MCTA: A Multi-stage Co-optimized Transformer Accelerator with Energy-Efficient Dynamic Sparse Optimization

Heng Liu, Ming Han, Jin Wu, Ye Wang, Jian Dong*
Harbin Institute of Technology, Harbin, China
hengliu@stu.hit.edu.cn, minghan@hit.edu.cn, wujin@hit.edu.cn, yewang@hit.edu.cn, dan@hit.edu.cn

***Abstract*—As Transformer-based models continue to enhance service quality across various domains, their intensive computational requirements are exacerbating the AI energy crisis. Traditional energy-efficient Transformer architectures primarily focus on optimizing the Attention stage due to its high algorithmic complexity ($O(n^2)$). However, linear layers can also be significant energy consumers, sometimes accounting for over 70% of total energy usage. Although existing approaches such as sparsity have improved the Attention stage, the optimization space within linear layers is not fully exploited. In this paper, we introduce the multi-stage co-optimized Transformer accelerator (MCTA) for optimizing energy efficiency. Our approach independently enhances the Query-Key-Value generation, Attention, and Feed-forward Neural Network stages. It employs two novel techniques: Low-overhead Mask Generation (LMG) for dynamically identifying unimportant calculations with minimal energy costs, and Cascaded Mask Derivation (CMD) for streamlining the mask generation process through parallel processing. Experimental results show that MCTA achieves an average energy reduction of 1.48× with only a 1% accuracy loss compared to state-of-the-art accelerators. This work demonstrates the potential for significant energy savings in Transformer models without the need for retraining, paving the way for more sustainable AI applications.**

***Index Terms*—approximate computing, dynamic sparse attention, energy-efficient design, transformer accelerator**

I. Introduction

The Attention mechanism in Transformer models significantly enhances performance across various AI application domains, including Natural Language Processing [4], [11], [22], Computer Vision [7], [25] , and Automatic Speech Recognition [21], among others [10], [12]. However, underpinning these transformer-based applications are their intensive computations and high energy costs. As generative AI models such as GPT [32] and LLama [15] continue to evolve, their energy consumption is expected to exacerbate the AI energy crisis. This escalating energy consumption has become a critical concern for AI applications.

An efficient principle for designing energy-friendly Transformer accelerator is to reduce the number of arithmetic operations based on the idea of approximate computing [8]. Utilizing the Transformer's inherent resistance against computation error, a designer can achieve energy saving by omitting those uncritical operations with little impact on the model's inference performance. A Transformer model comprises the Query-Key-Value (QKV) generation, Attention, and Feed-forward Neural Network (FFN) stages. In this architecture, both the QKV generation and FFN are implemented as linear layers [26]. Researches in [1], [9] exploited the redundancy brought by the `SoftMax()` function in the Attention stage and significantly reduced its arithmetic operations. The authors in [24] underscore the significance of runtime information in approximate design, and proposed a FFN-based classifier to categorize neurons into clusters with various critical levels. The neurons whose critical level exceeds a predefined threshold will be selected while others will be omitted.

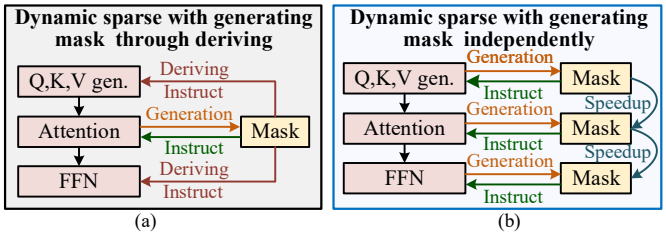

Fig. 1. Comparison of design concept between MCTA and previous accelerators.(a) Dynamic sparse with generating mask through deriving. (b) Dynamic sparse with generating mask independently.

State-of-the-art research in FACT [19] and Ayaka [18] has highlighted the necessity of designing full-process optimized accelerators with customized energy-friendly strategies for each stage, rather than focusing solely on a specific component of the entire model. As shown in Fig. 1(a), this framework first adopts the dynamic sparse attention (DSA) mechanism to generate a mask that identifies the less important computations in Attention. This mask is then further used to derive masks for instructing the calculations of QKV generation and the FFN. Compared to traditional approaches, this framework achieves higher energy efficiency. However, the authors did not account for the differences between QKV generation, FFN, and Attention, which may lead to misclassification of computations in the linear layers and an inadequate exploration of the energy optimization space.

To fully leverage the optimization capabilities of the dynamic sparse method, an effective improvement is to generate masks for the other stages, similar to those used in Attention. By designing approximation strategies tailored to each component independently, this approach can thoroughly explore the optimization space for each component. However, it faces two main challenges. Firstly, a multi-stage mask generation process

978-3-9826741-0-0/DATE25/© 2025 EDAA

will require additional computation. Secondly, due to data dependencies between layers, the mask generation process for a subsequent layer cannot commence until the corresponding source data computations are completed. This will negatively impact the model's execution time.

