AttBind: Memory-efficient Acceleration for Long-range Attention using Vector-derived Symbolic Binding

Weihong Xu, Jaeyoung Kang, and Tajana Rosing
University of California San Diego, La Jolla, CA 92093, USA
Email: {wexu, j5kang, tajana}@ucsd.edu

***Abstract*—Transformer models have achieved a number of breakthrough results in a variety of complex tasks. Transformer's promising performance originates from multi-head attention (MHA), which can model long-range sequence data dependency. Better performance has been demonstrated to be obtained by increasing the sequence length N. However, scaling up the sequence length is extremely challenging for memory-constrained hardware because the naive Transformer requires quadratic $\mathcal{O}(N^2)$ complexity. In this work, we address this challenge by leveraging the binding operation in vector symbolic architecture (VSA). We propose the memory-efficient MHA algorithm to simplify the MHA computation at the cost of linear complexity. Then, we present the ASIC hardware architecture with optimized timing and dataflow to accelerate the proposed algorithm. We extensively evaluate our design across various long-range attention tasks. Our experiments show that the accuracy is competitive to state-of-the-art MHA optimization approaches with lower memory consumption and inference latency. The proposed algorithm achieves $7.8\times$ speedup and $4.5\times$ reduction in data movement over the naive Transformer on ASIC. Meanwhile, our design supports 8 to 16$\times$ sequence lengths compared to existing hardware accelerators.**

***Index Terms*—Transformer, multi-head attention, vector symbolic architecture, accelerator.**

I. Introduction

Transformer is one of the most important backbones of deep learning in recent years. There have been various models built upon Transformer that have achieved significant accuracy improvements for various important machine learning tasks, such as natural language processing [1], computer vision [2], and video analysis. The secret of Transformer's powerful performance is the multi-head attention (MHA) mechanism that can accurately model long-range data dependency. Previous works [1, 3] show that Transformer performance can be further enhanced by increasing the input sequence length.

However, MHA turns out to be a major bottleneck when scaling the sequence length N because of the quadratic $\mathcal{O}(N^2)$ memory and computation complexity. This memory constraint hinders the further improvement of Transformer to long-sequence tasks. Previous Transformer models and hardware accelerators [4–6] have mainly been designed for relatively short sequences $<$1K. Although previous works, like SpAttn [4], utilize sorting and sparsity to reduce the memory footprint during inference, the area constraint and limited on-chip memory of hardware, such as ASIC, still limit the scaling of MHA. Therefore, a hardware design that supports long-range attention $>$1K length is needed.

On the other hand, various works [3, 7–9] have tried to devise a more memory-efficient attention mechanism at the algorithm level. These works present various strategies to avoid the quadratic computational cost resulting from the MHA module when processing input sequences. But these works still require additional overhead to process the MHA module. For example, Hrrformer [3] requires fast Fourier transform (FFT) while Nyströmformer [10] requires additional convolution operations, which both complicate the dataflow. The other drawback of previous works [7–9] are the inferior accuracy performance on long-range tasks [10].

In this paper, we address the mentioned challenges by utilizing the compact data structure and advanced computing paradigm in emerging vector symbolic architecture (VSA) [11]. The contributions are summarized as follows:

- We propose a software and hardware co-design, AttBind, to accelerate long-range multi-head attention (MHA) in Transformer [1]. By utilizing the VSA binding operation [3, 11], we first propose the memory-efficient VDS-based MHA algorithm with only $\mathcal{O}(ND\sqrt{D})$ complexity.
- According to our evaluation on the Long Range Arena benchmark [12], our algorithm provides competitive accuracy compared to the state-of-the-art MHA variants [3, 10], but requires less memory consumption and delivers faster inference speed.
- To further improve inference efficiency, we develop a reconfigurable ASIC accelerator to implement the proposed algorithm. Dataflow optimization is developed to maximize hardware utilization and reduce redundant data movement.
- Experiments show that the AttBind design scales well with the sequence length, delivering $7.8\times$ speedup and $4.5\times$ data movement saving over the naive Transformer. Compared to existing ASICs [4–6], AttBind has comparable hardware performance and efficiency while supporting 8-16$\times$ longer sequences.

