2023 Design, Automation & Test in Europe Conference (DATE 2023)

RAWAtten: Reconfigurable Accelerator for Window Attention in Hierarchical Vision Transformers

Wantong Li, Yandong Luo, and Shimeng Yu
Georgia Institute of Technology, Atlanta, GA 30332, USA
Email: shimeng.yu@ece.gatech.edu

***Abstract*—After the success of the transformer networks on natural language processing (NLP), the application of transformers to computer vision has followed suit to deliver unprecedented performance gains on vision tasks including image recognition and object detection. The multi-head self-attention (MSA) is the key component in transformers, allowing the models to learn the amount of attention paid to each input position. In particular, hierarchical vision transformers (HVTs) utilize window-based MSA to capture the benefits of the attention mechanism at various scales for further accuracy enhancements. Despite its strong modeling capability, MSA involves complex operations that make transformers prohibitively costly for hardware deployment. Existing hardware accelerators have mainly focused on the MSA workloads in NLP applications, but HVTs involve different parameter dimensions, input sizes, and data reuse opportunities. Therefore, we design the RAWAtten architecture to target the window-based MSA workloads in HVT models. Each w-core in RAWAtten contains near-memory compute engines for linear layers, MAC arrays for intermediate matrix multiplications, and a lightweight reconfigurable softmax. The w-cores can be combined at runtime to perform hierarchical processing to accommodate varying model parameters. Compared to the baseline GPU, RAWAtten at 40nm provides 2.4× average speedup for running the window-MSA workloads in Swin transformer models while consuming only a fraction of GPU power. In addition, RAWAtten achieves 2× area efficiency compared to prior ASIC accelerator for window-MSA.**

Keywords—vision transformer, multi-head self-attention, domain-specific accelerator, reconfigurable architecture, near-memory compute

I. Introduction

In recent years, deep learning has delivered remarkable breakthroughs in fields spanning from natural language processing (NLP) to computer vision (CV). Having witnessed the superior performance of transformer networks [1] over recurrent neural networks (RNNs) to solve sequential tasks, researchers are beginning to apply transformers to CV as well. While convolutional neural networks (CNNs) have long dominated CV [2], attention-based transformer is an attractive strategy to supply additional relationship contexts during the modeling process. Compared to static convolutional filters that are fixed for all contexts once trained, attention allows the dynamic computation of new set of kernels to extract the relational weightings among input positions.

The endeavors to construct vision transformers have met with great success [3]. Out of the vision transformer variants, hierarchical vision transformers (HVTs) [4-7] are promising candidates due to their ability to model self-attention at various granularities. They also avoid quadratic compute complexity to image size by introducing window-based attention. Although HVTs can often surpass CNNs that possess similar number of parameters and operations, their higher computational cost hinders their usages in hardware inference [8]. Multi-head self-attention (MSA), which is a major component in HVT models, accounts for substantial energy consumption and latency, and its complexity calls for specialized accelerators to allow feasible deployment of HVTs.

Existing accelerator solutions [9-13] for MSA have mostly focused on attention in NLP transformer models. However, these accelerators are not optimized to execute window-based MSA workloads, as the model structures of NLP and HVT differ in terms of parameter dimensions, input sizes, and reuse opportunities. Window-MSA workloads exhibit unique properties that challenge the acceleration efforts. 1) While NLP accelerators are designed for long sequences of one-dimensional inputs, accelerators for HVT should optimize for two-dimensional inputs with varying small window sizes. Thus, directly using NLP accelerators for window-MSA workloads can lead to low utilization rate and performance. 2) Each window-MSA workload requires the nonlinear softmax operation, so NLP accelerators that commonly contain only one softmax unit will quickly become bottlenecked as the number of windows increases. 3) HVT models present different reuse opportunities across the various stages of window-MSA blocks. In earlier stages, weight reuse opportunities are abundant because multiple small windows share the same set of weights. For later stages, input reuse becomes more common as the window sizes grow yet number of windows decreases.