To address these challenges and optimize the QKV generation and FFN stages alongside Attention, we present the multi-stage co-optimized Transformer accelerator, MCTA, which mitigates the introduced computational and time overhead. Unlike previous works, MCTA optimizes the three stages in the Transformer by generating masks for each component independently to identify the unimportant calculations in all three stages, as shown in Fig. 1(b). To overcome the aforementioned challenges, we have designed two corresponding novel techniques: Low-overhead Mask Generation (LMG) and Cascaded Mask Derivation (CMD). First, LMG generates the mask that identifies the locations of unimportant calculations in a more approximate manner using a low-overhead arithmetic unit, thus reducing the computational overhead introduced by identification. Second, CMD utilizes the information from the generated masks of prior matrix multiplications to accelerate the current mask generation process, making it cascaded and independent from the current input, thereby enabling parallel processing. Experiments show that MCTA achieves an average energy consumption reduction of 1.48× for inference tasks with only a 1% accuracy loss compared to the state-of-the-art accelerator.

Our main contributions are summarized below:

- We propose MCTA, a multi-stage co-optimization approach which optimizes all the three stages of the Transformer with no necessity for model retraining.
- We introduce a novel mask generation method which identifies the position of unimportant calculations with low energy consumption, allowing us to omit less important computations.
- We design a new technique to enable parallel mask generation for multiple layers, mitigating the time overhead of multi-stage co-optimization.
- We implemented a prototype of MCTA, a Transformer accelerator co-designed with the proposed optimization scheme. Evaluations show that it significantly reduces the energy consumption of Transformer models.

II. Related Work

Existing research on reducing the computational load of Transformer can be categorized into three primary areas: Attention-Oriented Optimization, FFN-Oriented Optimization, and Overall Transformer-Oriented Optimization.

A. Attention-Oriented Optimization

A substantial body of work has focused on optimizing Attention computations due to their $O(n^2)$ complexity. As n increases, the computational burden grows significantly. The presence of `SoftMax()` introduces a larger space for approximation.

Common strategies to reduce complexity include dimension reduction techniques such as low-rank Attention [13], [20], [33] and linear Attention [2], [3], [17]. Additionally, some methods exploit redundancy in the Attention mechanism by identifying and skipping non-essential computations or performing them approximately. These techniques can be categorized based on how the sparse patterns are generated: static sparse Attention [1], [6], [31] and dynamic sparse Attention [8], [9], [28], [30]. In static sparse Attention, sparse patterns are predefined.

B. FFN-Oriented Optimization

Recognizing the substantial computational cost of the FFN component in practical applications, researchers have proposed several optimization techniques. For example, neurons in the FFN can be grouped into clusters based on their activation frequency before inference [24]. During computation, frequently activated clusters are dynamically selected for processing.

Matrix factorization is another method used to reduce redundancy in the hidden-to-output projection matrix of the FFN [29]. Since the FFN is not unique to Transformers, similar optimization techniques are applicable to other neural networks as well. But there is limited research specifically targeting FFN optimization within Transformers.

C. Overall Transformer-Oriented Optimization

To achieve comprehensive optimization of Transformers, both Attention and FFN computations must be addressed. Simply combining optimization methods for the two components poses challenges in balancing accuracy and approximation due to differing optimization philosophies. Existing research [18], [19] begins with optimization on Attention. After getting the mask reflecting the position of unimportant calculation through DSA, they statistic the sparsity in the mask to invert the calculations can be skipped in QKV generation. Then, they select relatively unimportant tokens by counting the times each token is selected in the mask and perform approximate computing on them in FFN.

III. Background

A. Dynamic Sparse Attention

Dynamic Sparse Attention (DSA) is a traditional approximation technique based on sparse optimization, aimed at enhancing matrix operations within the Attention mechanism. By identifying and selectively executing critical operations while skipping less important ones, DSA effectively sparsifies dense matrix computations, leading to significant energy savings in the attention stage. Fig. 2 illustrates the fundamental working process of DSA. Initially, the input matrix undergoes quantization to produce an approximate matrix multiplication result. By designating the top-k elements in the approximate result as '1' and the remaining elements as '0', a mask matrix is generated. Each element in this mask indicates the importance of the corresponding value in QK^T. If $Mask_{ij}$ is '0', signifying it is unimportant, the corresponding calculation $Q_i \cdot K_j$ is skipped. Compared to other optimization methods for Attention, DSA achieves greater energy savings by precisely identifying unimportant calculations, albeit with the trade-off of introducing additional computational overhead.