II. Preliminary

A. Transformer and Multi-head Attention

Transformer. Transformer has been widely applied to different tasks, including computer vision (CV) and natural language processing (NLP). Transformer has a feedforward structure consisting of multiple encoders. Fig. 1 illustrates an example for the encoder block [1]. The embedding of sentences or other sequence data are first divided into different fragments with

979-8-3503-4859-0/DATE24/© 2024 EDAA

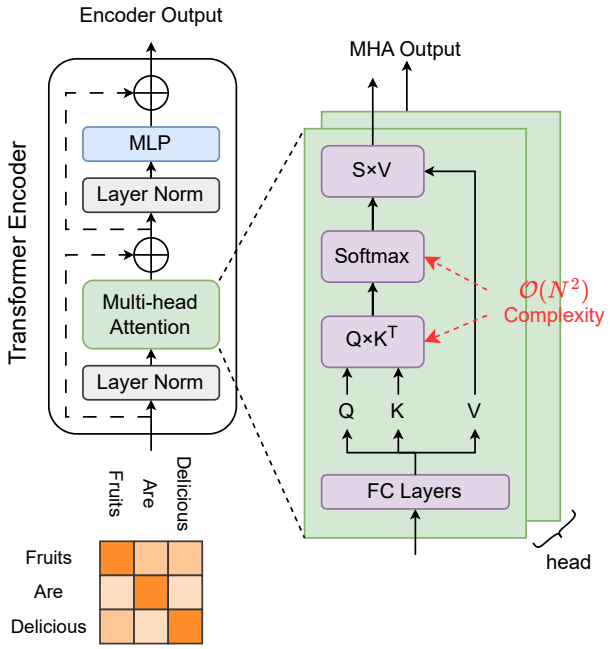


Fig. 1: Transformer and multi-head attention (MHA) module.

the same sequence length N. Then, the input sequence is fed into the Transformer encoder model chunk by chunk ($N = 3$ in Fig. 1). Here, each encoder is composed of three subblocks: layer normalization, multihead attention (MHA), and multilayer perceptron (MLP) module.

Multi-head Attention (MHA). Transformer's promising performance originates from the MHA module [1] because it provides long-range attention capablity. In essence, the MHA module identifies the relations between input sequences. To realize this mechanism, each MHA module has the fully-connected (FC) layer as well as a self-attention layer. For an input sequence with length N, the FC layer receives an embedding matrix of size $N \times H$ (H is the embedding dimension) and generates three different matrices: 1. query Q, 2. key K, and 3. value V using the corresponding weight matrices. The query, key, and value matrices have identical dimensions of $N \times D$, where D denotes the intermediate dimension of the encoder. The encoder block then feeds the Q, K, and V matrices into the self-attention (SA) layer. The original Transformer model uses the scaled dot-product attention [1] as follows:

$$\text{Self-attention}(Q, K, V) = \text{Softmax}(\frac{QK^T}{\sqrt{D}})V, \qquad (1)$$

where we denote $\frac{QK^T}{\sqrt{D}}$ as the pairwise correlation matrix between N input sequences while Softmax$(\cdot)$ is the Softmax function. The normalized Softmax scores $S = \text{Softmax}(\frac{QK^T}{\sqrt{D}})$ are finally multiplied by the value matrix V to generate the MHA output. Without loss of generality, the Transformer adopts the multi-head mechanism to further improve the performance. The difference is that the Q, K and V matrices are uniformly divided into h segments (heads) before being fed into the SA module.

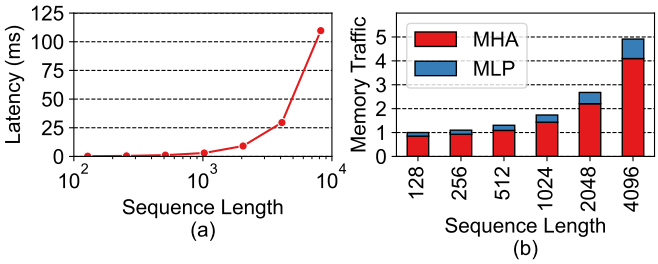


Fig. 2: (a) Increased inference latency and (b) Increased data traffic of MHA as a function of input sequence length.

