ViTA: A Highly Efficient Dataflow and Architecture for Vision Transformers

Chunyun Chen, Lantian Li and Mohamed M. Sabry Aly
Nanyang Technological University, Singapore
Email: {chunyun001, lantian001, msabry}@ntu.edu.sg

***Abstract*—Transformer-based DNNs have dominated several AI fields with remarkable performance. However, the scaling up of Transformer models up to trillions of parameters and computation operations has made them both computationally and data-intensive. This poses a significant challenge to utilizing Transformer models, *e.g.*, in the area- and power-constrained systems. This paper introduces ViTA, an efficient architecture for accelerating the entire workload of Vision Transformer (ViT), targeting enhanced area and power efficiency. ViTA adopts a novel memory-centric dataflow that reduces memory usage and data movement, exploiting computational parallelism and locality. This design results in a 76.71% reduction in memory requirements for Multi-Head Self Attention (MHA) compared to original dataflows with VGA resolution images. A fused configurable module is designed for supporting non-linear functions in ViT workloads, such as GELU, Softmax, and LayerNorm, optimizing hardware resource usage. Our results show that ViTA achieves 16.384 TOPS with area and power efficiencies of 2.13 TOPS/mm^2 and 1.57 TOPS/W at 1 GHz, surpassing current Transformer accelerators by 27.85× and 1.40×, respectively.**

***Index Terms*—ASIC, Transformer, Vision Transformer, GELU, Softmax, LayerNorm**

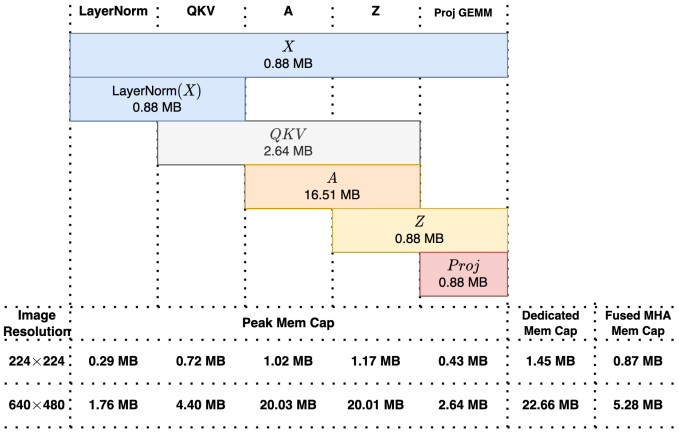


Image Resolution	Peak Mem Cap					Dedicated Mem Cap	Fused MHA Mem Cap
224×224	0.29 MB	0.72 MB	1.02 MB	1.17 MB	0.43 MB	1.45 MB	0.87 MB
640×480	1.76 MB	4.40 MB	20.03 MB	20.01 MB	2.64 MB	22.66 MB	5.28 MB

Fig. 1. The memory capability requirements of the MHA layer in the ViT-Base models, with a 224 × 224 and VGA resolution RGB image as input in INT8 mode. As illustrated, to support VGA images, 22.66 MB and 20.03 MB memory is required if dedicated memories are used for middle results or memories are reused respectively.

I. Introduction

Transformer-based architectures have superior performance in both Natural Language Processing (NLP) and Computer Vision (CV) applications, demonstrating state-of-the-art (SOTA) performances in numerous tasks ranging from machine translation [1] to image classification [2]–[4], employing the attention mechanism. GPT-4 [5], as an illustration, is powerful enough for multitasking across both NLP and CV fields.

However, as the scale-up of these models — especially models like GPT-4 with trillions of parameters — they become compute- and data-intensive, challenging their efficient deployment on both cloud and edge devices. Operations in Transformer models are matrix multiplications and non-linear functions. Matrix multiplication consumes over 60% of the execution time of Vision Transformer (ViT) [6] on NVIDIA 2080 Ti GPU [7]. This operation has seen significant improvements due to optimizations in algorithmic, *e.g.* quantization [2], [8], [9], sparsity [10] and function approximation [11], [12] and scalable ASIC [11]–[15] and FPGA-based accelerators [16], [17]. Non-linear functions in Transformer models, *e.g.*, GELU, Layer Normalization, and Softmax, occupy nearly 40% and 60% of the execution time and memory usage respectively [7] and occupy 45% of the total area in SwiftTron [15] even if the matrix multipliers are dedicated. However, they have not witnessed significant improvement and remain bottlenecks due to their computational complexity and high throughput requirements. Moreover, the computation dataflow is less explored, and many Transformer accelerators [15], [16] simply follow the original software implementation dataflow, which can be further optimized to reduce both memory footprint and data movement. As shown in Fig. 1, the memory requirements of the Multi-Head Self Attention (MHA) layer in the ViT-Base models to support 224×224 and VGA image resolutions are 1.45 MB and 22.66 MB, respectively.

Addressing these concerns, this paper introduces ViTA – a scalable architecture with a highly efficient memory-centric dataflow for ViT – to accelerate the entire ViT workload with high area efficiency and power efficiency. Our results show that ViTA achieves 16.384 TOPS with an area efficiency of 2.13 TOPS/mm^2 and a power efficiency of 1.57 TOPS/W at 1 GHz targeting 28 nm node, surpassing SOTA Transformer accelerators by factors of 27.85× and 1.40×, respectively.

