts," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 01–09, IEEE, 2023.

[29] Y. Han and Q. Liu, "Hpta: A high performance transformer accelerator based on fpga," in *2023 33rd International Conference on Field-Programmable Logic and Applications (FPL)*, pp. 27–33, IEEE, 2023.

[30] S. Lee *et al.*, "Vit-togo: Vision transformer accelerator with grouped token pruning," in *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6, IEEE, 2024.

[31] H. Chen *et al.*, "Vltp: Vision-language guided token pruning for task-oriented segmentation," *arXiv preprint arXiv:2409.08464*, 2024.

[32] T. Dao, "Flashattention-2: Faster attention with better parallelism and work partitioning," *arXiv preprint arXiv:2307.08691*, 2023.

[33] E. Qin *et al.*, "Enabling flexibility for sparse tensor acceleration via heterogeneity," *arXiv preprint arXiv:2201.08916*, 2022.

[34] Y. Liao *et al.*, "A 28nm scalable and flexible accelerator for sparse transformer models," in *Proceedings of the 29th ACM/IEEE International Symposium on Low Power Electronics and Design*, pp. 1–6, 2024.

[35] M. E. Sadeghi *et al.*, "Peano-vit: Power-efficient approximations of nonlinearities in vision transformers," in *Proceedings of the 29th ACM/IEEE International Symposium on Low Power Electronics and Design*, pp. 1–6, 2024.

[36] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*, pp. 6105–6114, PMLR, 2019.

[37] B. Zhou *et al.*, "Scene parsing through ade20k dataset," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 633–641, 2017.

[38] R. Mottaghi *et al.*, "The role of context for object detection and semantic segmentation in the wild," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 891–898, 2014.

[39] H. Caesar *et al.*, "Coco-stuff: Thing and stuff classes in context," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1209–1218, 2018.

[40] A. Paszke, , *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[41] V. Kathail, "Xilinx vitis unified software platform," in *Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 173–174, 2020.

[42] Z. Yuan, C. Xue, Y. Chen, Q. Wu, and G. Sun, "Ptq4vit: Post-training quantization for vision transformers with twin uniform quantization," in *European conference on computer vision*, pp. 191–207, Springer, 2022.

[43] W. Wang, S. Zhou, W. Sun, P. Sun, and Y. Liu, "Sole: Hardware-software co-design of softmax and layernorm for efficient transformer inference," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 1–9, IEEE, 2023.

[44] M. Huang *et al.*, "An integer-only and group-vector systolic accelerator for efficiently mapping vision transformer on edge," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2023.