Challenges. The key factor that determines the performance of the Transformer is the input sequence length used. However, the MHA module in the original Transformer model is a type of memory-intensive workload that imposes several critical challenges. We profile the original Transformer with intermediate dimension D =1K and $h = 4$ heads. Fig. 2 summarizes the profiling results by ranging the input sequence length N from 128 to 4K. The first challenge is the severely degraded inference speed for long sequences. Fig. 2-(a) shows the measured inference latency on the NVIDIA RTX 4090 GPU. We see a growth of nearly quadratic latency when linearly increasing the input sequence length N. This is mainly because the MHA module in Eq. (1) requires quadratic $\mathcal{O}(N^2)$ memory and computational complexity. The slowness effect is much more significant for $N >$1K. For this reason, most models and accelerators [4–6] are only capable of handling sequences of up to 1K.

The other challenge of boosting long-sequence MHA is prohibitive data movement. In Fig. 2-(b), we use the simulator of proposed ASIC design in Section III-B to estimate the data traffic resulting from two key modules in the Transformer encoder: MHA and MLP modules, respectively. The data traffic of the MHA module contributes most of the overall memory footprint. This is due to the fact that the limited on-chip buffer is unable to cache the large $N \times N$ pairwise attention matrix. Additional data movement between the off-chip memory and on-chip buffers is needed.

B. Binding in Vector Symbolic Architecture

Vector symbolic architecture (VSA) is proposed to perform brain-like symbolic processing in cognitive tasks [11]. Two key operations in VSA are *binding* and *unbinding*. The binding operation $\mathcal{B}$ combines abstract concepts into a single numerical vector. This can be expressed by the following equation:

$$\mathcal{B} : \mathbb{R}^d \circ \mathbb{R}^d \rightarrow \mathbb{R}^d, \qquad (2)$$

where the abstract concepts are presented by two high-dimensional vectors $\mathbb{R}^d$. Unbinding is the inverse process of binding but performs the similar operation: $\mathcal{B}^{-1} : \mathbb{R}^d \circ \mathbb{R}^d \rightarrow \mathbb{R}^d$.

Binding and unbind operations in VSA need to be discriminative for most given vectors x, y, which means that the following condition should satisfy:

$$\mathcal{B}^{-1}(\mathcal{B}(x, y), y) \approx x. \qquad (3)$$

The binding and unbinding in VSA allow us to construct compact data presentation by binding multiple high-dimensional vectors to a limited number of vectors, thus significantly reducing memory consumption. The key problem boils down to choosing binding and unbinding methods that can accurately retrieve the bound vectors as in Eq. (3). Previous work [3] adopts circular convolution-based binding, called holographic reduced representations (HRR).

III. Proposed AttBind Design

In this section, we propose a software and hardware co-design by leveraging the vector-derived symbolic (VDS) binding approach in VSA [11]. Section III-A presents the VDS-based attention algorithm with compact data structure and high memory efficiency. In Section III-B, we develop an ASIC accelerator to efficiently implement the proposed VDS-based attention.

A. Memory-efficient Attention using VSA Binding

The previous work [3] uses circular convolution-based binding and unbinding to reduce the memory footprint and speed up MHA. However, this method exhibits several drawbacks: The previous binding based on circular convolution is computationally intensive. Although the fast Fourier transform (FFT) is utilized to realize efficient convolution, FFT and inverse FFT still require additional overhead and memory for conversion.

The selection of VSA binding and unbinding methods should take into account both computing efficiency and retrieval accuracy. We leverage the vector-derived symbolic (VDS) binding and unbinding [11] to alleviate the overhead of FFT while achieving promising accuracy. According to [11], the VDS-based method shows competitive retrieval performance after unbinding. Moreover, the VDS only requires general matrix multiply (GEMM) operations, which are especially hardware-friendly. The VDS-based binding is given by:

$$\mathcal{B}(x, y) = x \circ y = \frac{1}{\sqrt{d}} \left(I \otimes \text{mat}(x)\right) \cdot y, \tag{4}$$

where $\otimes$ denotes the Kronecker product operator while I is the identity matrix. $\text{mat}(\cdot)$ denotes the reshape operation to convert a d^2-dimensional vector into the square-shape matrix with size $d \times d$. According to [11], the VDS-based unbinding follows a similar computation as in Eq. 4.

