SynGPU: Synergizing CUDA and Bit-Serial Tensor Cores for Vision Transformer Acceleration on GPU

Yuanzheng Yao[1], Chen Zhang[1]*, Chunyu Qi[1], Ruiyang Chen[1], Jun Wang[2], Zhihui Fu[2],
Naifeng Jing[1], Xiaoyao Liang[1], and Zhuoran Song[1]*
[1]Shanghai Jiao Tong University, Shanghai, China
[2]OPPO Research Institute
[1]{yzh_yao, songzhuoran}@sjtu.edu.cn

Abstract*—Vision Transformers (ViTs) have demonstrated remarkable performance in computer vision tasks by effectively extracting global features. However, their self-attention mechanism suffers from quadratic time and memory complexity as image resolution or video duration increases, leading to inefficiency on GPUs. To accelerate ViTs, existing works mainly focus on pruning tokens based on value-level sparsity. However, they miss the chance to achieve peak performance as they overlook the bit-level sparsity. Instead, we propose Inter-token Bit-sparsity Awareness (IBA) algorithm to accelerate ViTs by exploring bit-sparsity from similar tokens. Next, we implement IBA on GPUs that synergize CUDA and Tensor Cores by addressing two issues: firstly, the bandwidth congestion of the Register File hinders the parallel ability of CUDA and Tensor Cores. Secondly, due to the varying exponent of floating-point vectors, it is hard to accelerate bit-sparse matrix multiplication and accumulation (MMA) in Tensor Core through fixed-point-based bit-level circuits. Therefore, we present SynGPU, an algorithm-hardware co-design framework, to accelerate ViTs. SynGPU enhances data reuse by a novel data mapping to enable full parallelism of CUDA and Tensor Cores. Moreover, it introduces Bit-Serial Tensor Core (BSTC) that supports fixed- and floating-point MMA by combining the fixed-point Bit-Serial Dot Product (BSDP) and exponent alignment techniques. Extensive experiments show that SynGPU achieves an average of 2.15× ∼ 3.95× speedup and 2.49× ∼ 3.81× compute density over A100 GPU.

I. Introduction

Recently, attention-based models in the form of Vision Transformers (ViTs) [1] have achieved exceptional performance in the field of computer vision (CV) tasks such as image classification [2], object detection [3], and video recognition [4]. Compared to traditional convolutional neural network (CNN) models, ViTs leverage their ability to extract global features effectively, offering both enhanced performance and greater flexibility in diverse applications.

However, the achievements of ViTs are at the cost of inefficiency. As is well known, the self-attention mechanism in ViTs exhibits quadratic time and memory complexity with respect to the number of input tokens. Consequently, as image resolution or video duration increases, the computational performance of ViTs on GPUs deteriorates significantly. To address this limitation, various approaches have been proposed [5]–[8], focusing on reducing value-level redundancy by exploiting token similarity across image patches or video frames. Beyond value sparsity, our analysis reveals that bit-level redundancy offers an opportunity to further enhance the computational efficiency of ViTs.

As illustrated in Fig. 1, adjacent image pixels often manifest similar values, leading to many 0-bits in their differences. Building on this observation, we propose an Inter-token Bit-sparsity Awareness (IBA) algorithm that calculates the distances between input tokens and computes token differences by subtracting similar pairs. These token differences demonstrate increased bit-level sparsity, which can be effectively leveraged to accelerate computations. To implement this strategy on GPUs, we utilize the two inherently separate computing cores within the GPU: CUDA Cores and Tensor Cores, ***utilizing their complementary functions to enhance the efficiency of ViTs by harnessing bit-level sparsity***.

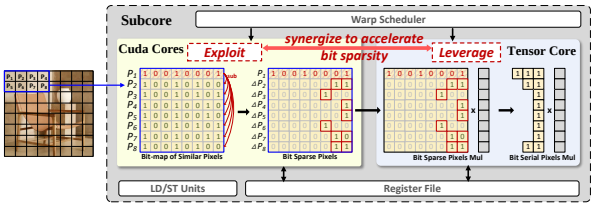
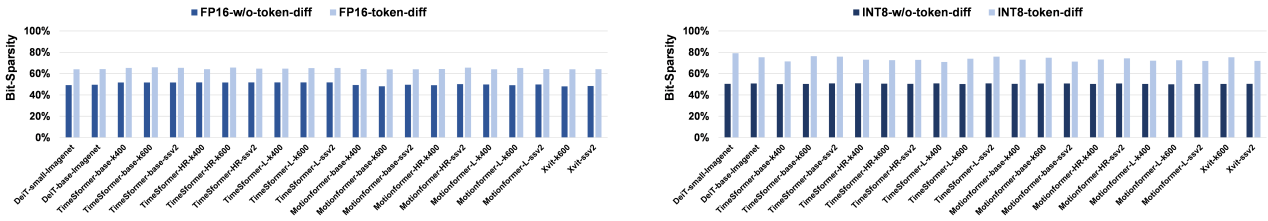


Fig. 1. Synergy mechanism of SynGPU.