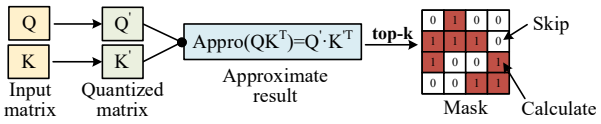

Fig. 2. The mask generation in DSA.

B. Dynamic Sparsity Based Multi-stage Optimization for Transformer

Some research [18], [19] has utilized the sparsity introduced by the mask generated in the Attention mechanism to optimize all three stages of the Transformer model. They begin by applying the DSA technique to optimize the Attention stage. By replacing the original multiplication with a shift operation to calculate the approximate result, they reduce the computational overhead involved in mask generation. After optimizing the Attention mechanism, they analyze the sparsity in the mask to identify which calculations can be skipped during QKV generation. Subsequently, they determine the relatively unimportant tokens by counting the number of '1's for each token in the mask and then perform approximate computing on these tokens in the Feed-Forward Network (FFN). While this approach optimizes all three stages, the derivation of masks for the linear layers from the Attention mask does not fully exploit the redundancy in the linear layers, thereby limiting the overall optimization in these layers.

IV. Motivation

The dynamic sparsity-based optimization has succeeded in the Attention stage by identifying and dropping unimportant elements before matrix multiplications, enhancing matrix sparsity and improving energy efficiency. Extending this approach to the QKV generation and FFN stages can significantly reduce energy consumption across the entire Transformer model. However, this extension introduces challenges. First, extending the scope to more stages requires generating more masks, which may increase computational overhead and offset optimization benefits. Second, mask generation cannot run parallel to other matrix operations, causing considerable time overhead and increased static energy consumption.

Although the overhead of approximate matrix multiplication can be mitigated with more aggressive approximation, there is a trade-off between reducing computational overhead and maintaining accuracy in the results. However, focusing on maintaining the relative magnitude relationships of the precise results can achieve a better balance between efficiency and accuracy. Addressing the time overhead involves resolving data dependencies in mask generation. The primary obstacle is creating the quantized matrix. By leveraging information from previously generated masks, these dependencies can be mitigated, streamlining the process.

To tackle these issues, we propose two novel techniques: Low-overhead Mask Generation (LMG) to reduce computational costs, and Cascaded Mask Derivation (CMD) to address data dependencies and minimize time overhead. These methods aim to effectively reduce the energy consumption of Transformer architectures while maintaining performance.

Algorithm 1 Low-overhead mask generation

Input: $A \in R^{n\times d}$, $B \in R^{n\times d}$, k
Input: $Thre_1$, $Thre_2$,...,$Thre_{m-1}$
Input: $Quan_1$, $Quan_2$,...,$Quan_m$
Output: $mask \in R^{n\times n}$
1: $A'_{ij} \leftarrow Quan_1$, if $A_{ij} > Thre_1$ $Quan_2$, if $A_{ij} > Thre_2$...
2: $B'_{ij} \leftarrow Quan_1$, if $B_{ij} > Thre_1$ $Quan_2$, if $B_{ij} > Thre_2$...
3: **for** each $i \in [0, n-1]$ **do**
4: $\quad sam\_mean \leftarrow Mean(A'_i \cdot B'_l$, if $l\%d == 0)$
5: $\quad sam\_min \leftarrow Min(A'_i \cdot B'_l$, if $l\%d == 0)$
6: $\quad New\_Thre_i \leftarrow (sam\_mean - sam\_min) * k + sam\_min$
7: $\quad$ **for** each $j \in [0, n-1]$ **do**
8: $\quad\quad mask_{ij} \leftarrow 1$, if $A'_i \cdot B'_j > New\_Thre_i$ else 0
9: **return** $mask$

V. Methodology

This section introduces two novel techniques to reduce the computational and time overhead of mask calculations.

A. Low-overhead Mask Generation

The Low-overhead Mask Generation (LMG) method is proposed to reduce the computational overhead associated with the mask generation. Using LMG, a mask is generated by calculating only the relative magnitude information of the matrix multiplication, thereby optimizing both energy consumption and performance. LMG can be divided into three steps: quantization of the matrix, calculation of approximate results, and sample-based generation of the threshold. Algo. 1 illustrates the execution process of LMG.