Fig. 3 shows the dataflow comparison for computing the naive attention [1] and the proposed VDS-based attention. The overall flow of the proposed VDS-based attention is similar to that of Hrrformer [3]. The main differences lie in two factors: 1. We use more efficient VDS-based binding and unbinding. 2. The inner product is adopted as the metric instead of Cosine similarity. The first step is to bind the K and V matrices to one $\mathbf{KV}$ matrix. Without loss of generality, we extend the binding definition in Eq. 4 from two vectors to two matrices as given by:

$$\mathbf{KV} = \mathcal{B}(K, V) = \sum_i K_i \circ V_i, \tag{5}$$

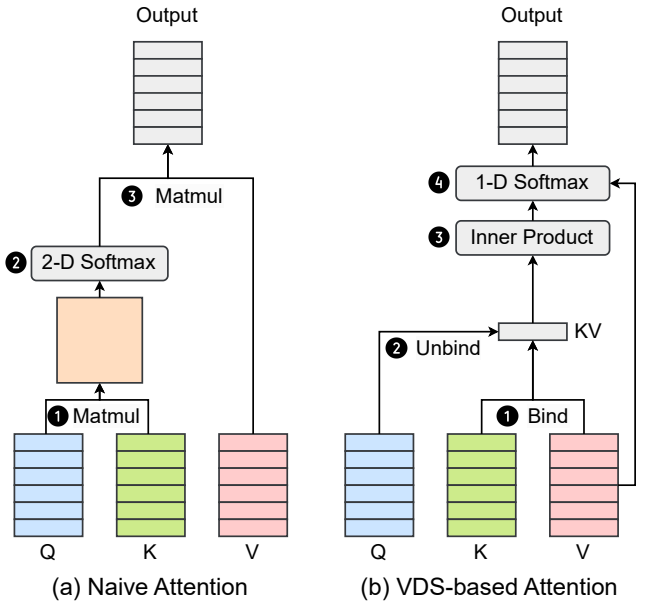


Fig. 3: Comparison for (a) naive attention [1] and (b) VDS-based attention.

where the VDS-based binding is applied to each pair of row vectors when the inputs are two matrices, and then summation is performed along the row dimension. In this way, we can generate the compact vector $\mathbf{KV}$.

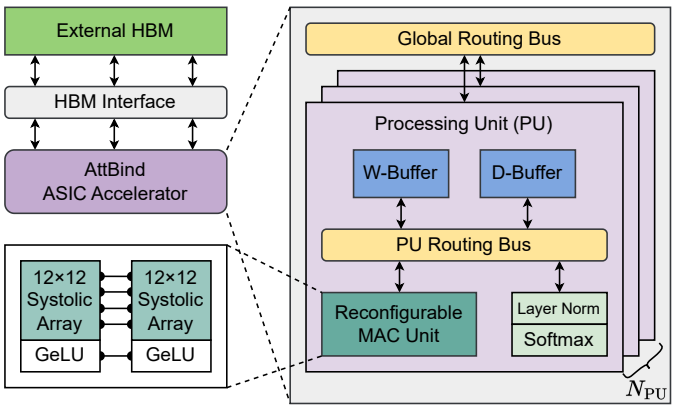
The second step is to use the query Q to retrieve the bound $\mathbf{KV}$ based on the unbind operation as $\mathcal{B}^{-1}(Q_j, \mathbf{KV})$. Unbinding results are measured against the original query Q using the inner product (IP) similarity metric in the 3rd step. As a result, the similarity sim_j for j-th row vector of matrix Q is given by:

$$sim_j = \mathcal{B}^{-1}(Q_j, \mathbf{KV}) \cdot Q_j. \tag{6}$$

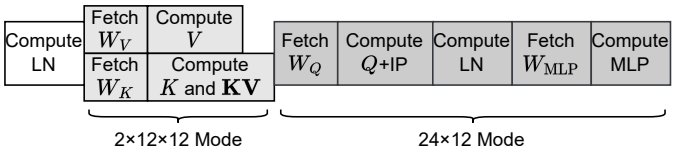
In the 4th step, the generated IP similarity vector from Eq. 6 passes through a one-dimensional (1-D) Softmax function to normalize the similarities. The encoder output is generated by weighting the value matrix V using the Softmax output. After putting Steps 1 to 4 in Fig. 3-(b) together, the proposed VSD-based attention can be expressed as:

$$\text{VSD-attention}(Q, K, V) = \text{Softmax}\left(\mathcal{B}^{-1}(Q, \mathbf{KV}) \cdot Q\right) \cdot V. \tag{7}$$