The key insights of this scheme lie in two points: *first, leveraging the high-precision capabilities of CUDA Cores to exploit bit-level sparsity among similar tokens without compromising accuracy; second, utilizing Tensor Cores, optimized for high-throughput matrix multiplication and accumulation (MMA) operations, to enable bit-sparse acceleration.* However, achieving efficient synergy between CUDA Cores and Tensor Cores presents two critical challenges. Firstly, CUDA Cores and Tensor Cores compete for Register File (RF), which can result in RF bandwidth congestion, hindering their ability to operate in full parallelism. Secondly, Tensor cores implement the floating-point MMA by executing dot product (DP) operations in parallel. It is non-trivial to design bit-level circuits to support both fixed-point and floating-point DP due to the variability in the exponents of floating-point values within the vectors.

* Zhuoran Song and Chen Zhang are corresponding authors.
This work is partly supported by the National Natural Science Foundation of China (Grant No.62202288)



(a) Bit-level sparsity in FP16

(b) Bit-level sparsity in INT8

Fig. 2. Bit-level sparsity in FP16/INT8 between token differences and token without differences

In this paper, we propose SynGPU, an algorithm-hardware collaborative framework, that synergizes CUDA Cores and Tensor Cores for accelerating ViTs on the GPU. Specifically, to address the first challenge, SynGPU includes a novel data mapping scheme that enhances data reuse in Tensor Cores. This approach significantly reduces the frequency of RF accesses by Tensor Cores, thereby freeing up RF resources for CUDA Cores. Moreover, to tackle the second challenge, we develop a Bit-Serial Dot Product (BSDP) algorithm along with the Bit-Serial Tensor Core (BSTC), which aligns the exponents of floating-point values within each vector and repackages the results of DP operations. This design enables SynGPU to support both floating-point and fixed-point bit-sparse operations effectively.

In summary, our contributions are as follows:

- We propose SynGPU, an algorithm-hardware co-design framework that leverages the synergy between CUDA Cores and Tensor Cores on GPUs to accelerate bit-sparse ViTs. By utilizing CUDA Cores to exploit bit-level sparsity derived from token similarity and coordinating with Tensor Cores to perform bit-sparse MMA, SynGPU achieves substantial performance improvements for ViTs. To the best of our knowledge, SynGPU is the first framework specifically designed for bit-level ViTs acceleration on GPUs.
- At the algorithmic level, we propose two algorithms: IBA and BSDP. IBA is designed to extract bit-level sparsity among similar tokens, and BSDP focuses on combining both fixed-point and floating-point MMA by aligning the exponents of floating-point vectors to enable efficient bit-level sparse floating-point MMA.
- At the hardware level, we make two contributions. Firstly, we design a new data mapping that resolves the issue of bandwidth congestion, enabling full parallelism between CUDA cores and tensor cores. Secondly, we develop BSTC that is capable of accelerating bit-sparse MMA operations in fixed- and floating-point formats.

II. Background and Motivation

A. Vision Transformer

Inspired by the remarkable performance of Transformer [9] on NLP tasks, numerous efforts have been made to adapt Transformers for visual applications. Among them, ViTs stands out with its concise design and powerful capabilities in image and video tasks. ViTs divide images or video frames into patches, which are then converted into tokens through a Patch-Embedding Layer, enabling them to be directly fed into the Transformer.

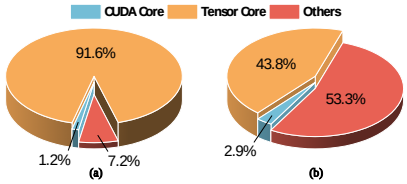


Fig. 3. The occupancy of RF bandwidth (a) and activation time (b) of CUDA Cores and Tensor Cores on A100 when executing ViTs.

Several accelerator architectures [5]–[7] have been proposed to optimize ViTs. For instance, DynamicViT [6] designs a dedicated dynamic prediction module to predict and prune redundant tokens. ViTCoD [5] employs an autoencoder module to reorder attention maps, creating denser or sparser fixed patterns to balance ViTs' workload. However, these advancements primarily focus on optimizing value-level sparsity and neglect the potential bit-level sparsity among similar tokens.

B. Ampere GPU

In recent years, the rapid advancement of neural networks and their substantial computational requirements have driven significant evolution in GPU structure and performance. The Nvidia A100 [10] serves as a prime example, featuring multiple Streaming Multiprocessors (SMs). Each Subcore of SM comprises several CUDA cores and a Tensor Core, sharing a common RF. Specifically, CUDA cores are utilized for high-precision general-purpose computations, whereas Tensor Cores are designed for high-throughput MMA operations, incorporating numerous DP units that support various fixed- and floating-point formats.