In this work, we propose the RAWAtten architecture to tackle the aforementioned design challenges. RAWAtten contains parallel processing cores, coined w-cores, to execute window-MSA workloads. Each w-core is equipped with near-memory compute (NMC) engines for linear layers, multiply-and-accumulate (MAC) arrays for intermediate matrix multiplications (MatMul), and a novel softmax module. RAWAtten is reconfigurable to accommodate different HVT model parameters, while providing high data reuse and high utilization rate. The main contributions of this work are as follows:

- We propose the RAWAtten architecture to accelerate the complex window-MSA workloads in HVT models. The compute units in each w-core are reconfigurable to maintain high utilization rates across model parameters.
- We design a lightweight reconfigurable softmax (LR-Softmax) module and assign one unit per w-core to solve the bottleneck problem. Through rearrangement of the softmax equation, we eliminate the expensive exponential and division operations. An estimation strategy for the remaining terms brings significant area savings with negligible model accuracy loss.
- RAWAtten is evaluated on the popular Swin transformers [7]. Results show that RAWAtten achieves 2.4× speedup

978-3-9819263-7-8/DATE23/© 2023 EDAA

over the baseline GPU while only consuming 0.6% of GPU power. Compared to prior accelerator for vision transformer [13], RAWAtten also improves the area efficiency by 2×.

II. Background

A. Vision Transformer Models

Transformer Networks for Vision. The transformer networks are sequence transduction models involving the attention mechanism [1]. They were first introduced to solve machine translation tasks and have since achieved success on various NLP tasks. A typical NLP transformer adopts the encoder-decoder architecture. Stacked encoders map the input sequence of symbol representations into continuous representations, which are then transformed into output sequence of symbols by the stacked decoders. More recently, transformer networks have been extended to solve computer vision tasks. The Vision Transformer (ViT) [3] is a pioneering work that proposes to interpret input images as a sequence of patches and process them by a standard transformer encoder. ViT is able to outperform ResNets [2] on image recognition tasks when pre-trained on larger datasets. However, the global attention used in ViT renders this model intractable to process high-resolution images, since the computational cost of global attention scales quadratically to image size. Hence, hierarchical vision transformers that employ window-based attention mechanism have been investigated to lower the computational complexity of vision transformers. As shown in Fig. 1 (a), HVTs break down an input feature map into multiple windows for window-based local attention. The window sizes may change across the stages of HVTs, allowing feature extractions at different scales. In each stage, an encoder processes the input feature map in each window, with an example encoder shown in Fig. 1 (b).

The Pyramid Vision Transformer (PVT) [4] takes patches of the input image as inputs. It follows a pyramid model structure that progressively shrinks the attention window sizes through the stages, as opposed to the columnar structure of ViT. The Focal Transformer [5] introduces focal self-attention to simultaneously capture global and local relationships. Each window attends to its closest surrounding windows with fine-grained local attention and far windows with coarse-grained global attention. The Twins-SVT [6] can similarly capture local attention by interleaving locally-grouped attention and global sub-sampled attention in every encoder. To enlarge the receptive field size, the Swin Transformer [7] proposes shifted-window operations to allow both non-overlapping local windows and cross-window connections. At the time of this writing, the Swin transformer backbones have achieved unprecedented accuracy for image recognition [14] and object detection [15] tasks. Therefore, we focus on the Swin transformer when evaluating the proposed RAWAtten architecture. Although the encoder structures may differ among HVT models, RAWAtten is generally compatible since all HVTs use window-based attention mechanism with similar dimensional parameters.

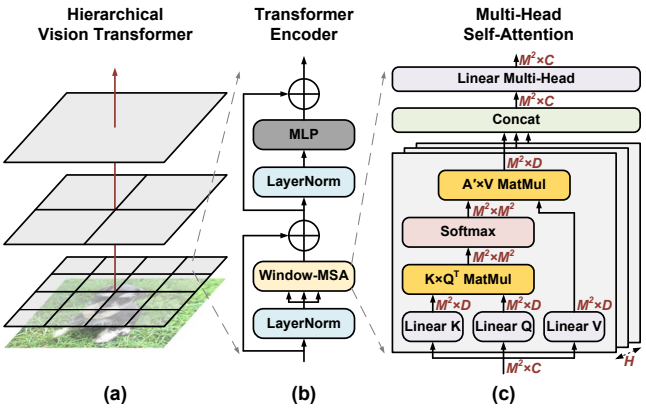

Fig. 1. (a) Schematic of a hierarchical vision transformer. (b) An example of transformer encoder. (c) Multi-head self-attention module with annotated model parameters.