The proposed VSD-based AttBind attention benefits from the advantages of low memory and computational complexity of VSA. This can be seen from the dataflow comparison in Fig. 3. AttBind and Hrrformer [3] do not directly compute the pairwise attention matrix with $\mathcal{O}(N^2)$ complexity as naive attention [1]. Instead, the key K and value V are first bound together for lower memory complexity. Then the query Q retrieves the corresponding similarities from the bound $\mathbf{KV}$. The required memory and computation complexity of AttBind attention is reduced from original Transformer's $\mathcal{O}(N^2D)$ to only $\mathcal{O}(ND\sqrt{D})$, where N and D denote the sequence length and encoder intermediate dimension, respectively. This property allows us to efficiently scale the model size as well



(a) Overall architecture of AttBind accelerator



(b) Timing diagram

Fig. 4: (a) AttBind dataflow and partial encoding scheme, (b) data organization in PU.

as the sequence length used. It should be noted that although Hrrformer's $\mathcal{O}(ND \log D)$ complexity is theoretically lower than AttBind, the actual performance of Hrrformer is worse than AttBind. This is because AttBind only needs GEMM operations which are more efficient than Hrrformer's FFT (see details in Fig. 5).

B. Hardware Architecture

The optimized VDS-based AttBind attention is memory-efficient and allows us to realize more efficient MHA using limited on-chip buffers. For better efficiency, we choose to implement AttBind on ASIC.

Overview. Previous work [4] demonstrates that Transformer is a memory-intensive workload that requires high memory bandwidth. Hence, we adopt the high-bandwidth memory (HBM) [13] as the external memory. HBM is able to provide over 128GB/s bandwidth for the ASIC accelerator. Fig. 4-(a) shows the overall architecture of AttBind accelerator connected to the off-chip HBM with the HBM interface. The AttBind accelerator is composed of routing buses and N_{PU} processing units (PUs), which are extensible.

Processing unit (PU). The data transfer from the external HBM is distributed to different PUs through the global and PU routing buses. Inside each PU, the PU routing bus connects a reconfigurable MAC unit, a 128KB weight buffer (W-Buffer), a 128kB data buffer(D-Buffer), layer norm (LN), and Softmax module. W-Buffer is used to store the weight matrices for FC or MLP layer while D-Buffer is used to store the input sequence or intermediate data. Different instructions can be loaded into each PU to realize different computation functionalities. The reconfigurable MAC unit consists of two 12×12 systolic arrays with GeLU activation gates. These two systolic arrays can either independently receive two matrix inputs in $2 \times 12 \times 12$ mode or process GEMM for the same matrix in 24×12 cooperative mode. The details are introduced in the following sections.

Softmax approximation. The exact Softmax calculation incurs a large overhead for Transformer inference [4]. To simplify the calculation, we adopt the 2-based Softmax approximation [14] to reduce the circuit complexity. The basic idea is to replace the original exponential function with the two-based function. Given a vector $x \in \mathbb{R}^d$, the approximate Softmax function becomes:

$$\text{Softmax}(x_i) = \frac{2^{x_i}}{\sum_{j=1}^{d} 2^{x_j}}, \tag{8}$$

where the approximation allows us to easily implement the Softmax function using floating-point (FP) adders and dividers, giving better area and timing performance.

Timing and dataflow. Similar to naive Transformer, the VDS-based MHA algorithm has a data dependency between Q, K, and V, as shown in Fig. 3-(b), where K and V will be bound to form the compact $\mathbf{KV}$ and then Q is used to retrieve from $\mathbf{KV}$. Ideally, we need to optimize the data flow to reduce the overhead to cache Q, K, and V. Previous MHA accelerators [4, 5] realize this goal by pipelining the attention calculation. However, this requires a large number of MACs. Based on the VDS-based MHA algorithm, Fig. 4-(b) shows the optimized timing diagram of AttBind ASIC. We use two systolic arrays in the reconfigurable MAC unit to compute the K and K branches in parallel. Then the bound $\mathbf{KV}$ is stored in the D-Buffer. The remaining calculations required for Q and MLP layer are sequential without branches. Hence, we configure the two systolic arrays in 24×12 mode to process the same matrix. The optimized timing is helpful to maintain high MAC utilization while reducing redundant data movement.