C. Motivation

While current research on ViT accelerators mainly focuses on value-level sparsity, our experimental results display a more promising form of bit-level sparsity among similar tokens. As demonstrated in Fig. 2, we run the implementation of various representative ViT models and subtract similar tokens within the attention map. As a result, we observe a significantly higher number of 0-bits for both FP16 and INT8, with bit sparsity increasing from 50.19% to 65.98% and from 50.48% to 75.82%, respectively.

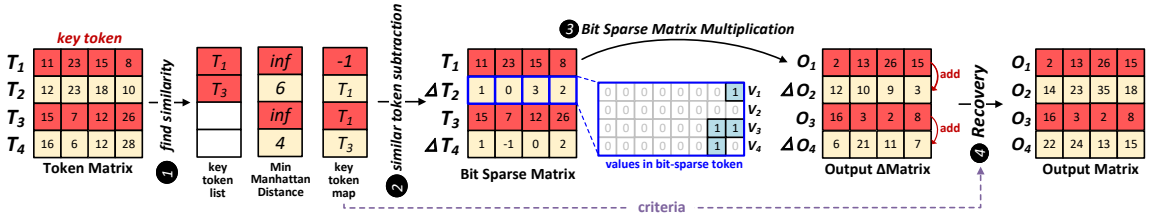


Fig. 4. Overview of the Inter-token Bit-sparsity Awareness algorithm.

Based on such observations, we intend to extract and leverage the bit-level sparsity to accelerate ViTs by collaborating with CUDA Cores and Tensor Cores. However, as shown in Fig. 3, when Tensor Core operates at full capacity with the RF bandwidth occupancy of 91.6%, CUDA Cores are mostly idle due to the competitive access to RF. Furthermore, different exponents of floating-point vectors impede accelerating MMA in Tensor Cores with fixed-point bit-level designs [11], [12]. Despite the effort made by Bitlet [13] to support bit-level floating-point formats, it leads to insufficient hardware utilization in lower-bit fixed-point computations. To address such inefficiencies, we propose SynGPU, an algorithm-hardware co-design framework that facilitates concurrent utilization of CUDA Cores and Tensor Cores to accelerate ViTs.

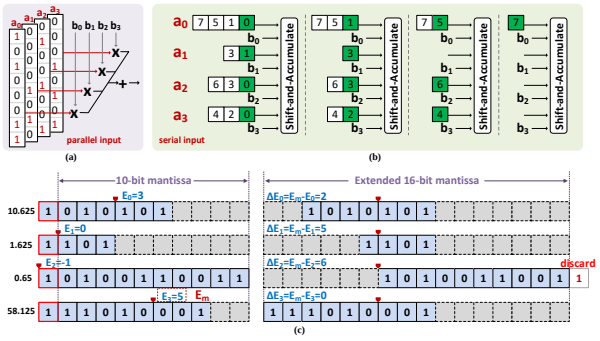


Fig. 5. Details of: (a) Parallel Dot Product. (b)Bit-Serial Dot Product. (c) Exponent alignment algorithm

III. Algorithm

A. Inter-token Bit-sparsity Awareness

In this section, we introduce Inter-token Bit-sparsity Awareness (IBA), an algorithm aimed at exploiting bit-level sparsity within ViTs. As illustrated in Fig. 4, in **step ❶**, we select a subset of key tokens at a regular interval, while the remaining tokens are non-key ones. Next, we compute the minimum Manhattan Distance as the criteria to locate the most similar key token to each of the non-key tokens and record them in a key token map. In the given example, we select key tokens at an interval of 2 and find that the most similar tokens to T_2 and T_4 are T_1 and T_3. **Step ❷** then subtracts the similar token pairs to acquire the Bit Sparse Matrix composed of numerous bit-sparse difference tokens. The blue-highlighted part displays the binary expressions of the values in the difference token $\Delta T_2 = T_2 - T_1$, exhibiting that the similarity between tokens leads to more redundant 0-bits. Subsequently, **step ❸** executes bit-sparse MMA operations to obtain the Output ΔMatrix efficiently. Finally, based on the linearity of MMA operations, **step ❹** utilizes the information in the key token map to add the outputs of O_1, O_3 to ΔO_2 and ΔO_4, respectively, thereby recovering the Output Matrix losslessly.

B. Bit-Serial Dot Product

The main challenge of performing bit-sparse MMA is to support both floating-point and fixed-point formats, particularly given the varying exponents of floating-point vectors. Since floating-point multiplication consists of two main operations: exponent addition and mantissa multiplication, with the latter following the same mechanism as fixed-point multiplication, we propose Bit-Serial Dot Product (BSDP), an algorithm that combines fixed-point bit-serial multiplication with floating-point exponent alignment to address the challenge.