Multi-Head Self-Attention. Transformers utilize the self-attention mechanism to identify the pairwise relations within the input elements. Through the training process, a transformer network learns how much attention to pay to each input region, and these attention weights are stored in the Key (W_K), Query (W_Q), and Value (W_V) matrices. Each head represents a different subspace projection to capture distinctive attentional dependencies.

Fig. 1 (c) describes the operations in the window-based MSA of HVT models. The model parameters of window area, channel depth, head size, and number of heads are denoted by M^2, C, D, and H respectively. Typically D is set to C/H. An input matrix of dimension $M^2 \times C$ is linearly projected by each head through dense multiplications with the W_K, W_Q, and W_V matrices, each of dimension $C \times D$. After the linear transformations, the attention score matrix A is calculated through the $K \times Q^T$ multiplication. The matrix A has dimension $M^2 \times M^2$, so the compute complexity of intermediate MatMul may be reduced when the window size is kept small. Softmax is used on A to obtain A′, in which the highly-weighted scores are further enlarged whereas other scores are suppressed. A′ is then multiplied with V to assign attention weightings to the input features. Multiple heads are finally concatenated, and the combined result that reverts to the original input dimension goes through a multi-head linear layer.

Fig. 2 shows that window-MSA accounts for 30% to 40% of the total workloads in Swin transformer models, further motivating the need for specialized hardware accelerator. Note that Swin-T and Swin-S models only differ by the number of third stages. One crucial observation we make is that unlike NLP transformers whose intermediate MatMul often dominates the MSA computation, HVTs involve substantially more computations for linear layers than the intermediate MatMul. This observation will be used to drive our design decisions as discussed in Section III.C.

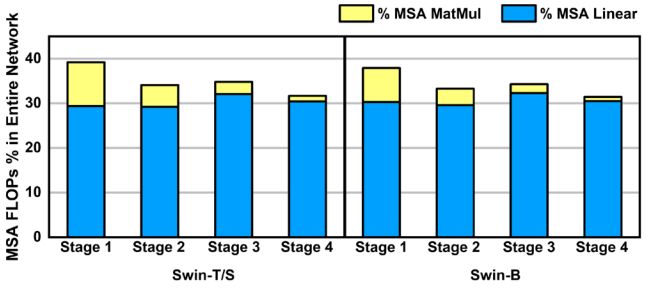

Fig. 2. Distribution of MSA operations in various stages of Swin-T/S and Swin-B workloads.

B. Hardware Accelerators for Transformers

Prior transformer accelerators have largely targeted the attention block in NLP workloads. SpAtten [9] is a hardware-software co-design effort to exploit the quantization and sparsity opportunities in NLP attention. Pruned tokens and heads are selected on the fly to reduce computation and memory access for the attention. DTQAtten [10] proposes to dynamically quantize the NLP attention heads with different precisions according to their importance scores, decreasing the computational complexity. A variable-speed systolic array is designed to alleviate the pipeline stall problem arising from the mixed-precision quantization. In iMCAT [11], SRAM crossbar arrays and ternary content addressable memories are leveraged to provide speed and energy improvements for processing long input sequences. ReTransformer [12] explores ReRAM-based processing-in-memory for NLP transformers to harness the high density and low power properties of emerging non-volatile devices. The Row-wise accelerator [13] is the only existing ASIC solution for vision transformers. It uses row-wise scheduling to decompose transformer operations into single dot product primitives for unified execution. Model weights are broadcast for data reuse.

III. Proposed Architecture

A. Architecture Overview

An overview of the RAWAtten architecture is shown in Fig. 3. The accelerator consists of N w-cores that parallelly execute the window-MSA workloads in HVTs. Each w-core is capable of processing all computation types in window-MSA, including linear layers, intermediate MatMul, and softmax. They can process window workloads independently or combined as a group by connecting their compute modules with those of other neighboring w-cores.