IV. Evaluation

A. Methodology

Hardware modeling. The proposed hardware architecture is implemented using Verilog HDL and synthesized on TSMC 40nm CMOS library with clock frequency at 1GHz. $N_{\text{PU}} = 1$ is implemented with a total of 256KB on-chip buffers. CACTI [15] is used to estimate the area and power consumption of on-chip buffers. We assume an HBM bandwidth of 128GB/s, and the off-chip energy is extracted from [13].

Workloads. We use the Long Range Arena (LRA) benchmark [12] to evaluate the performance of the MHA algorithms. The tasks span across four workloads and sequence lengths: 1. **Image (1K)** - image classification on CIFAR-10 dataset with 1K sequence length. 2. **ListOps (2K)** - modeling capability test for hierarchically structured data [16] with 2K sequence length. 3. **Retrieval (4K)** - byte-level document retrieval [17] with 4K sequence length. 4. **Text (4K)** - byte-level text classification using IMDb reviews [18] with 4K sequence length.

Baselines. We compare AttBind with six state-of-the-art counterparts: 1. The naive Transformer [1], 2. Reformer [8], 3. Linformer [7], 4. Performer [9], 5. Nyströmformer [10], and 6. Hrrformer [3]. The Transformer structure used follows the setting in [10], where a 2-layer Transformer model with

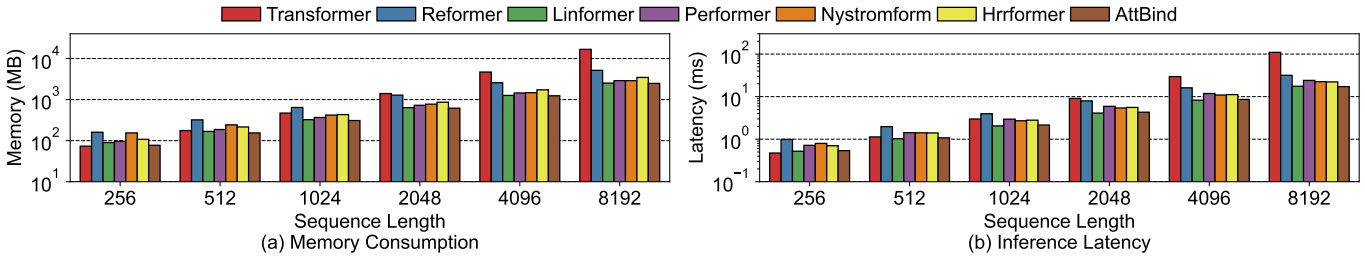


Fig. 5: GPU performance comparison using sequence lengths from 256 to 8K for MHA algorithms (Transformer [1], Reformer [8], Linformer [7], Performer [9], Nyströmformer [10], Hrrformer [3], and AttBind).

TABLE I: Accuracy results for various MHA algorithms on the Long Range Arena (LRA) benchmark [12].

MHA Algorithm	Image (1K)	ListOps (2K)	Retrieval (4K)	Text (4K)	Avg.
Transformer [1]	38.2	37.1	79.4	65.0	54.9
Reformer [8]	43.3	19.1	78.6	64.9	51.5
Linformer [7]	37.8	**37.2**	79.4	55.9	52.6
Performer [9]	37.1	18.8	78.6	63.8	49.6
Nyströmformer [10]	41.6	37.1	**79.6**	**65.5**	55.9
Hrrformer [3]	42.5	37.0	79.2	**65.5**	**56.0**
This work	**42.6**	**37.2**	78.5	65.3	55.9

embedding dimension of 64, hidden dimension of 128, 4 attention heads. Mean pooling is used for all tasks. The number of hashes for Reformer is 2, the projection dimension for Linformer is 256, and the random feature dimension for Performer is 256. AttBind algorithms are implemented using PyTorch on a system with an 8-core Intel i7-11700K CPU and NVIDIA RTX 4090 GPU.