Fixed-point BSDP. The conventional DP operation depicted in Fig. 5(a) computes all bits in parallel. To avoid unnecessary 0-bits, as illustrated in Fig. 5(b), we need to transform the multiplication process into a bit-serial input mode by encoding the matrix elements into bit index format. This approach converts multiplication into a series of shift-and-accumulate operations, effectively skipping the 0-bits.

Floating-point BSDP. A floating-point operand is expressed as $X = 2^E \times M$, where M is the mantissa, and E is the exponent (we omit the sign bit for simplicity). The floating-point BSDP consists of three steps: exponent alignment, mantissa multiplication, and repackage. As illustrated in Fig. 5(c), we first align the exponents of all elements in the vector to the largest E_m by right-shifting the mantissa. Additionally, since the right-shifting may cause overflow and subsequent loss of precision, we extend the 10-bit mantissa to 16 bits:

$$X_i = 2^{E_i} \times M_i = 2^{E_m} \times (M_i \gg \Delta E_i) \tag{1}$$

where $\Delta E_i = E_m - E_i$ are the right-shifting bits for mantissa. As for floating-point BSDP $\vec{a} \times \vec{b}$, we separately align both exponents and then multiply the mantissa following the same way as fixed-point BSDP. Subsequently, we repackage the result into the standard floating-point format by adding the unified exponents by Eqn. (2):

$$\begin{aligned} dp &= \sum_{i=0}^{m} [2^{E_i^a} \times M_i^a] \times [2^{E_i^b} \times M_i^b] \\ &= \sum_{i=0}^{k} 2^{E_m^a + E_m^b} \times (M_i^a \gg \Delta E^a) \times (M_i^b \gg \Delta E^b) \end{aligned} \tag{2}$$

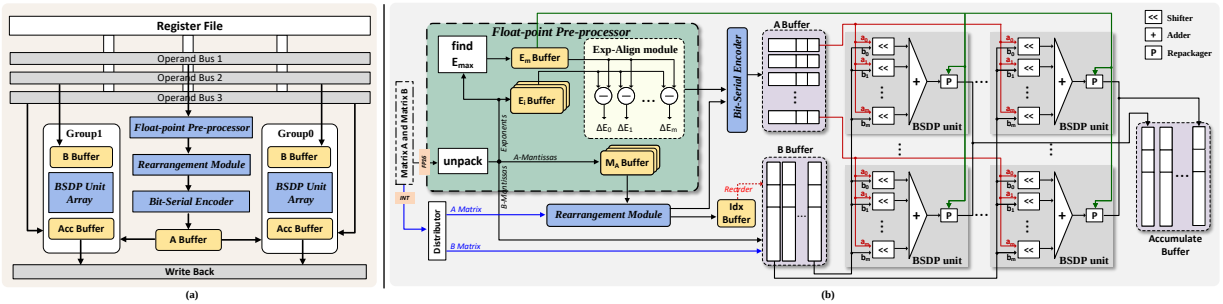


Fig. 6. Architecture of Bit-Serial Tensor Core: (a) Overview of BSTC. (b) Details of BSTC.

To reduce redundant computations, we further combine the two right shifts in alignment operations and the left shift in fixed-point BSDP to obtain the final result of floating-point BSDP operations, as shown in Eqn. (3):

$$dp = \sum_{i=0}^{k} \sum_{j=0}^{l} 2^{E_m^a + E_m^b} \times [M_i^b \ll (I_j - \Delta E^a - \Delta E^b)] \quad (3)$$

where I_j denotes the index of the 1-bit positions in the elements of $\vec{a}$.

IV. Architecture

A. Overview

The SynGPU architecture comprises two key designs. First, we introduce a data mapping scheme to solve the RF bandwidth congestion issue between CUDA Cores and Tensor Cores, enabling parallel processing capabilities for both cores. Second, we propose the Bit-Serial Tensor Core (BSTC), which is capable of computing both fixed- and floating-point MMA. Additionally, to maximize the efficiency of accelerating bit sparse MMA in BSTC, we design a rearrangement module to balance the matrix bit-sparsity.

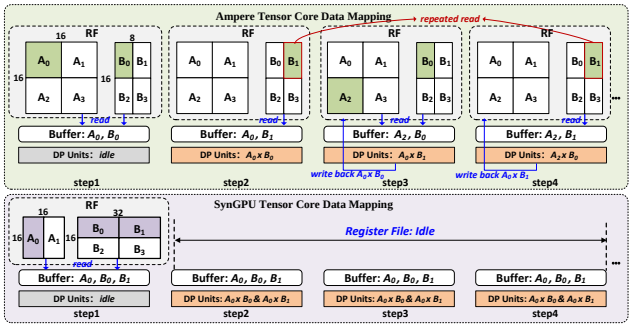


Fig. 7. Details of data mapping.