The global input buffer stores the transformer model activations fetched from DRAM. The input dispatcher routes these activations to each w-core, and it supports unicast, multicast, and broadcast to offer design reconfigurability. Activations from the input dispatcher serve as the inputs to the SRAM-based NMC engines, which both store the model weights and execute the linear K, Q, and V layers. MAC array is utilized for the intermediate MatMul steps and follows a weight-stationary approach. For the MatMul, the linear Q and V results are first computed together as they share the same SRAM subarray, and the results are stored to the local buffers of $K \times Q^T$ and $A' \times V$ MAC arrays respectively as weights. Then, linear K results arrive from NMC as inputs to the $K \times Q^T$ MAC array. Multiple cycles are necessary to complete the MatMul steps due to the sequential nature of NMC, so an accumulator is used to add the partial sums from the MAC array. This accumulator can be reconfigured to connect to accumulators of adjacent w-cores to perform hierarchical accumulation. Next, the accumulated partial sums enter the LR-Softmax module row-wise to compute the attention map A'. The LR-Softmax can also share data with adjacent LR-Softmax modules to accommodate varying network parameters. The A' vector multiplies with the preloaded linear V results in the $A' \times V$ MAC array, and another reconfigurable accumulator is used for the MAC partial sums. A final NMC engine computes the multi-head linear layer, and the results from each w-core are accumulated by the global accumulator. Lastly, the global output buffer transfers the results back to DRAM.

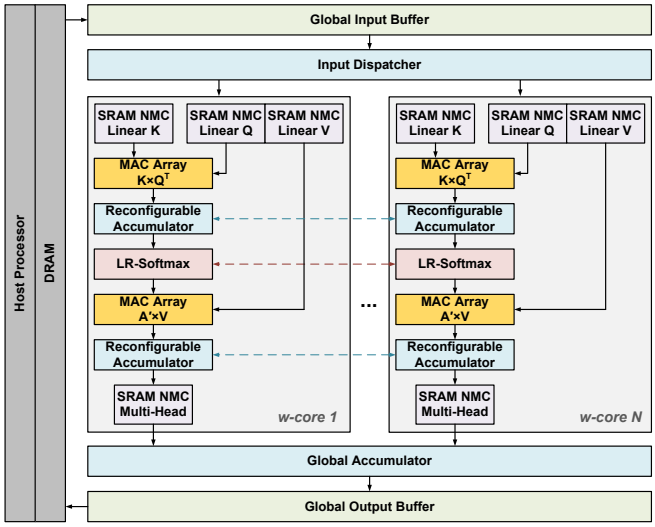

Fig. 3. The proposed RAWAtten architecture. The reconfigurable accumulators and LR-Softmax in each w-core can be connected to those of adjacent w-cores for hierarchical processing.

B. Design for Reconfigurability

To accommodate the varying HVT model parameters including M^2, C, D, and H, the RAWAtten accelerator is likewise parameterized with m^2, c, d, and N. Table I summarizes the hardware parameters used in RAWAtten and which hardware dimension each parameter relates to, along with the allocation of global and local buffers. Table II shows the progression of model parameters across the 4 stages of Swin-T and Swin-S for reference. Input size is in the dimension of height × width × channel depth (C).

TABLE I. Hardware Parameters of RAWAtten

RAWAtten Parameters		
Parameter	Value	Corresponding Hardware Dimension
N	32	# w-cores in RAWAtten
m^2	16	# 8b multipliers in one MAC array
c	32	# rows in one NMC subarray
d	16	# 8b weights in one row of NMC

Buffer Allocation		
Buffer	Value	Type
Total NMC capacity for weights	64 KB	SRAM
Global input buffer	64 KB	SRAM
Global output buffer	64 KB	SRAM
Total local input buffer	1 KB	Register
Total local weight buffer	16 KB	Register
Total local psum buffer	4 KB	Register

TABLE II. Model Parameters of Swin-T/Swin-S

	Stage 1	Stage 2	Stage 3	Stage 4
Input size	64×64×96	32×32×192	16×16×384	8×8×768
H	3	6	12	24
M^2	64	64	64	64
# windows	64	16	4	1

Because multiple w-cores can be combined to process larger model dimensions, the hardware parameters per w-core are minimized such that high compute utilization rate may be maintained regardless of the HVT models and stages. We define utilization rate as the ratio between the utilized number of compute units during peak inference and the total number of available compute units. To combine d into D and c into C, the model weights are split into the NMC engines of different w-cores. Intermediate results are accumulated by the local reconfigurable accumulators. Combining m^2 into M^2 can be done in the same w-core, as activations from the global input buffer are split into different segments and the intermediate results are accumulated by the global accumulator. After accommodating the model parameters for each head, multiple heads H may be processed in parallel by different w-cores. Since HVTs offer more weight reuse in earlier stages and more input reuse in later stages, we assign the same amount of weight, input, and output buffers to balance the reuse opportunities throughout the model stages.