Our contributions are summarized as follows:

- A scalable architecture, ViTA, for the entire ViT workload, exploiting a memory-centric, hardware-efficient dataflow to reduce memory usage and data transfer.
- A novel fused special function module for non-linear functions in ViT models, optimizing resource sharing and

This research is supported in part by the NRF AME programmatic fund titled Hardware-Software Co-optimisation for Deep Learning (A1892b0026).

979-8-3503-4859-0/DATE24/© 2024 EDAA

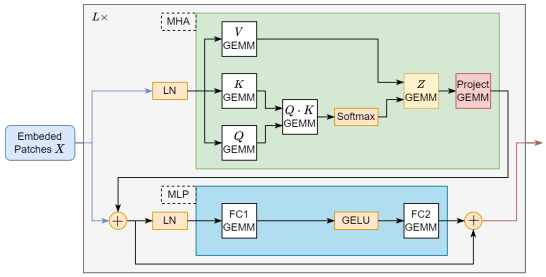


Fig. 2. The Transformer encoder, adopted from [8].

elevating area and power metrics.

- A comprehensive design space exploration of ViTA Kernels and VMUs, enabling area and power trade-offs.
- A performance analysis between ViTA targeting 28 nm node and other SOTA Transformer accelerators.

The structure of this paper is as follows: Section II reviews related work. Sections IV and III discuss ViTA dataflows and architecture, respectively. Section V offers performance evaluations. Conclusions are presented in Section VI.

II. Background and Related Work

A. Transformer Models

Introduced by Vaswani *et al.* [1], Transformers consist of encoders (Fig. 2) and decoders using Multi-Head Self Attention (MHA) and Feed-Forward Networks (FFN). ViT [6] extends Transformer principles to vision by treating image patches as tokens. Variants, *e.g.*, DeiT [3], PVT [18], and Swin [4] further explored this approach.

B. Accelerations of Entire Transformers

Transformer accelerating on expensive and high-power GPUs is well explored. I-BERT [9] and I-ViT [2] employ integer-only inferences to cut costs, while NVIDIA's FasterTransformer [8] offers two quantization modes, where the input and output of GEMM operations are INT8 and INT32/INT8 respectively, while other non-linear functions are FP32. FlashAttention [19] refines memory access. We adopt GEMM quantizations from [8] and extend the non-linear functions to INT32.

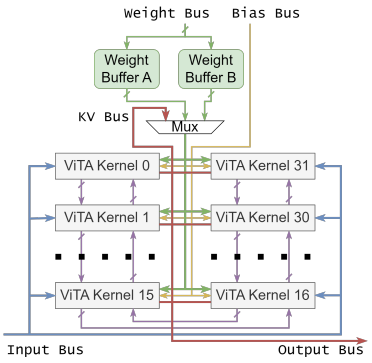
SwiftTron [15] and Nag *et al.*'s architecture [16] directly mapped Transformer models to hardware, leading to large chip size, high power, and low operation frequency. NN-LUT [20] employs Lookup Tables (LUT) while Wang *et al.* [11], [12] emphasize energy efficiency. However, they require re-training, which is not flexible and time-consuming for large models.

C. Accelerations of Individual Operations in Transformers

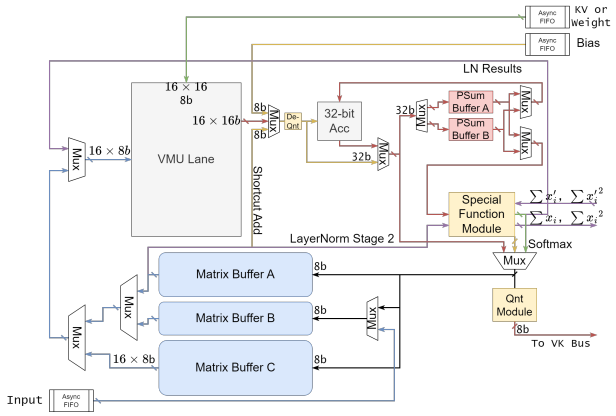
1) Acceleration on MHA Layer: A^3 [13] and ELSA [14] utilize approximated MHA to reduce the number of operations.

2) Acceleration on Non-linear Functions:

- **GELU**: Introduced in [21], GELU has several approximations, including those based on the Tanh and Sigmoid functions [2], [9], [20], [22].
- **Softmax**: Softmax is integral for predicting class probabilities. Hardware acceleration methods encompass direct optimization [20], [23], [24], mathematical transformations [23], and mathematical reformulations [25].
- **LayerNorm**: LayerNorm (LN) is critical in Transformer models. Techniques like parallel variance computation on GPUs [26](Eq. 1), approximation for the square root function [2], [9], and approximation for the reciprocal square root function [20] aid in improving its efficiency.