B. Data Mapping

When performing a 16 × 16 × 8 matrix operation in the original Tensor Core, it is separated into 8 steps of 8 × 8 × 4 MMA process (only 4 steps are given in Fig. 7), operating in a pipelined manner, with each process consisting of 3 parts: reading from RF, calculation, and writing back to RF. As depicted in the figure, in step 1, the Buffer reads A_0 and B_0, while the calculation and writing back correspond to step 2 and step 3. Unfortunately, the above execution consumes most of the RF's memory bandwidth, causing the CUDA cores to remain idle during Tensor Core operations. Additionally, this data flow of Tensor Core also leads to repeated data read-write operations, such as B_1 in steps 2 and 4, resulting in unnecessary overhead.

To address this issue, we propose to increase the capacity of the Tensor Core's data buffer, enabling it to read all required data for the current computation at once during step 1. By converting the bits of elements from parallel to serial processing in BSDP operations, the RF becomes idle during subsequent DP computation steps, allowing it to be utilized by CUDA cores.

Moreover, as the serial process decreases parallelism in the Tensor Core, we expand the dimension of the B matrix to ensure that the throughput does not decrease. Specifically, due to the 16-bit extended mantissa detailed in III-B, the worst-case scenario requires up to 16× cycles to compute MMA than Ampere Tensor Core. Thus, the expansion of the B matrix is set as 16 × 32. Additionally, to handle the repeated read-write operations, we split the A matrix into two groups, each with a size of 16 × 8, to compute $A_0 \times B_0$ and $A_0 \times B_1$ simultaneously.

C. Bit-Serial Tensor Core Design

As mentioned in III-B, to achieve bit-sparse MMA, floating-point numbers are required to be pre-processed for exponent alignment, while fixed-point numbers are able to directly perform BSDP operations. Therefore, as illustrated in Fig. 6 the architecture of BSTC mainly consists of two modules: the BSDP Array and the Floating-point Pre-processor.

BSDP Array. To implement bit-sparse matrix A × matrix B, we replace the DP Units within the Tensor Core with BSDP Units and arrange them in a grid array to concurrently work on performing a number of BSDP operations, where the (x, y)-th Unit executes BSDP between the x-th row of A and the y-th column of B. Each BSDP Unit consists of several shifters, an adder, and a repackage module to accomplish the fixed-point BSDP described in III-B. Within each cycle, a vector from the B Buffer and a serial of bit index code of matrix A from the A Buffer are fetched to perform shift-and-accumulate operations, which are repeated until the entire vector DP is finished.

Floating-point Pre-processor. The primary function of the Floating-point Pre-processor is to unpack floating-point elements in matrices and align the exponents in terms of

vectors. Initially, the unpacking process splits and extracts the exponents and mantissa from the floating-point data, and then finds the largest exponents as E_m, respectively storing all the segments into E_i Buffer, M_A Buffer, and E_m Buffer. Notably, since the B-Mantissas do not necessitate bit-serial encoding, they can be directly deposited in the B Buffer for subsequent DP computations. Next, the Exp-Align module works on subtractions $\Delta E_i = E_m - E_i$ as mentioned in Eqn. (3). Subsequently, based on the principles discussed in III-B, the Bit-Serial Encoder combines ΔE_i and M_A for encoding into bit-serial forms and stores them in the A Buffer for BSDP operation with B-mantissas. Finally, the repackage module that interfaces with the E_m Buffer is added at the end of each Unit to normalize the DP results to standard floating-point format, merge $E_m^a + E_m^b$ into the final exponent, and store them into the Accumulate Buffer before writing back.

D. Rearrangement Module Design

Despite the capability of BSTC in IV-C to facilitate the acceleration of fixed- and floating-point bit-sparse MMA operations, we discover that a primary bottleneck limiting the compute performance is the bit imbalance in the elements of matrix A. Apparently, the speed of BSDP operation executed in a BSDP Unit shown in Fig. 5(b) is constrained by the element a_0 including the most 1-bits in the vector. Likewise, in the BSDP Array, there is the same barrier over the entire matrix A. Moreover, due to the random matrix distribution, there is usually at least one bit-dense element in the matrix, significantly decreasing the speed of the BSDP Array.

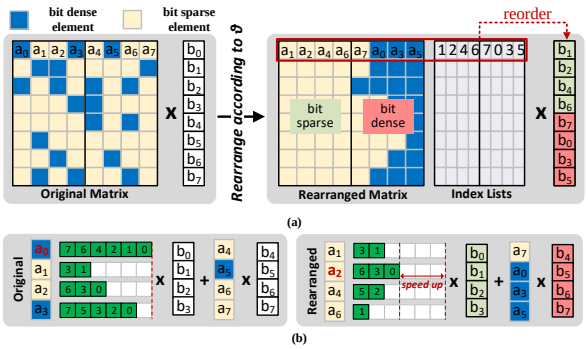


Fig. 8. Details of: (a) The Rearrangement scheme. (b) Speedup.