B. Algorithm and Software Evaluation

Comparison to other MHA algorithms. We first evaluate the accuracy of different MHA algorithms in four workloads of the LRA [12] benchmark. Table I compares the accuracy of AttBind to six state-of-the-art MHA algorithms: Transformer [1], Reformer [8], Linformer [7], Performer [9], Nyströmformer [10], and Hrrformer [3]. Our proposed AttBind achieves the best accuracy on **Image (1K)** and **ListOps (2K)** tasks. Meanwhile, AttBind also produces competitive performance with Nyströmformer [10] and Hrrformer [3] on the other two tasks. In general, AttBind achieves the second highest average accuracy 55.9% in all tasks, which is comparable ($< 0.1\%$ gap) to Nyströmformer and Hrrformer. The experiment demonstrates that AttBind can provide satisfactory performance on different types of sequence data and at different sequence lengths.

Inference performance on GPU. As pointed out in Section II, the key challenge of accelerating long-range MHA is prohibitive memory consumption due to the quadratic self-attention layer. The other aspect that we focus on is how AttBind can help improve inference efficiency of the Transformer. We implement the six baselines as well as AttBind on Pytorch. Then, the average GPU memory consumption and latency per instance using 256 to 8K sequence lengths are measured and summarized in Fig. 5. We use different batch sizes (from 4 to 256) depending on the sequence length to saturate the GPU performance. For memory consumption in Fig. 5-(a), AttBind consistently consumes less memory footprint than the other benchmarked models. As a result, AttBind yields $6.7\times$ memory efficiency on 8K length over the naive MHA [1]. Compared to Hrrformer [3], AttBind reduces 15% to 50% memory consumption because AttBind avoids using FFT and IFFT functions for binding and unbinding operations. Instead, only GEMM is required for AttBind, suggesting that our design gains better memory efficiency.

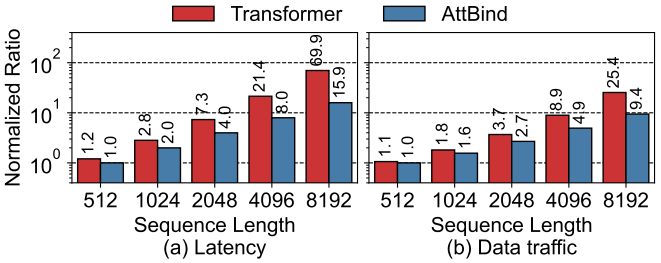


Fig. 6: (a) Inference latency and (b) Off-chip data traffic on ASIC for various input sequence lengths.

The inference speed comparison is given in Fig. 5-(b). The speedup of AttBind over naive Transformer [1] is more significant for long sequences (up to $6.5\times$ at 8K). AttBind achieves comparable latency to Linformer [7] which uses a low-dimensional projection to speed up the MHA module. However, AttBind has 3.3% accuracy improvement compared to Linformer. Compared to the most accurate Nyströmformer and Hrrformer models, AttBind generates $1.1\times$ to $1.3\times$ speedup over Nyströmformer. $> 20\%$ runtime reduction over Hrrformer is observed in different sequence lengths due to AttBind's lower binding and unbinding overhead. In summary, AttBind provides a good balance between memory efficiency and inference speed among all baselines.

C. Hardware Performance Comparison

Effectiveness of AttBind algorithm on ASIC. AttBind is memory-efficient due to its compact data structure after VDS-based binding operation. This is especially effective for an ASIC accelerator that has limited on-chip buffers. We evaluate the benefits of implementing AttBind's VDS-based MHA algorithm on the proposed ASIC design. The latency and data movement estimations are illustrated in Fig. 6, where the naive Transformer [1] is regarded as the baseline. We use the same configurations as the experiment in Fig. 5 while the sequence length ranges from 512 to 8K. The naive

TABLE II: AttBind vs. existing ASICs in 40nm node.

Design	SpAtten [4]	DTQAtten [6]	RAWAtten [5]	AttBind
Clk. Freq. (GHz)	1.0	1.0	1.0	1.0
Area (mm^2)	1.55	1.41	1.64	1.84
Throughput (GOP/s)	360	953	768	489
Energy Efficiency (GOP/J)	382	1298	1170	1178
Area Efficiency (GOP/mm^2)	238	678	469	266
Supported seq. length	$\leq$1K	$\leq$512	–	**$\leq$8K**

Transformer's attention matrix is assumed to offload to off-chip HBM. As the input sequence length scales up, AttBind incurs near-linear inference latency increase (Fig. 6-(a)). It should be noted that though AttBind has $\mathcal{O}(ND\sqrt{D})$ MHA complexity, the resulting computation complexity is still small as compared to the overall complexity. In contrast, Transformer's MHA module requires quadratic attention complexity and contributes to a more significant overhead for the sequence length $>$ 4K. As a result, AttBind achieves $1.1\times$ to $7.8\times$ speedup over naive Transformer.