C. Compute Units

Near-Memory Compute. As shown in Fig. 4 (a), the SRAM-based NMC engines receive activations from the input dispatcher as inputs, which are decoded to sequentially activate the SRAM wordlines (WLs) in a bit-serial fashion. This step acts as the multiplication between activations and weights. The K subarray holds h 8b weights in each row, whereas the Q/V subarray holds $2h$ 8b weights in each row as the Q/V cells share the same WLs for simultaneous activation. Each subarray has c rows, and each column is equipped with a sense amplifier so all multiplication results from the same row are read out at once. Shift-add and accumulate are used to complete the MAC computations in linear layers.

Since the linear layers dominate the window-MSA computation, SRAM is chosen over registers due to higher density, and we employ NMC to avoid excessively reading model parameters and storing them into local buffers before starting the computation. Therefore, weight reuse becomes more favorable as the loaded weights do not require further data movements. In addition, the sequential nature of NMC helps ease the routing complexity between the input dispatcher and w-cores, thus allowing more parallel w-cores to be deployed.

MAC Array. Each MAC array consists of m^2 8b-multipliers and an adder tree, as shown in Fig. 4 (b). As a weight stationary design, the weights are stored in the local weight buffers. Every cycle an 8b input is read from the local input buffers to perform MAC operation with m^2 8b weights. The results either go through individual or hierarchical accumulation, depending on the hardware configuration.

If SRAM-based NMC were chosen to also compute intermediate MatMul, a transposable SRAM array would be needed for Q^T, thus complicating the design. Instead, MAC array can naturally fulfill the transpose requirement through routing. Because computing the intermediate MatMul involves frequent buffer accesses, we use register-based local buffers in the MAC array to help lower access energy compared to SRAM buffers. The larger area of registers is compensated by the relatively small fractions and dimensions of intermediate MatMul operations in HVTs. Furthermore,

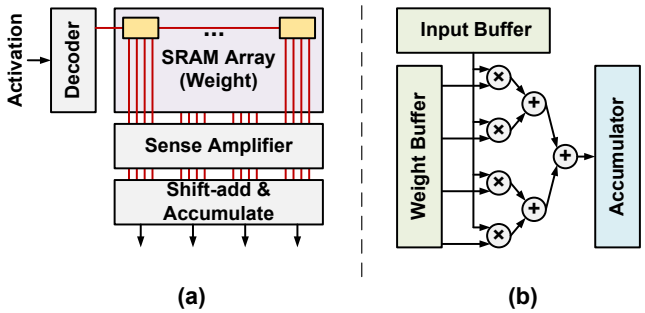

Fig. 4. (a) SRAM-based near-memory compute unit. (b) MAC array for intermediate MatMul operation.

introducing a second MAC array to each w-core for A′×V MatMul can prevent the intermediate A′ results from being first written back to global buffers, which would require higher access energy than local buffers.

LR-Softmax. Softmax is a complex nonlinear function that contains exponential and division operations. Due to the typically long input sequences in NLP transformers, allocating one softmax unit per NLP hardware accelerator may be justifiable. However, HVTs gain performance benefits by relying on small window sizes, each of which requires softmax calculations. For instance, stage 1 of Swin-S consists of 64 windows for 256×256 input image size, and the number of windows grows with input size. Therefore, assigning one softmax unit per w-core is advantageous for window-based processing in RAWAtten, as parallelization and scalability of the w-cores can be improved.