To settle the imbalance problem, as described in Fig. 8, we take twice the length of the BSDP Unit's vector as a unit to organize the rows of the matrix into bit-sparse and bit-dense sections, which respectively correspond to the mapping size stated in IV-B. Specifically, we sample a few elements from matrix A to roughly predict the sparsity of the matrix, which serves as a threshold θ for data rearrangement. Next, we compare the sparsity of each value against θ, thus creating a bit-sparse balanced matrix. In the given example, we rearrange the elements from a_0, a_1, a_2, a_3 into a_1, a_2, a_4, a_6. As a result, the rearrangement module reduces the cycle from 6 to 3. Moreover, to correctly realize MMA based on the rearranged A matrix, we maintain an index list, stored in Idx Buffer, as the reference to reorder the B matrix.

V. Evaluation

A. Methodology

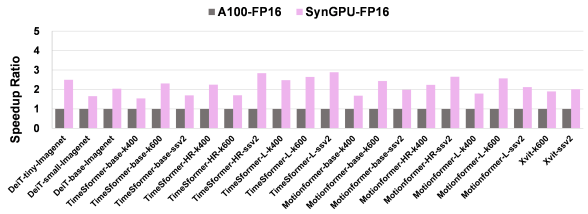
Software implementation. To verify the effectiveness of the SynGPU framework, we select DeiT [2] with ImageNet2021 [14] dataset as the representative of ViTs in image tasks and select TimeSformer [4], Motionformer [15], and Xvit [16] with Kinetics-400 (K400), Kinetics-600 (K600) [17], and SSv2 [18] datasets as the representative of ViTs in video tasks. All models are implemented using the official source code and pre-trained models without fine-tuning. Moreover, the interval used in the IBA algorithm is 80 and the batch size is set as 8 in the following experiments.

Hardware implementation. For kernel-level evaluation, we set the NVIDIA A100 GPU as the baseline. We employ Accel-Sim [19], an open-source cycle-accurate simulator, with integrating SynGPU model to simulate the computing performance in FP16 and INT8 format of A100 and SynGPU. We implement the proposed SynGPU architecture in Verilog and synthesize it by Synopsys Design Compiler to get the area and power consumption under 28nm technology. Given the absence of public data on the A100 Tensor Core area, and considering that the A100 utilizes a 7nm process while our study is based on a 28nm process, we follow work [20] to estimate A100's area and power under 28nm technology and 1.41GHz frequency. For SRAM-based on-chip buffers, we use CACTI 7 [21] to model their area and power consumption under 32nm technology and scale them to 28nm using DeepScaleTool [22].

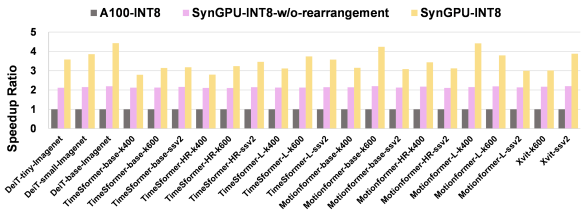
B. Experiment

Speedup. We compare the performance between A100 Tensor Core and SynGPU BSTC on various models and datasets in FP16 and INT8 formats respectively, as shown in Fig. 9(a) and Fig. 9(b). Due to the fact that the A100 Tensor Core does not skip any 0-bits, it consumes the same number of cycles when processing tensors of the same data dimension. On the other hand, SynGPU BSTC, equipped with the capability to accelerate bit-sparse MMA, achieves varying speedup compared to A100 depending on the bit sparsity of the attention map for different input data. The average speedup achieved by SynGPU-FP16 and SynGPU-INT8-w/o-rearrangement is 2.2× and 2.15× respectively, with peak speedup ratios of 2.89× and 2.20×, which do not reach the ideal speedup according to the bit-sparsity ratio indicated in Fig 2. The reason is that BSTC cannot utilize bit-sparsity with maximum efficiency without rearrangement. Therefore, based on the INT8 format, we conduct ablation experiments to demonstrate that rearrangement effectively leverages the bit sparsity acceleration capability of BSTC, providing a 3.38× speedup on average.

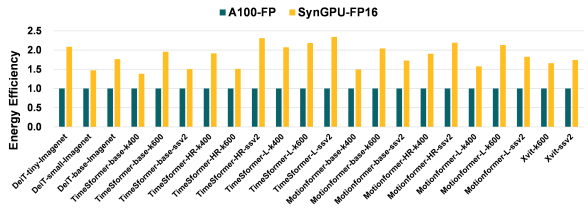
Energy Efficiency. Fig. 9(c) and Fig. 9(d) illustrate the energy efficiency results. Compared to the A100 GPU, SynGPU delivers an average of 1.84×, 1.85×, and 2.68× energy efficiency improvements in FP16, INT8-w/o-rearrangement, and INT8 situations. The optimization in FP16 and INT8 without rearrangement (INT8-w/o-rearrangement) attributed to skipping unnecessary 0-bit calculations using SynGPU,