(a) ViTA Top



(b) ViTA Kernel

Fig. 3. ViTA hardware architecture. (a) An overview highlighting the buses: input (blue), weight (green), output/multiplexed Key and Value (red), and bias (yellow). ViTA kernels are asynchronous; LayerNorm computations, $\sum x_i$, and $\sum x_i^2$, use the purple bus for accumulated updates. (b) Inside the ViTA kernel: data crossing clock domains enters via asynchronous FIFOs, including input, weight, Key/Value (KV), and bias data.

$$\sigma = \sqrt{\frac{1}{C} \times \sum_{i=1}^{C} x_i^2 - \mu^2} \tag{1}$$

III. ViTA Architecture

A. ViTA Architecture and Bus Overview

As drawn in Fig. 3a, ViTA incorporates asynchronous and scalable kernels that handle all operations in the Transformer, interconnected via buses. The input bus directs current decoder layer data to each kernel, while the weight and bias buses distribute their respective data across all kernels. The output bus gathers each kernel's output data, forwarding it to the next decoder layer. The key and value bus facilitates the collection and distribution of KV matrix among all kernels. ViTA employs

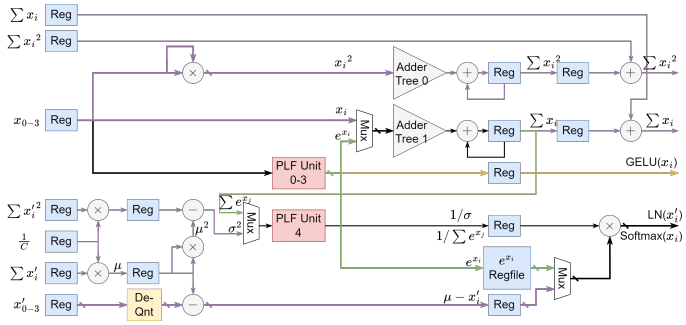


Fig. 4. The data path of the fused and configurable special function module for non-linear functions. Color-coded arrows represent the data paths specific to individual functions (Section III-D), while black arrows are shared paths.

TABLE I
FUNCTIONALITY OF PLF UNITS AND ADDER TREES

Mode	PLF Unit 0-3	PLF Unit 4	Adder Tree 0	Adder Tree 1
Softmax	$f(x) = e^x$	$f(x) = 1/x$	–	$\sum e^{x_i}$
LayerNorm	–	$f(x) = 1/\sqrt{x}$	$\sum {x_i}^2$	$\sum x_i$
GELU	$f(x) = \text{GELU}(x)$	–	–	–

two weight buffers as ping-pong buffers, multiplexing their data to the Key and Value bus, which is then broadcast to all kernels.

B. ViTA Kernel

As shown in Fig. 3b, the ViTA kernel consists of a VMU Lane, three matrix buffers, an accumulator, two partial sum buffers as ping-pong buffers, and a special function module. The input data, *e.g.*, input matrix, KV matrix, weight matrix, and bias are synchronized through asynchronous FIFOs.

C. VMU Lane

Each VMU Lane consists of n Vector-Multiplication Units (VMUs). As illustrated in Fig. 6b, in temporal, the VMU Lane computes a single GEMV operation. Every VMU outputs the inner product result of two signed vectors, whose length is k.

D. Special Function Module

The Special function module computes non-linear functions in ViT, namely GELU, Softmax, and LayerNorm, as drawn in Fig. 4. The role of PLF units and adder trees for these functions is summarized in Table I. The design utilizes the PLF unit from [20] which is adopted in BERT without accuracy degradation.

1) Softmax: Softmax paths are marked by green arrows in Fig. 4. It is calculated using approximated exponential and reciprocal functions. Exponential values, e^{x_i}, are computed by PLF Units 0 to 3 and stored in an exponential buffer. Meanwhile, the exponential values are summed up by the adder tree and the accumulator to compute the sum of the exponential values. Instead of using expensive dividers, the vector division is converted to one reciprocal function and one relatively cheaper vector multiplication. After the whole row is processed, the reciprocal of the sum of the exponential values is computed by PLF Units 4, which acts as the reciprocal function. Then it is multiplied by the exponential values read from the exponential buffer, to compute the Softmax values. The Softmax is row-wise, aligning with the GEMM dataflow.

2) LayerNorm: LayerNorm paths are marked by purple arrows in Fig. 4. LayerNorm is computed in two phases for memory efficiency. The first stage calculates mean, μ, and variance, σ. The second uses these values to compute the layer normalization. Variance σ is computed using Eq. 1 for better latency and throughput. Due to the matrix-based computation nature of LayerNorm, values across ViTA kernels are aggregated for the final result. As the LayerNorm is computed in two stages, there is no special constraint on the GEMM dataflow.

3) GELU: GELU-specific paths are highlighted by yellow arrows in Fig. 4. As summarized in Table II, the GELU function and the second stage of the LayerNorm function are computed simultaneously in the Special function module. GELU computation involves PLF Units 0 to 3 and is element-wise, posing no unique constraints on the GEMM dataflow.

IV. VITA DATAFLOWS

We introduce an innovative approach to optimize the dataflow within the ViTA architecture to reduce the memory bandwidth and capability requirements. Specifically, Row Stationary (RS) and Column Stationary (CS) General Matrix Multiply (GEMM) dataflows are adopted for this purpose. Two Output Stationary (OS) dataflows are introduced to compute special functions to decrease throughput requirements. A fused Multi-Head Attention dataflow, vital for Transformer models, is also introduced. The dataflows of each layer in the ViT are summarized in Table II.