The cost of standard softmax implementation is prohibitively high for having one unit per w-core, so we design LR-Softmax that involves the following rearrangements of the softmax equation to decrease hardware design complexity. The first transformation we apply is the log-sum-exp trick [16], which removes the division operation.

$$Softmax(\boldsymbol{x})_i = \frac{exp(x_i)}{\sum_j exp(x_j)} = exp(x_i - ln\sum_j exp(x_j)) \quad (1)$$

During the integer quantization of softmax, logarithmic quantization [17] is applied instead of linear quantization to remove the exponential operation at no accuracy loss.

$$Q_{ln}[exp(x_i - ln\sum_j exp(x_j))] = x_i - ln\sum_j exp(x_j) \quad (2)$$

Consequently, the softmax function is reduced to the subtraction between an input element x_i and the log-sum-exp (LSE) term that is dependent on $\mathbf{x}$. We observe that the LSE term is lower-bounded by the maximum element x_{max}, and can be rewritten as the addition between x_{max} and an error term ε.

$$ln\sum_j exp(x_j) = x_{max} + \varepsilon > x_{max} \quad (3)$$

We obtain an approximate ε by estimating the individual contribution of each non-maximum element in $\mathbf{x}$ to ε. The contribution of each x_j is estimated through the difference between x_{max} and x_j. This technique works for input vectors of any distribution owing to the shift-invariance property of softmax, where a shift term δ to the input vector does not affect the outcome.

$$x_i - ln\sum_j exp(x_j) = x_i + \delta - ln\sum_j exp(x_j + \delta) \quad (4)$$

Fig. 5 illustrates the hardware implementation of LR-Softmax. The x_{max} in the input vector **x** is searched through shift-registers and comparator. Quantizing the difference between x_{max} and every other element in **x** accounts for the estimated contribution of each non-maximum element to the error ε. The quantized difference values are stored in a lookup table (LUT), and the number of LUT entries can be reduced as discussed in Section IV.B. Finally, **x** subtracts the x_{max} + ε for the attention map A′. To achieve reconfigurability, the shift-registers of LR-Softmax can connect to those of adjacent w-cores and find the global maximum, thus allowing the x_{max} + ε estimation across the input vectors combined from multiple w-cores.

IV. Evaluation

A. Evaluation Setup

For energy and area evaluations, we implement the proposed RAWAtten in SystemVerilog RTL. The design is synthesized with TSMC 40nm ULP library using Cadence Genus, with a target frequency of 1 GHz. With representative window-MSA workloads from Swin-T and Swin-S, we estimate the design power consumption through Synopsys PrimeTime PX. Global buffers and NMC SRAM subarrays are evaluated using DNN+NeuroSim [18]. A LPDDR4 [19] with bandwidth of 34 GB/s is included in our analysis to consider DRAM access cost.

The RAWAtten architecture is evaluated on the window-MSA workloads in Swin-T and Swin-S models, targeting image recognition on the ImageNet-1K dataset with 256×256 image size. Post-training quantization [17] is applied to quantize all model parameters to INT8, resulting in accuracy loss of 0.89% for Swin-T and 0.53% for Swin-S. We use the accuracy after quantization as the baseline for later analysis.

For baseline platform comparison, the investigated programs written in PyTorch are run on NVIDIA Titan V GPU with CUDA 11.3. We utilize the *nivida-smi* tool for measuring GPU power and resource utilization. Latency is measured through *torch.cuda.Event*. All workloads on GPU are also quantized to INT8 for fair comparison with RAWAtten. We assume that the HVT workloads not run on RAWAtten are run on the baseline GPU.

B. Network Accuracy Analysis

In the design of LR-Softmax, the number of LUT entries may be reduced because input elements that are far away from x_{max} contribute trivially to the estimation of ε in LSE. Therefore, we can safely ignore their contributions by setting

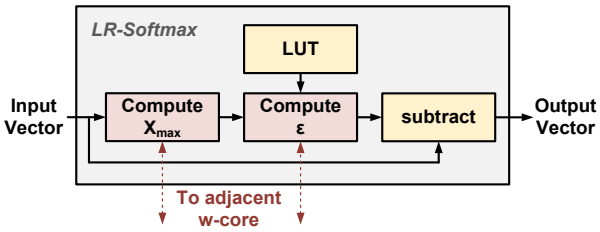

Fig. 5. Hardware implementation of the LR-Softmax module.