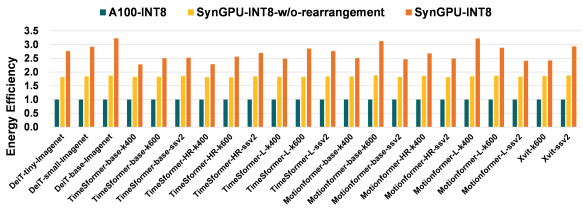
(a) Speedup in FP16



(b) Speedup in INT8



(c) Energy efficiency in FP16



(d) Energy efficiency in INT8

Fig. 9. Speedup and energy efficiency of SynGPU over A100 in FP16 and INT8 formats

thereby saving the dynamic on-chip energy. For INT8 with rearrangement, the enhancement is achieved by utilizing a rearrangement module to accelerate computation cycles, which in turn reduces the static on-chip energy.

TABLE I
AREA AND POWER OF SYNGPU ARCHITECTURE.

Parameter	Area(mm^2)	Area Ratio	Power(W)
16x16 Bit-Serial DP Units	0.9306		0.3000
1.25 KB A Buffer	0.0516		0.0358
1KB B Buffer	0.0233	**96.45%**	0.0269
2KB Acc Buffer	0.0857		0.1835
BSDP Array	**1.0913**		**0.5462**
FindExpMax	0.0006		0.0013
Exp-align	0.0003		0.0008
0.3KB Ma Buffer	0.0138	**1.91%**	0.1813
0.15KB Exp Buffer	0.0069		0.1813
Floating-point Pre-processor	**0.0216**		**0.3647**
Bit-Serial Encoder	**0.0019**	**0.16%**	**0.0430**
Comparators and selectors	0.0098		0.0225
0.15KB IdxList Buffer	0.0069	**1.48%**	0.1813
Rearrangement Module	**0.0167**		**0.2039**

Hardware Overhead and Area. Table I provides an overview of the hardware configurations and the area and power of the BSTC in each SM of SynGPU. The primary overhead of the BSTC is the BSDP Array, with an area of 1.0913mm^2. The other modules, the Floating-point Pre-processor, the Bit-Serial Encoder, and the Rearrangement Module with areas of 0.0216mm^2, 0.0019mm^2, and 0.0167mm^2, respectively, introduce about 1.91%, 0.16%, and 1.48% overhead of the entire Tensor Core, which are negligible compared to the area of the BSDP Array that occupies 96.45% of the chip area. Notably, the Rearrangement Module brings considerable speedup effects with little hardware overhead.

Compute Performance. Table II presents the compute performance of A100 Tensor Core and SynGPU BSTC in FP16 and INT8 formats. Despite the area of BSTC in SynGPU introduces a slight area overhead compared to the Tensor Core in A100 (1.13 mm^2 v.s. 0.98 mm^2), the peak computation performance of SynGPU has a significant advantage, reaching 901 TFLOPs in FP16 and 2761 TOPs in INT8, compared to 312 TFLOPs in FP16 and 624 TOPs in INT8 on A100. Thus, SynGPU achieves 2.49× and 3.81× improvements of compute density in FP16 and INT8 over A100, respectively under the same area budget. Moreover, according to the power simulation of computation modules given in Table I, we obtain the power efficiency of SynGPU for the FP16 and INT8 formats, which reaches 7.69 TFLOPs/W and 23.57 TOPs/W, respectively, delivering 2.58× and 3.95× increase over A100. Due to the acceleration and energy reduction arising from bit-level sparsity, SynGPU achieves substantial advancements in both compute density and compute power efficiency.

TABLE II
COMPUTE PERFORMANCE OF A100 AND SYNGPU.

Devices	Peak Perf.	Area Per SM	Compute Density	Power Efficiency
A100-FP	312 TFLOPs	0.98 mm^2	2.96 TFLOPs/mm^2	2.98 TFLOPs/W
A100-INT8	624 TOPs	0.98 mm^2	5.93 TOPs/mm^2	5.96 TOPs/W
SynGPU-FP	901 TFLOPs	1.13 mm^2	7.37 TFLOPs/mm^2	7.69 TFLOPs/W
SynGPU-INT8	2761 TOPs	1.13 mm^2	22.60 TOPs/mm^2	23.57 TOPs/W

VI. CONCLUSION

In this paper, we introduce SynGPU, an innovative algorithm-hardware co-design framework that synergizes CUDA Cores and Tensor Cores on GPUs to exploit and harness the bit-level sparsity among similar tokens, achieving significant performance enhancement of ViTs. SynGPU tackles two crucial challenges: firstly, the low parallelism caused by competitive access to the RF by CUDA Cores and Tensor Cores; secondly, the inability of tensor cores to accelerate bit-level sparse MMA due to differing exponents in floating-point vectors. Our innovative scheme fully exploits the bit sparsity of ViTs for acceleration, yielding remarkable outcomes without incurring accuracy loss.