A. GEMM and Special Functions Dataflows in ViTA

1) GEMM Dataflow: GEMM is the foundational operation in Transformer models. For a GEMM operation formulated as $C_{M\times N} = A_{M\times K} \times B_{K\times N}$, there are six ways to implement it in software and hardware using three nested for loops, which is summarized in Table III, where the sequence of M, N, and K represent the order of nested for loop, from outer to inner. Different permutations impact both performance and energy efficiency with different access patterns for A and B. ViTA supports both MKN, *i.e.*, RS and NKM, *i.e.*, CS dataflows for higher energy efficiency and less memory footprint [27].

Tiling is employed for large-scale GEMM operations, as visualized in Fig. 6a. Each GEMM operation is distributed across multiple ViTA kernels to enhance parallelism and performance.

2) Special Function Dataflow: Non-linear operations, critical to Transformer computations, are resource-intensive. Two OS dataflows, MNK and NMK, have been adopted in the special function dataflow to reduce the throughput requirement hence reducing the number of complex arithmetic units. To convert the GEMM dataflow to the special function dataflow, two partial sum buffers are introduced as ping-pong buffers.

B. Fused Multi-Head Attention Dataflow

Shown in Fig. 5, ViTA adopts a memory-centric, hardware-efficient fused MHA dataflow to reduce the memory requirements across the MHA layer. In ViTA, MHA is computed per head, and memories are reused within every self-attention. Algorithm 1 describes this dataflow from the input matrix X to the input of the MLP layers, which is drawn in Fig. 5a. The

TABLE II

SUMMARY OF VITA DATAFLOWS, INCLUDING THE NAME OF VIT OPERATIONS, THE SOURCE AND DESTINATION OF THE GEMM COMPUTATION, GEMM DATAFLOWS, THE SPECIAL FUNCTIONS BEFORE AND AFTER THE GEMM OPERATIONS, AND DATAFLOWS IN THE VIEW OF SPECIAL FUNCTIONS.

Layer	Operation		Matrix A	Matrix B	Post Special Function	GEMM Dataflow	(Pre) Special Function	Special Function Dataflow	Matrix C
Input	–		–	–	–	–	–	–	MB B
Pre-Processing	Pre-Processing	Patches embedding	MB[a]B	WB[b]A/B	–	NKM	LN Stage 1	NMK	MB A
Encoder	MHA	$[Q, K, V]$	MB A	WB A/B	LN Stage 2	MKN	–	MNK	MB B/C[c]
		$A = \text{Softmax}(Q \cdot K)$	MB B	MB C	–	MKN	Softmax	MNK	MB A
		$Z = A \cdot V$	MB A	MB C	–	MKN	–	MNK	MB B
		Project GEMM	MB B	WB A/B	–	NKM	LN Stage 1	NMK	MB A
	FFN	FC 1	MB A	WB A/B	LN Stage 2	NKM	GELU	NMK	MB B&C
		FC 2	MB B&C	WB A/B	–	NKM	–	NMK	MB A
Final Classifier	MLP- Head		MB A	WB A/B	–	NKM	–	NMK	–

[a] MB is short for Matrix Buffer.
[b] WB is short for Weight Buffer. Weight Buffers A and B are ping-pong buffers for the weight.
[c] The Query matrix is stored in Matrix Buffer B, and the Key matrix and Value matrix are stored in Matrix Buffer C.