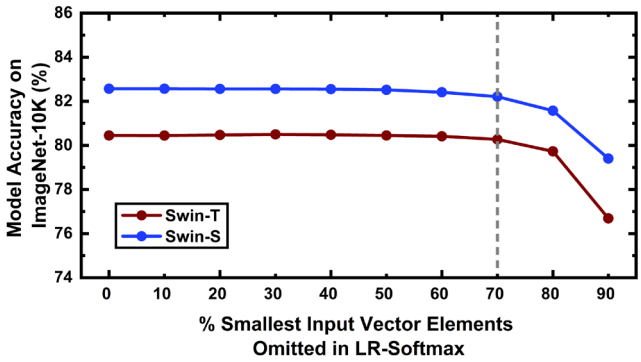

Fig. 6. Effect on model accuracy by omitting the smallest input vector elements during LR-Softmax computation.

them to zero. This estimate is enabled by another property of softmax, that large elements in the input vector are further amplified but smaller elements are more diminished. Fig. 6 plots the Swin-T and Swin-S model accuracies as a function of percentage of smallest input elements omitted during LR-Softmax calculation. We observe that the knee for the plots is located at around 80% omitted, so we set the % omitted elements to 70%, where Swin-T and Swin-S experience only 0.18% and 0.35% accuracy loss respectively. This optimization leads to 53% area reduction for each LR-Softmax unit, making the allocation of one unit per w-core more economical.

C. Energy and Compute Efficiencies

RAWAtten is evaluated to consume an average power of 0.66 W with area of 1.64 mm^2. Fig. 7 shows the power and area breakdowns of RAWAtten. With access energy of 1.3 pJ/bit [19], the DRAM data transfer dominates power consumption at 62.4% and is not shown in the breakdown. The majority of the remaining power is consumed by the compute units, followed by global buffer access. Similarly, NMC engines (that store model weights) and MAC arrays together occupy 70.6% area, whereas the global input and output buffers take 18.5% area. The area overhead to support the reconfigurable compute units is 5.5% for each w-core. By using the calibrated transistor parameter trending in DNN+NeuroSim, we further scale the RAWAtten to 7nm process to showcase the design's potential at an advanced technology node. At 7nm, RAWAtten is projected to have an area of 0.1 mm^2 and power consumption of 73 mW at iso-frequency, showing 16× and 9× improvements in area and power compared to the 40nm version.

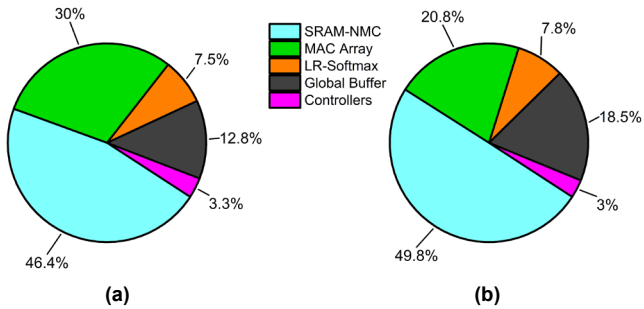

Fig. 7. (a) Power and (b) area breakdowns of the implemented RAWAtten design. DRAM access is not shown in the power breakdown.

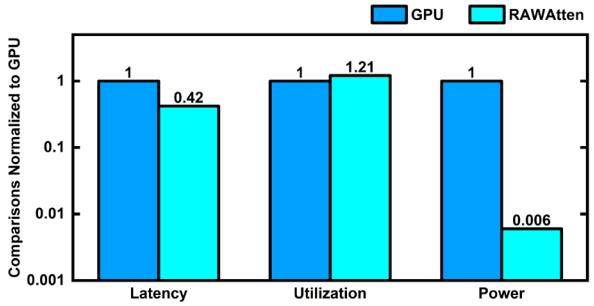

Fig. 8. Comparisons between GPU and RAWAtten.

GPU can suffer from nonlinear functions, matrix transpose, and unoptimized memory access patterns when executing attention workloads [9]. Fig. 8 shows the comparisons in average latency, utilization rate, and power consumption, normalized to the baseline GPU. We measure that the GPU can reach average compute utilizations of 71% for Swin-T and 78% for Swin-S. In contrast, RAWAtten achieves better utilizations at 94% for Swin-T and 86% for Swin-S, thanks to the minimal hardware parameters. For latency and power, we measure that the GPU completes Swin-T workloads in 6.94 ms and Swin-S workloads in 11.45 ms at 110 W, whereas RAWAtten completes the same workloads in 2.67 ms and 5.23 ms at 0.66 W. Table III presents the comparison of RAWAtten with other state-of-the-art MSA accelerators designed at the same technology node. For the Row-wise accelerator, the reported area excludes output buffers and softmax module, and the energy consumption is not reported. Even when normalized to the same frequency, RAWAtten outperforms the Row-wise accelerator in area efficiency partly due to the second MAC array in each w-core for A′×V multiplication, which eliminates the store and load time of intermediate data to global buffers. Note that SpAtten and DTQAtten target and are evaluated on NLP workloads. The high energy and area efficiencies of DTQAtten are derived from dynamic quantization of 8b input tokens into 4b or even 0b (pruned) precision. With 8b computations throughout, RAWAtten achieves competitive energy efficiency stemming from balanced data reuse and reduced on-chip data movements.