Quantization of Matrix. To obtain an approximate result, the input is quantized into a set of discrete values: $\{Quan_1, Quan_2, \ldots, Quan_m\}$, through a process of comparison with corresponding thresholds: $\{Thre_1, Thre_2, \ldots, Thre_{m-1}\}$. If the input is larger than $Thre_1$, it is quantized to $Quan_1$. If it is larger than $Thre_2$ but less than $Thre_1$, it is quantized to $Quan_2$, and so on (Lines 1-2 in Algo. 1). This threshold-based quantization process is more hardware-friendly. The number of thresholds and quantized digits can be artificially settled, and the principle for determining such parameters will be further elucidated in Section VI-B.

Calculation of Approximate Results. After quantizing the source data, the number of possible results of multiplications among these source data also narrowing from the entire range values to a determined number of elements. Therefore, the matrix computations in the prediction phase can be viewed as a selecting operation, where all results can be chosen from the candidate set which contains all possible values. The power consumption of the selecting operation is proportional to the size of the candidate set, and can be lower than a single shift or an addition when the candidate set is small. This enables dynamic sparsity for all model stages with minimal computational overhead.

Sample-based Threshold Generation. After obtaining the approximate matrix multiplication result, the corresponding mask is derived based on its value. Directly using the top-k

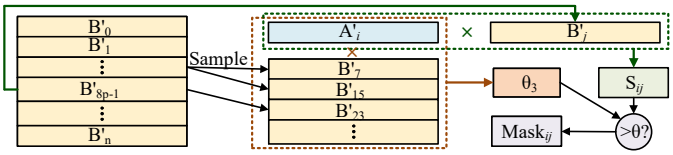

Fig. 3. Sample-based threshold generation.

algorithm to obtain the larger elements from the result matrix is complex and resource-consuming from the hardware aspect. Therefore, we propose a sample-based method to determine the thresholds for comparison, as shown in Fig. 3. We first sample at a certain interval (here $d = 8$), and calculate the average of the sum and minimum of their corresponding results as an approximation of the actual results. The value of k is preset to configure the threshold and the approximate ratio (Lines 3-8 in Algo. 1). When aiming for higher accuracy, k is set to a smaller value. Finally, the critical level of all values in the result matrix can be evaluated based on the obtained threshold.

In the context of Attention calculations, a '0' in the mask indicates that the corresponding calculation on this '0's position can be skipped, while a '1' means that this calculation should be executed accurately. In linear layers, the absence of the `SoftMax()` function causes that entirely omitting relatively unimportant calculations may also introduce a significant degradation on the accuracy. Therefore, in linear layers, the calculations which have $mask_{ij} = 1$ will be also executed accurately, same as in Attention. When $mask_{ij} = 0$, the calculations will be approximately executed instead of being directly dropped.

B. Cascaded Mask Derivation

The Cascaded Mask Derivation (CMD) method is designed for enabling parallel generating masks for multiple layers. Different from traditional methods which derive masks for QKV generation and FFN from the Attention mask, CMD calculates masks for each stage independently. Through leveraging the data passed from the previous stage, CMD avoids the dependence of the mask on the input data of the current layer, allows the mask generation process of the current layer to begin without the completion of the previous layer. What should be noticed is that CMD only takes advantage of the fact that the input for the current multiplication is passed from the previous one. It does not rely on any features of the Attention stage for mask generation. This marks a significant improvement from traditional approaches which utilize the Attention stage's result to generate masks for subsequent stages.

The design of CMD is inspired by the following observation. There are two matrices associated with the output of a matrix multiplication: the mask matrix used to guide the calculation and the quantized matrix, both of which contain overlapping information. The mask determines the significance of each position in the output by comparing the relative sizes of the elements. Similarly, the quantized matrix captures the magnitude relationships between elements at different positions in the output. Thus, the quantized matrix can be derived directly from the mask without the need for additional quantization.

Based on the insight above, the quantized matrix can be obtained by comparing the approximate result with some new

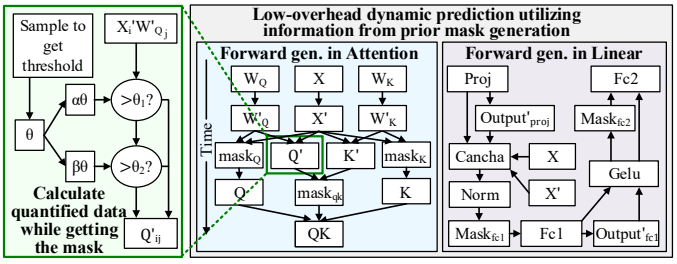

Fig. 4. The Cascaded Mask Derivation of Mask Among Each Layer.

thresholds. The primary distinction between the mask and the quantized matrix is their elements. In the mask, there are only two states, 1 and 0, representing important and unimportant positions, respectively. However, in the quantized matrix, to better preserve the size relationships between the data, multiple states are used to represent each number. Fig. 4 illustrates the methodology for linking the mask generation between layers. As shown on the left of Fig. 4, when using the threshold obtained by sampling to generate the mask, additional thresholds are set to obtain the quantized matrix. The new thresholds are typically set as multiples of the sample-based threshold.