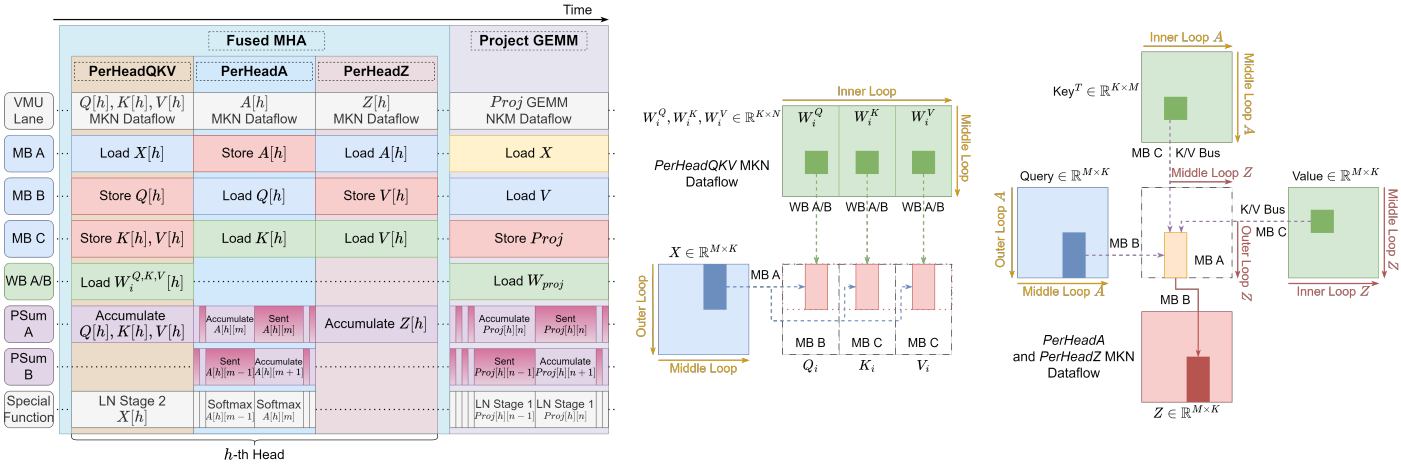


Fig. 5. (a) The fused multi-head attention dataflow in the view of each module, where MB is short for Matrix Buffer, WB is short for Weight Buffer, PSum A and B are Partial Sum Buffer A and B, which are acting as a ping-pong buffer. The GEMMs in PERHEADQKV, PERHEADA, PERHEADZ are MKN dataflow, while the GEMM in PROJGEMM is NKM dataflow. (b) The dataflow of the PERHEADQKV function. (c) The dataflow of the PERHEADA and PERHEADZ.

TABLE III

THE SUMMARY OF DATAFLOWS, OPTIMUM CONDITIONS, MEMORY AND BANDWIDTH BOTTLENECKS IN GEMM, THE THROUGHPUT REQUIREMENTS OF SPECIAL FUNCTIONS, AND WHETHER THEY ARE SUPPORTED IN VITA.

Dataflow[a]	Optimum Conditions[b]	In the View of GEMM		In the view of Special Functions	Supported in ViTA
		Memory Bottleneck	*Bandwidth Bottleneck*	*Throughput*[c]	
MNK-OS	$M \geq N$	Matrix B	Matrix A, B	n / K/k cycle	✓
NMK-OS	$M < N$	Matrix A	Matrix A, B	n / K/k cycle	✓
KMN-IS	$M \geq N$	Partial Sum Buffer	Matrix B	n / cycle	✗
MKN-RS	$M \geq N$	Matrix B	Matrix B	n / cycle	✓
KNM-WS	$M < N$	Partial Sum Buffer	Matrix A	n / cycle	✗
NKM-CS	$M < N$	Matrix A	Matrix A	n / cycle	✓

[a] OS is short for Output Stationary; IS is short for Input Stationary; RS is short for Row Stationary; WS is short for Weight Stationary; CS is short for Column Stationary.
[b] M and N are the number of rows and columns of the matrix C respectively.
[c] n and k are the computation parallelism of the hardware on N and K dimensions respectively.

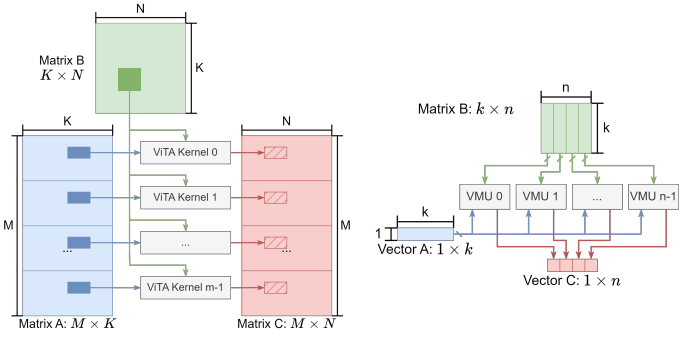


Fig. 6. (a) ViTA uses tiling to compute the large GEMM operation. For every GEMM, the matrix A ($\mathbb{R}^{M \times K}$) is split into m chunks, which are mapped into m ViTA kernels. In temporal, the ViTA computes a GEMM $c = a \times b$, where $a \in \mathbb{R}^{m \times k}$, $b \in \mathbb{R}^{k \times n}$. (b) In temporal, a VMU Lane computes a GEMV $c = a \cdot b$, where $a \in \mathbb{R}^{1 \times k}$, $b \in \mathbb{R}^{k \times n}$. The matrix b is mapped into n VMUs.

function PERHEADQKV is drawn in Fig. 5b, and the functions PERHEADA and PERHEADZ are drawn in Fig. 5c.

C. Gains of the ViTA Dataflows

Fig. 1 demonstrates that the original MHA that supports VGA images requires 22.66 MB when dedicated memories are used for each matrix. With our fused MHA, this requirement drops to 5.28 MB, marking a 76.71% reduction. In the MLP layers, while FC 1 results require 3.52 MB, the weights demand 2.25 MB. By adopting dual GEMM dataflows, 48 KB is needed for the weight buffers while maintaining the weight reuse. The bandwidth requirements for the weights are reduced from 256