References

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

[2] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *International conference on machine learning*. PMLR, 2021, pp. 10 347–10 357.

[3] D. Wodajo, "Deepfake video detection using convolutional vision transformer," *arXiv: Computer Vision and Pattern Recognition,arXiv: Computer Vision and Pattern Recognition*, Nov 2020.

[4] G. Bertasius, H. Wang, and L. Torresani, "Is space-time attention all you need for video understanding?" in *ICML*, vol. 2, no. 3, 2021, p. 4.

[5] H. You, Z. Sun, H. Shi, Z. Yu, Y. Zhao, Y. Zhang, C. Li, B. Li, and Y. Lin, "Vitcod: Vision transformer acceleration via dedicated algorithm and accelerator co-design," in *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2023, pp. 273–286.

[6] Y. Rao, W. Zhao, B. Liu, J. Lu, J. Zhou, and C.-J. Hsieh, "Dynamicvit: Efficient vision transformers with dynamic token sparsification," *Advances in neural information processing systems*, vol. 34, pp. 13 937–13 949, 2021.

[7] P. Dong, M. Sun, A. Lu, Y. Xie, K. Liu, Z. Kong, X. Meng, Z. Li, X. Lin, Z. Fang *et al.*, "Heatvit: Hardware-efficient adaptive token pruning for vision transformers," in *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2023, pp. 442–455.

[8] Z. Li, M. Sun, A. Lu, H. Ma, G. Yuan, Y. Xie, H. Tang, Y. Li, M. Leeser, Z. Wang *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)*. IEEE, 2022, pp. 109–116.

[9] A. Vaswani, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017.

[10] J. Choquette, W. Gandhi, O. Giroux, N. Stam, and R. Krashinsky, "Nvidia a100 tensor core gpu: Performance and innovation," *IEEE Micro*, vol. 41, no. 2, pp. 29–35, 2021.

[11] J. Albericio, A. Delmás, P. Judd, S. Sharify, G. O'Leary, R. Genov, and A. Moshovos, "Bit-pragmatic deep neural network computing," in *Proceedings of the 50th annual IEEE/ACM international symposium on microarchitecture*, 2017, pp. 382–394.

[12] A. Delmas Lascorz, P. Judd, D. M. Stuart, Z. Poulos, M. Mahmoud, S. Sharify, M. Nikolic, K. Siu, and A. Moshovos, "Bit-tactical: A software/hardware approach to exploiting value and bit sparsity in neural networks," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2019, pp. 749–763.

[13] H. Lu, L. Chang, C. Li, Z. Zhu, S. Lu, Y. Liu, and M. Zhang, "Distilling bit-level sparsity parallelism for general purpose deep learning acceleration," in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, 2021, pp. 963–976.

[14] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[15] M. Patrick, D. Campbell, Y. Asano, I. Misra, F. Metze, C. Feichtenhofer, A. Vedaldi, and J. F. Henriques, "Keeping your eye on the ball: Trajectory attention in video transformers," *Advances in neural information processing systems*, vol. 34, pp. 12 493–12 506, 2021.

[16] A. Bulat, J. M. Perez Rua, S. Sudhakaran, B. Martinez, and G. Tzimiropoulos, "Space-time mixing attention for video transformer," *Advances in neural information processing systems*, vol. 34, pp. 19 594–19 607, 2021.

[17] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev *et al.*, "The kinetics human action video dataset," *arXiv preprint arXiv:1705.06950*, 2017.

[18] R. Goyal, S. Ebrahimi Kahou, V. Michalski, J. Materzynska, S. Westphal, H. Kim, V. Haenel, I. Fruend, P. Yianilos, M. Mueller-Freitag *et al.*, "The" something something" video database for learning and evaluating visual common sense," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5842–5850.

[19] M. Khairy, Z. Shen, T. M. Aamodt, and T. G. Rogers, "Accel-sim: An extensible simulation framework for validated gpu modeling," in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2020, pp. 473–486.

[20] Z. Mo, L. Wang, J. Wei, Z. Zeng, S. Cao, L. Ma, N. Jing, T. Cao, J. Xue, F. Yang *et al.*, "Lut tensor core: Lookup table enables efficient low-bit llm inference acceleration," *arXiv preprint arXiv:2408.06003*, 2024.

[21] R. Balasubramonian, A. B. Kahng, N. Muralimanohar, A. Shafiee, and V. Srinivas, "Cacti 7: New tools for interconnect exploration in innovative off-chip memories," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 14, no. 2, pp. 1–25, 2017.

[22] S. Sarangi and B. Baas, "Deepscaletool: A tool for the accurate estimation of technology scaling in the deep-submicron era," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2021, p. 1–5. [Online]. Available: http://dx.doi.org/10.1109/iscas51556.2021.9401196