For example, consider two consecutive matrix multiplications: the generation of Q and K, and the QK multiplication. As shown in the center of Fig. 4, the generation of the masks for Q and K also yields their quantized matrices. Therefore, the mask for QK^T can be calculated ahead. This allows QK^T to be performed immediately when Q and K are calculated without waiting for their masks to be generated, which makes the mask generation step imperceptible in terms of time. Therefore, the new matrix multiplication has avoided the limitation caused by data dependency. The result's mask will be no longer dependent on the input data.

To apply CMD to Transformer models, we divide the Transformer architecture into two main components: Attention and Linear, and implement CMD independently to each part. Theoretically, the masks of Q, K, and V in the first layer can be used to infer the masks for other matrix multiplications within current layer and for operations in subsequent layers. However, because these masks are derived from approximate results, the corresponding quantized matrices are also approximate. The impact on the computation accuracy will significantly increase if we let the approximate error propagate. Therefore, in MCTA, not all quantized matrices are derived from previous masks.

For the Linear component, we focus on two main part. Between adjacent linear projections and the first fully connected layer in FNN, denoted as Fc1, there are residual connections and normalization functions. The normalization function does not disrupt the size relationships between the data, so its effect can be ignored in our approach. To leverage the information from the linear projection layer, we perform residual connections on the obtained mask matrix. We then add this mask matrix to the quantized matrix corresponding to the input data, using it as the quantized matrix for the Fc1 input. Between Fc1 and the second fully connected layer in FNN, denoted as Fc2, there is a nonlinear function, `Gelu()`. The `Gelu()` function converts all negative numbers into values

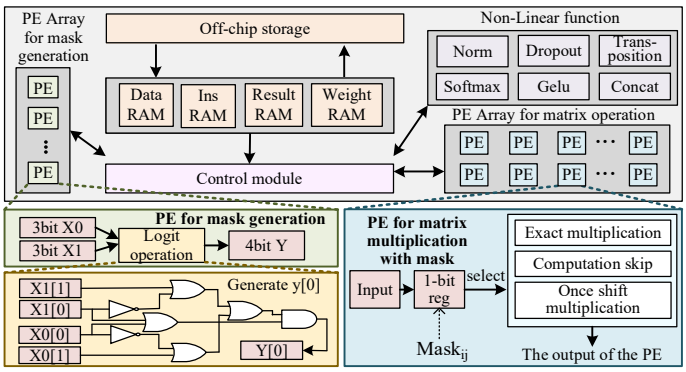

Fig. 5. The Architecture of MCTA.

close to 0, while positive numbers remain almost unchanged. This transformation alters the size relationships between the data. Therefore, we adjust the approximate quantized matrix generated by Fc1 by keeping the positive numbers unchanged and converting the negative numbers into quantized values representing the smallest negative numbers.

VI. Implementation

A. Architecture

Based on the proposed techniques, a prototype of MCTA is implemented, as shown in Fig. 5. MCTA comprises five main modules: on-chip and off-chip storage, a control module, a non-linear functions module, a PE array for matrix operation, and a PE array for mask generation. It performs matrix operations, includes matrix multiplication, matrix addition, and various nonlinear functions such as `SoftMax()` and `Gelu()`.

B. Mask Generation

As described in §V-A, the generation of mask can be divided into three stages. Since the sample based threshold θ can be obtained dynamically within the inference, the subsection focuses on the implementation of the rest two stages: quantization of mask, calculation of approximate results.

Quantization of Matrix. In this paper, the input data is quantized into three numbers 4, 2, and 1, which means m in Algorithm 1 is equal to 3. When m is excessively large, the overhead of the prediction part will increase accordingly; and when m is too small, the quantization operation may not effectively maintain the size relationship in the source data, impacting the final accuracy. The reason behind selecting m as 3 is that it can maintain a good balance between the above two aspects. The selection of these three values is relatively unimportant, provided that their difference is sufficiently substantial.

To complete the quantization, we compare the input data with thresholds. For thresholds in the first matrix multiplication of Attention and FFN, we statistics the average μ of the inputs before the actual inference, and use 3μ and 2μ as thresholds. In the other matrix multiplication of Attention and FFN, the quantization of input data is finished with the generation of its mask. For thresholds $\alpha\theta$ and $\beta\theta$, we adjust the α and β to make the number of data bigger than them account for about 10% and 20% of the total, respectively.