Algorithm 1 Fused Multi-Head Attention Algorithm

Require: Number of heads H
Require: Matrix $X \in \mathbb{R}^{(N+1)\times D}$ stored in MB A
Require: $W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{D\times D_h}$, $W_{proj} \in \mathbb{R}^{H\times D_h \times D}$
Let m, k and n be the computation parallelism of ViTA on M, K, and N dimensions
$T_h = H$
$T_m = \lceil (N+1)/m \rceil$
$T_k = \lceil D/(H \cdot k) \rceil$
$T_n = \lceil D_h/n \rceil$
Tile X into $T_h \times T_m \times T_k$ blocks ($m \times k$ each), and tile W_i^Q, W_i^K, W_i^V into $T_h \times T_k \times T_n$ blocks ($k \times n$ each)
for $1 \le t_h \le T_h$ **do**
    PERHEADQKV($X[t_h]$, $W_i^Q[t_h]$, $W_i^K[t_h]$, $W_i^V[t_h]$)
    PERHEADA($Q[t_h]$, $K[t_h]$)
    PERHEADZ($A[t_h]$, $V[t_h]$)
end for
PROJGEMM(X, Z, W_{proj})

TABLE IV
EXPERIMENTAL SETUP

Design Tools	
HDL	Chisel
Verilog simulator	Verilator [29]
Logic synthesis	Cadence Genus
Design Parameters	
VMU length	8/16
# VMU lanes	8/16/32
GEMM input bit width	8 bit
GEMM output bit width	32 bit
Image resolution	224×224
Technology node	Foundry 28 nm FD-SOI
Core supply voltage	1.05 V

GB/s to an affordable 56 GB/s when our ViTA operates at 1 GHz. Such savings increase the efficiency of area and power.

V. EVALUATION

A. Experimental Setup

We implement the ViTA in Chisel [28] in a configurable manner and evaluate it for ViT-base with 224×224 RGB images as input. Memory IPs (Regfiles and SRAMs) are generated by ARM memory compilers. The generated Verilog by Chisel and memory IPs are synthesized by Cadence Genus with Foundry 28 nm FD-SOI technology node for both area and power estimation. The experimental setup is summarized in Table IV.

B. Experimental Results

We evaluated the performance (GOPS, throughput, and latency), area, power, area efficiency, power efficiency, and memory bandwidth of ViTA at different clock frequencies and hardware configurations. The parameters m, k, and n, ranging from 8 to 32, balance the performance with the area- and power-efficiency, scaling from ViTA Tiny's low-power to Huge's high performance, with Base and Large showing the metrics trends.

1) ViTA at Different Clock Frequencies: We synthesized ViTA Huge at 200 MHz, 300 MHz, 500 MHz, and 1 GHz, which are summarized in Table V. ViTA achieves an area efficiency of 2,131.85 GOPS/mm^2 at 1 GHz and achieves a power efficiency of 1,929.49 GOPS/W at 200 MHz.

TABLE V
THE EXPERIMENTAL RESULTS OF VITA HUGE AT DIFFERENT CLOCK FREQUENCIES, SUPPORTING A 224×224 RGB IMAGE AS INPUT.

ViTA Huge $m=32$ $k=n=16$	**200 MHz**		**300 MHz**		**500 MHz**		**1 GHz**	
	Area ***(mm^2)***	***Power*** ***(mW)***	***Area*** ***(mm^2)***	***Power*** ***(mW)***	***Area*** ***(mm^2)***	***Power*** ***(mW)***	***Area*** ***(mm^2)***	***Power*** ***(mW)***
ViTA	6.48	1,698	6.79	2,689	6.97	4,626	7.69	10,403
GOPs	3,276.80		4,915.20		8,192.00		16,384.00	
GOPS/mm^2 or GOPS/W	506	1,930	724	1,828	1,176	1,771	2,132	1,575
Bandwidth (GB/s) [a]	7.4		11.1		18.5		37.0	
Frame/s[a]	104.483		156.724		261.207		522.414	
Latency (ms)[a]	9.571		6.381		3.828		1.914	

[a] Bandwidth, latency and frame rate are calculated with a 224×224 RGB image as input on ViT-Base.

TABLE VI
THE EXPERIMENTAL RESULTS OF VITAS WITH DIFFERENT COMPUTATION CAPABILITIES, SUPPORTING A 224×224 RGB IMAGE AS INPUT.

ViTA @ 300 MHz	**ViTA Tiny** $m=8$ $k=n=8$		**ViTA Base** $m=16$ $k=n=8$		**ViTA Large** $m=16$ $k=n=16$		**ViTA Huge** $m=32$ $k=n=16$	
	Area ***(mm^2)***	***Power*** ***(mW)***	***Area*** ***(mm^2)***	***Power*** ***(mW)***	***Area*** ***(mm^2)***	***Power*** ***(mW)***	***Area*** ***(mm^2)***	***Power*** ***(mW)***
ViTA	2.00	325	2.53	637	4.37	1,487	6.79	2,689
GOPs	307.20		614.40		2,457.60		4,915.20	
GOPS/mm^2 or GOPS/W	153	944	243	964	563	1,652	724	1,828
Bandwidth (GB/s) [a]	0.9		1.5		6.0		11.1	
Frame/s[a]	11.006		21.158		84.451		156.724	
Latency (ms)[a]	90.856		47.264		11.841		6.381	

[a] Bandwidth, latency and frame rate are calculated with a 224×224 RGB image as input on ViT-Base.