TABLE III. COMPARISON TABLE

	This Work	Row-wise [13]	SpAtten [9]	DTQAtten [10]
Workload	Vision	Vision	NLP	NLP
Node (nm)	40	40	40	40
Frequency (GHz)	1.0	0.6	1.0	1.0
Area (mm^2)	1.64	1.71 *	1.55	1.41
Throughput (GOPS)	768	403	360	953
Energy Effi. (GOP/j)	1170	Not reported	382	1298
Area Effi. (GOPS/mm^2)	469	236 *	238	678

* Estimated from reported area in KGE.

V. CONCLUSION

In this work, we demonstrate the RAWAtten architecture aimed to accelerate the window-based MSA workloads in hierarchical vision transformers. RAWAtten contains reconfigurable compute units to accommodate various model parameters with high utilization rate. With an area of 1.64 mm^2 and power consumption of 0.66 W in 40nm node, the implemented design can achieve 2.4× speedup over the baseline GPU while consuming only a fraction of GPU power. Our work improves upon the prior work [13] on ASIC implementation of vision transformer with 1.9× throughput and 2× area efficiency.

ACKNOWLEDGMENT

This work is supported in part by PRISM, one of the DARPA / JUMP 2.0 centers.

REFERENCES

[1] A. Vaswani *et al.*, "Attention is all you need," *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[2] K. He *et al.*, "Deep residual learning for image recognition," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[3] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv*, 2020.

[4] W. Wang *et al.*, "Pyramid vision transformer: A versatile backbone for dense prediction without convolutions," *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[5] J. Yang *et al.*, "Focal self-attention for local-global interactions in vision transformers," *arXiv*, 2021.

[6] X. Chu *et al.*, "Twins: Revisiting the design of spatial attention in vision transformers," *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[7] Z. Liu *et al.*, "Swin transformer: Hierarchical vision transformer using shifted windows," *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[8] X. Wang *et al.*, "Towards efficient vision transformer inference: A first study of transformers on mobile devices," *ACM International Workshop on Mobile Computing Systems and Applications (HotMobile)*, 2022.

[9] H. Wang *et al.*, "SpAtten: Efficient sparse attention architecture with cascade token and head pruning," *IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2021.

[10] T. Yang *et al.*, "DTQAtten: Leveraging dynamic token-based quantization for efficient attention architecture," *IEEE Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2022.

[11] A. F. Laguna *et al.*, "In-memory computing based accelerator for transformer networks for long sequences," *IEEE Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021.

[12] X. Yang *et al.*, "ReTransformer: ReRAM-based processing-in-memory architecture for transformer acceleration," *ACM International Conference on Computer-Aided Design (ICCAD)*, 2020.

[13] H.-Y. Wang and T.-S. Chang, "Row-wise accelerator for vision transformer," *IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2022.

[14] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision (IJCV)*, 2015.

[15] T.-Y. Lin *et al.*, "Microsoft COCO: Common objects in context," *European Conference on Computer Vision (ECCV)*, 2014.

[16] B. Yuan, "Efficient hardware architecture of softmax layer in deep neural network," *IEEE International System-on-Chip Conference (SOCC)*, 2016.

[17] Y. Lin *et al.*, "FQ-ViT: Post-training quantization for fully quantized vision transformer," *International Joint Conference on Artificial Intelligence (IJCAI)*, 2022.

[18] X. Peng *et al.*, "DNN+NeuroSim: An end-to-end benchmarking framework for compute-in-memory accelerators with versatile device technologies," *IEEE International Electron Devices Meeting (IEDM)*, 2019.

[19] D. Skinner, "LPDDR4 moves mobile," *JEDEC Mobile Forum Conference*, 2013.