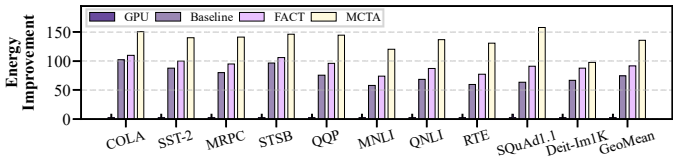

Fig. 6. Energy reduction for inference over GPU, Baseline, and FACT.

Calculation of Approximate Results. After quantization, we utilize the PE array for mask generation as shown on the left side of Fig. 5 to calculate the approximate matrix operation result. Since m is 3, it can be represented by a 3-bit register with the sign bit, and there are five results of multiplication, which can be represented by a 4-bit register with the sign bit. Each bit of the result is obtained from the input register by logic operation. Fig. 5 shows the logic circuit of the generation for the lowest bit of the result $y[0]$. After calculation of the approximate matrix operation, we compare it with New_Thre_i to get the mask matrix.

C. Matrix Multiplication with Mask

When performing matrix multiplication, as shown on the right side of Fig. 5, we select the corresponding computing components for calculation based on the value of $mask_{ij}$ and whether it belongs to the Attention or other Linear layers. For Attention, if $mask_{ij}$ is '0', we directly skip the calculation. For the other linear layers, if $mask_{ij}$ is '0', we use the weights that retain the precious significant bit for calculation.

VII. Evaluation

A. Experimental Setup

We implemented the proposed optimization scheme in PyTorch to evaluate its algorithmic performance. For hardware performance assessment, we developed a prototype of MCTA with INT8 data type using 28nm CMOS technology. The implementation was synthesized in Verilog RTL using Cadence Genus Synthesis Solution 15.20, with the 28nm library at 100MHz. The power and area of the storage module were simulated using CACTI [16]. The overall energy consumption is obtained by adding the overall static energy consumption to the dynamic energy consumption of each part during operation.

We compared MCTA with the state-of-the-art accelerator FACT [19], which is the first Transformer accelerator to optimize all three modules of the Transformer model explicitly. Given FACT's comprehensive advantages over approaches that target only specific stages of the Transformer [9], [14], we omitted comparisons with these works. Additionally, we included the NVIDIA A100 GPU(FP16) as a reference for GPU performance. For the workload, we utilized BERT-base [11] and Deit-base [25], two state-of-the-art Transformer models representing key application domains: Natural Language Processing (NLP) and Computer Vision (CV). For the datasets, we selected the GLUE text classification tasks [27] and the Stanford Question Answering Dataset (SQuAD) v1.1 [23] for NLP, and ImageNet-1k classification tasks [5] for CV. Both datasets are well-recognized and widely used in their respective domains. We sourced the pre-trained parameters for each model and dataset from Huggingface.

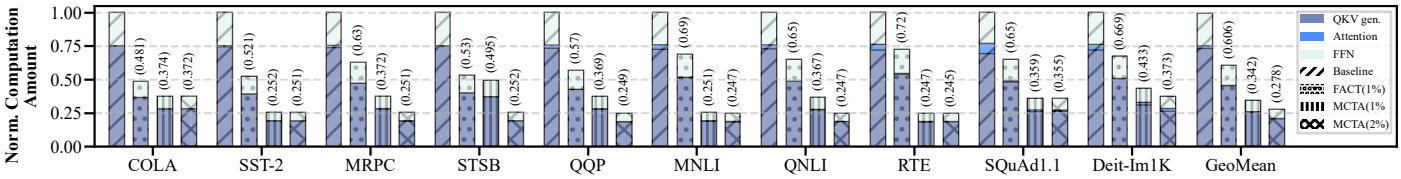

Fig. 7. Computation reduction of MCTA at varying accuracy levels compared to the vanilla Transformer and FACT. Accuracy levels are indicated as percentages in the legend.

B. Energy Consumption

We evaluated the overall energy consumption of MCTA, as depicted in Fig. 6. This figure illustrates the energy consumption differences between MCTA, FACT, and GPU, with the baseline representing the accelerators without optimization. Compared to FACT, our accelerator achieved an average energy consumption reduction of 1.48× for reasoning tasks, while for GPU, the reduction is 135×. The energy reduction was distributed as follows: memory 53.7%, QKV generation 10.1%, Attention 5.6%, and FFN 30.6%. Although we only reduce the size of weights in memory, the ahead generation of masks reduces the time consumption, thus reducing the static energy consumption in memory. Since memory occupies a considerable proportion of energy consumption, it accounts for nearly 50% of the overall energy reduction. Additionally, our reduction in computation also accounts for nearly 50% of the overall energy reduction. The energy efficiency of our accelerator is 6,682 GOPS/W, which is 1.52× higher than that of FACT.