2) ViTA with Different Hardware Configurations: We synthesized ViTA with different hardware configurations at 300 MHz, which are summarized in Table VI. The area and power of a Tiny ViTA are 2.00 mm^2 and 325.45 mW, respectively, which is 3.40× smaller than the area of a Huge ViTA and 8.26× smaller than the power of a Huge ViTA.

C. Area and Power Breakdown

The area and power breakdown of the ViTA Kernel in ViTA Huge and that in ViTA Tiny at 300 MHz are shown in Fig. 7.

As Fig. 7 shows, memories occupy 44% area and 38% power in the ViTA kernel in ViTA Huge, due to the efficient ViTA dataflow, achieving a high area efficiency of 724.05 GOPS/mm^2 and a high power efficiency of 1,828.13 GOPS/W at 300 MHz.

With the number of ViTA kernels decreasing, the capability of the matrix buffers increases, which leads to an increased percentage in the area and power of memories. In ViTA Tiny, memories occupy 84% area and 57% power.

Even with the optimized special function module, it occupies 22% area in ViTA Huge, which is 66% of the VMU lane.

D. Comparison with Related Works

We summarized the comparisons between the related works and our introduced ViTA architectures, ViTA Tiny at 200 MHz, and ViTA Huge at 1 GHz for area- and power-constraint applications and high-performance applications, respectively, in Table VII. Only SwiftTron [15] and our work supports the entire workload. Compared with the work in [11], ViTA demonstrates a balance of power and area efficiency, outperforming 27.85× in the latter metric. ViTA Huge achieves 1.40× and 2.46× better power and area efficiency than ELSA [14]. The power of ViTA Tiny is 155× and 4.05× less than that of SwiftTron [15] and ViTA [16]. In short, our architecture achieves a competitive area and power efficiency while supporting the full workload.

TABLE VII
SUMMARY OF COMPARISONS BETWEEN THE RELATED WORKS AND OUR INTRODUCED VITA ARCHITECTURE.

Accelerator	Year	Non-Linear Functions			Entire Workload	Tech	Area	Power	Freq.	Perf.	Power Eff.	Area Eff.	Type	Precision
		GELU	*Softmax*	*LN*		*nm*	*mm²*	*mW*	*MHz*	*GOPS*	*GOPS/W*	*GOPS/mm²*		
A3 [13]	2020	✗	INT9	✗	✗	40	2.08	110	1000	221	2,001.41	106.25	ASIC	INT9
ELSA [14]	2021	✗	INT8	✗	✗	40	1.26	969	1000	1,090	1,124.45	865.08	ASIC	INT8 FP16
Wang et al. *[11]*	2022	✗	INT12	✗	✗	28	6.82	273	510	522	1,913.49	76.54	ASIC	INT12
ViTA [16]	2023	INT8	✗	INT8	✗	28	–	880	150	–	–	–	FPGA	INT8
SwiftTron [15]	2023	INT32	INT32	INT32	✓	65	273	33,640	143	–	–	–	ASIC	INT8
ViTA Tiny (This Work)	2023	INT32	INT32	INT32	✓	28	2.00	217	200	204.80	943.42	102.57	ASIC	INT8
ViTA Huge (This Work)							7.69	10,403	1000	16,384	1,574.91	2,131.85		

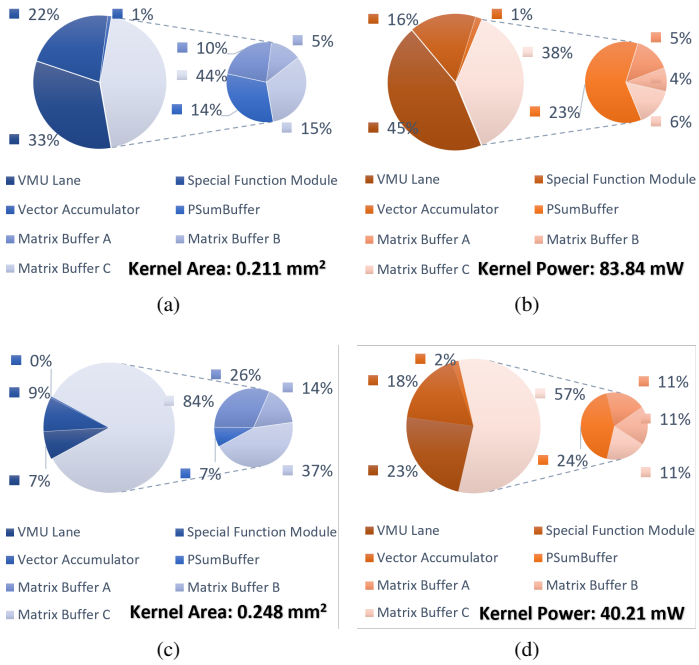


Fig. 7. (a) Area and (b) power of the ViTA Kernel in the ViTA Huge. (c) Area and (d) power in the ViTA Tiny. In all sub-figures, the sections depicted with a lighter shade indicate modules associated with memory.