C. Calculation Reduction

We conducted a statistical analysis to evaluate how our method reduces the computational burden in each model under varying degrees of accuracy loss and compared it with FACT, as illustrated in Fig. 7. On average, our method can reduce computation by 65.7% and 72.1% with 1% and 2% accuracy loss, respectively. The average reduction ratio for the three components of the model is 65.0%, 80.9%, and 65.3% with a 1% accuracy loss. When optimizing the entire model using the dynamic sparsity method, we reduced the total computational cost for mask generation from 10% (when using shift operations) to 2% with LMG. For some datasets, the difference in the amount of computation that can be reduced when the accuracy drops by 1% compared to 2% is minimal. This phenomenon occurs because these datasets reach a certain threshold of approximation at a 1% accuracy drop, and further increasing the degree of approximation slightly results in a significant accuracy decline.

D. Latency Improvement

Fig. 8 illustrates the end-to-end latency improvement of the MCTA accelerator. Compared to FACT, we achieve an improvement of 1.43-1.49× and 1.07-1.38× in the 1% and 2% precision drop cases, respectively. Compared to the GPU, MCTA achieves a speedup of 4-5.4×. While the LMG reduces the time required for matrix operations by 65.7%, in the absence of CMD, the overall time overhead would be 1.34× higher than the pre-optimized system. Using CMD, we reduce the time overhead for mask generation to 16.7% of the original.

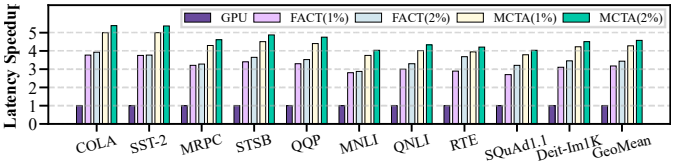

Fig. 8. Latency Speedup compared with GPU and FACT in various accuracy loss. Accuracy levels are indicated as percentages in the legend.

TABLE I
MCTA AREA AND POWER BREAKDOWN.

Module	Parameter	Area[mm^2]	Power[mW]
PE Array	32×32 MAC unit	2.89	113.56
Mask gen.	32×32 specific PE	0.04	2.37
Non-Linear fun.	`Gelu`,`SoftMax`,`Norm`	0.42	1.72
Memory	4×128KB buffer	0.82	52.43
Total	Area = 4.17mm^2, Power = 170.08mW		

E. Area and Power Breakdown

Tab. I presents the area and power consumption of each module within the MCTA accelerator, including the PE array, the mask generation computation component, storage units, and the non-linear function. Given the extensive matrix operations in the Transformer, the PE array for these operations constitutes 69.3% of the total area and 66.7% of the total power consumption. Notably, the computation component for mask generation accounts for only 1.0% of the total area and 1.4% of the total power consumption, representing a minimal overhead.

VIII. Conclusion

The Attention mechanism in Transformer models enhances performance across various AI domains but incurs high computational and energy costs. To address these issues, we introduced the multi-stage co-optimized Transformer accelerator (MCTA), which optimizes the Query-Key-Value generation, Attention, and Feed-forward Neural Network stages independently. MCTA employs Low-overhead Mask Generation (LMG) and Cascaded Mask Derivation (CMD) techniques to reduce energy consumption. Our experiments show that MCTA achieves an average energy reduction of 1.48× with only a 1% accuracy loss compared to existing accelerators. In summary, MCTA offers significant energy savings without compromising performance, marking a substantial step towards energy-efficient Transformer models.

Acknowledgment

We are very grateful to the reviewers for their valuable feedback and comments. This work was supported by the Fundamental Research Funds for the Central Universities (Grant No. HIT.OCEF.2024012).

REFERENCES

[1] Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. Etc: Encoding long and structured inputs in transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 268–284, 2020.

[2] Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. In *International Conference on Learning Representations*, 2020.

[3] Jyotikrishna Dass, Shang Wu, Huihong Shi, Chaojian Li, Zhifan Ye, Zhongfeng Wang, and Yingyan Lin. Vitality: Unifying low-rank and sparse approximation for vision transformer acceleration with a linear taylor attention. In *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 415–428. IEEE, 2023.

[4] Pieter Delobelle, Thomas Winters, and Bettina Berendt. Robbert: a dutch roberta-based language model. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3255–3265, 2020.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[6] Xiaoyi Dong, Jianmin Bao, Dongdong Chen, Weiming Zhang, Nenghai Yu, Lu Yuan, Dong Chen, and Baining Guo. Cswin transformer: A general vision transformer backbone with cross-shaped windows. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12124–12134, 2022.

[7] Benjamin Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. Levit: a vision transformer in convnet's clothing for faster inference. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12259–12269, 2021.

[8] Tae Jun Ham, Sung Jun Jung, Seonghak Kim, Young H Oh, Yeonhong Park, Yoonho Song, Jung-Hun Park, Sanghee Lee, Kyoung Park, Jae W Lee, et al. Aˆ 3: Accelerating attention mechanisms in neural networks with approximation. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 328–341. IEEE, 2020.

[9] Tae Jun Ham, Yejin Lee, Seong Hoon Seo, Soosung Kim, Hyunji Choi, Sung Jun Jung, and Jae W Lee. Elsa: Hardware-software co-design for efficient, lightweight self-attention mechanism in neural networks. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, pages 692–705. IEEE, 2021.

[10] Mingqiang Huang, Junyi Luo, Chenchen Ding, Zikun Wei, Sixiao Huang, and Hao Yu. An integer-only and group-vector systolic accelerator for efficiently mapping vision transformer on edge. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2023.

[11] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, page 2. Minneapolis, Minnesota, 2019.

[12] Bingbing Li, Santosh Pandey, Haowen Fang, Yanjun Lyv, Ji Li, Jieyang Chen, Mimi Xie, Lipeng Wan, Hang Liu, and Caiwen Ding. Ftrans: energy-efficient acceleration of transformers using fpga. In *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*, pages 175–180, 2020.

[13] Liu Liu, Zheng Qu, Zhaodong Chen, Fengbin Tu, Yufei Ding, and Yuan Xie. Dynamic sparse attention for scalable transformer acceleration. *IEEE Transactions on Computers*, 71(12):3165–3178, 2022.

[14] Liqiang Lu, Yicheng Jin, Hangrui Bi, Zizhang Luo, Peng Li, Tao Wang, and Yun Liang. Sanger: A co-design framework for enabling sparse attention using reconfigurable architecture. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 977–991, 2021.

[15] Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. Fine-tuning llama for multi-stage text retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2421–2425, 2024.

[16] Naveen Muralimanohar, Rajeev Balasubramonian, and Norman P Jouppi. Cacti 6.0: A tool to model large caches. *HP laboratories*, 27:28, 2009.

[17] Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah Smith, and Lingpeng Kong. Random feature attention. In *International Conference on Learning Representations*, 2020.

[18] Yubin Qin, Yang Wang, Dazheng Deng, Xiaolong Yang, Zhiren Zhao, Yang Zhou, Yuanqi Fan, Jingchuan Wei, Tianbao Chen, Leibo Liu, et al. Ayaka: A versatile transformer accelerator with low-rank estimation and heterogeneous dataflow. *IEEE Journal of Solid-State Circuits*, 2024.

[19] Yubin Qin, Yang Wang, Dazheng Deng, Zhiren Zhao, Xiaolong Yang, Leibo Liu, Shaojun Wei, Yang Hu, and Shouyi Yin. Fact: Ffn-attention co-optimized transformer architecture with eager correlation prediction. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, pages 1–14, 2023.

[20] Zheng Qu, Liu Liu, Fengbin Tu, Zhaodong Chen, Yufei Ding, and Yuan Xie. Dota: detect and omit weak attentions for scalable transformer acceleration. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 14–26, 2022.

[21] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In *International conference on machine learning*, pages 28492–28518. PMLR, 2023.

[22] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[23] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2016.

[24] Jie Tang, Shuai Wang, Song Chen, and Yi Kang. Dp-ffn: Block-based dynamic pooling for accelerating feed-forward layers in transformers. In *2024 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE, 2024.

[25] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021.

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

[27] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[28] Hanrui Wang, Zhekai Zhang, and Song Han. Spatten: Efficient sparse attention architecture with cascade token and head pruning. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 97–110. IEEE, 2021.

[29] Haiyang Xu, Zhichao Zhou, Dongliang He, Fu Li, and Jingdong Wang. Vision transformer with attention map hallucination and ffn compaction. *arXiv preprint arXiv:2306.10875*, 2023.

[30] Haoran You, Zhanyi Sun, Huihong Shi, Zhongzhi Yu, Yang Zhao, Yongan Zhang, Chaojian Li, Baopu Li, and Yingyan Lin. Vitcod: Vision transformer acceleration via dedicated algorithm and accelerator co-design. In *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 273–286. IEEE, 2023.

[31] Pengchuan Zhang, Xiyang Dai, Jianwei Yang, Bin Xiao, Lu Yuan, Lei Zhang, and Jianfeng Gao. Multi-scale vision longformer: A new vision transformer for high-resolution image encoding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2998–3008, October 2021.

[32] Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and William B Dolan. Dialogpt: Large-scale generative pre-training for conversational response generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 270–278, 2020.

[33] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021.