VI. CONCLUSION

We introduced ViTA, a highly efficient and scalable hardware accelerator for the entire ViT model, optimizing and unifying operations via a novel dataflow. Our memory-centric fused MHA dataflow significantly reduces memory requirements by 76.71%, weight buffer size to 48 KB, and the bandwidth requirements to 56 GB/s by supporting MKN-RS and NKM-CS GEMM dataflows. Ping-pong partial sum buffers enable OS dataflows, reducing the throughput requirements of special functions. Furthermore, we introduce a configurable module to support approximated non-linear functions, *i.e.*, Softmax, LayerNorm, and GELU. Empowered by these improvements, ViTA achieves an area efficiency of 2,131.85 GOPS/mm^2 and a power efficiency of 1,574.91 GOPS/W at 1 GHz, ensuring the acceleration of the entire ViT workload. Through the comprehensive design space exploration, we present ViTA's flexibility, supporting 0.20-16.38 TOPS, with the area and power consumption of 2.00-7.69 mm^2 and 0.22-10.40 W, respectively, making it apt for a wide range of applications.

REFERENCES

[1] A. Vaswani *et al.*, "Attention is all you need," *NeurIPS*, vol. 30, 2017.
[2] Z. Li and Q. Gu, "I-vit: integer-only quantization for efficient vision transformer inference," *arXiv*, 2022.
[3] H. Touvron *et al.*, "Training data-efficient image transformers & distillation through attention," in *ICML*. PMLR, 2021, pp. 10 347–10 357.
[4] Z. Liu *et al.*, "Swin transformer: Hierarchical vision transformer using shifted windows," in *ICCV*, 2021, pp. 10 012–10 022.
[5] OpenAI, "Gpt-4 technical report," 2023.
[6] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv*, 2020.
[7] H. Tabani *et al.*, "Improving the efficiency of transformers for resource-constrained devices," in *DSD*. IEEE, 2021, pp. 449–456.
[8] "Fastertransformer," GitHub, 12 2022. [Online]. Available: https://github.com/NVIDIA/FasterTransformer/tree/main
[9] S. Kim, A. Gholami, Z. Yao, M. W. Mahoney, and K. Keutzer, "I-bert: Integer-only bert quantization," in *ICML*. PMLR, 2021, pp. 5506–5518.
[10] L. Liu *et al.*, "Dynamic sparse attention for scalable transformer acceleration," *IEEE Transactions on Computers*, 2022.
[11] Y. Wang *et al.*, "A 28nm 27.5 tops/w approximate-computing-based transformer processor with asymptotic sparsity speculating and out-of-order computing," in *ISSCC*, vol. 65. IEEE, 2022, pp. 1–3.
[12] ——, "A 28nm 77.35 tops/w similar vectors traceable transformer processor with principal-component-prior speculating and dynamic bit-wise stationary computing," in *VLSIC*. IEEE, 2023, pp. 1–2.
[13] T. J. Ham *et al.*, "Aˆ 3: Accelerating attention mechanisms in neural networks with approximation," in *HPCA*. IEEE, 2020, pp. 328–341.
[14] ——, "Elsa: Hardware-software co-design for efficient, lightweight self-attention mechanism in neural networks," in *ISCA*, 2021, pp. 692–705.
[15] A. Marchisio *et al.*, "Swifttron: An efficient hardware accelerator for quantized transformers," *arXiv*, 2023.
[16] S. Nag *et al.*, "Vita: A vision transformer inference accelerator for edge applications," *arXiv*, 2023.
[17] Y. Chen *et al.*, "High-frequency systolic array-based transformer accelerator on field programmable gate arrays," *Electronics*, vol. 12, no. 4, p. 822, 2023.
[18] W. Wang *et al.*, "Pyramid vision transformer: A versatile backbone for dense prediction without convolutions," in *ICCV*, 2021, pp. 568–578.
[19] T. Dao, D. Fu *et al.*, "Flashattention: Fast and memory-efficient exact attention with io-awareness," *NeurIPS*, vol. 35, pp. 16 344–16 359, 2022.
[20] J. Yu *et al.*, "Nn-lut: neural approximation of non-linear operations for efficient transformer inference," in *DAC*. IEEE, 2022, pp. 577–582.
[21] D. Hendrycks *et al.*, "Gaussian error linear units (gelus)," *arXiv*, 2016.
[22] H. Lu *et al.*, "An efficient piecewise linear approximation of non-linear operations for transformer inference," in *FCCM*, 2023, pp. 206–206.
[23] B. Yuan, "Efficient hardware architecture of softmax layer in deep neural network," in *SOCC*, 2016.
[24] X. Geng *et al.*, "Hardware-aware softmax approximation for deep neural networks," in *ACCV*, 2018.
[25] G. C. Cardarilli *et al.*, "A pseudo-softmax function for hardware-based high speed image classification," *Scientific reports*, 2021.
[26] J. Fang *et al.*, "Turbotransformers: an efficient gpu serving system for transformer models," in *PPoPP*, 2021, pp. 389–402.
[27] Y.-H. Chen *et al.*, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," *ACM SIGARCH computer architecture news*, vol. 44, no. 3, pp. 367–379, 2016.
[28] J. Bachrach *et al.*, "Chisel: constructing hardware in a scala embedded language," in *DAC*. IEEE, 2012, pp. 1212–1221.
[29] W. Snyder, "Verilator and systemperl," in *DAC*. IEEE, 2004.