The reduced memory consumption of AttBind algorithm helps the ASIC accelerator reduce the off-chip data movement, which dissipates expensive energy consumption. As shown in Fig. 6-(b), increasing the sequence length increases the data traffic gap between the naive Transformer and AttBind. AttBind attains up to $4.5\times$ data traffic saving at 8K sequence. This is because AttBind skips the computation for the quadratic attention matrix and the bound K and V matrices can be fully cached on-chip.

Comparison with other ASIC designs. AttBind ASIC accelerator consumes 1.84mm^2 area and 415mW power. We compare it with three existing ASIC accelerators for MHA, including SpAtten [4], DTQAtten [6], and RAWAtten [5]. The performance of AttBind is simulated based on a Transformer model with intermediate dimension D =1K and 4 heads. Table II summarizes the key hardware metrics of the four ASIC accelerators. The throughput of AttBind ASIC is between SpAtten and RAWAtten while the energy efficiency is slightly lower than the most efficient DTQAtten. In particular, AttBind supports $8\times$ to $16\times$ maximum sequence length compared to SpAtten and DTQAtten. The advantages of AttBind come from: 1. VSD-based MHA algorithm reduces off-chip data movement as well as MHA complexity, and 2. The proposed reconfigurable architecture guarantees a high hardware utilization. AttBind is more suitable for tasks that require long-range attention.

V. Conclusion

This work presents software and hardware co-design, AttBind, to accelerate long-range multi-head attention (MHA) in Transformer [1]. By utilizing the VSA binding operation [3, 11], we first propose the memory-efficient VDS-based MHA algorithm with only $\mathcal{O}(ND\sqrt{D})$ complexity. Our algorithm yields competitive accuracy to the state-of-the-art MHA variants on the LRA benchmark [12], but requires less memory consumption and delivers faster inference speed. We also develop a reconfigurable ASIC accelerator to implement the proposed algorithm. The experiments show that the AttBind design scales well with the sequence length, providing $7.8\times$ speedup and $4.5\times$ data movement saving over the naive Transformer. Compared to existing ASICs, AttBind has comparable hardware performance and efficiency while supporting 8-16$\times$ longer sequences.

Acknowledgements

This work was supported in part by the Center for Processing with Intelligent Storage and Memory (PRISM) SRC grant number 2023-JU-3135, CoCoSys, centers in JUMP 2.0, an SRC program sponsored by DARPA, and TILOS AI Research Institute (NSF CCF-2112665).

References

[1] A. Vaswani *et al.*, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[2] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European Conference on Computer Vision (ECCV)*, 2020, pp. 213–229.

[3] M. M. Alam *et al.*, "Recasting self-attention with holographic reduced representations," in *International Conference on Machine Learning*, 2023.

[4] H. Wang, Z. Zhang, and S. Han, "Spatten: Efficient sparse attention architecture with cascade token and head pruning," *ArXiv*, vol. abs/2012.09852, 2020.

[5] W. Li *et al.*, "Rawatten: Reconfigurable accelerator for window attention in hierarchical vision transformers," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2023, pp. 1–6.

[6] T. Yang *et al.*, "Dtqatten: Leveraging dynamic token-based quantization for efficient attention architecture," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2022, pp. 700–705.

[7] S. Wang *et al.*, "Linformer: Self-attention with linear complexity," *arXiv preprint arXiv:2006.04768*, 2020.

[8] N. Kitaev *et al.*, "Reformer: The efficient transformer," in *International Conference on Learning Representations*, 2020.

[9] K. M. Choromanski *et al.*, "Rethinking attention with performers," in *International Conference on Learning Representations*, 2020.

[10] Y. Xiong *et al.*, "Nyströmformer: A nyström-based algorithm for approximating self-attention," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 16, 2021, pp. 14 138–14 148.

[11] J. Gosmann and C. Eliasmith, "Vector-derived transformation binding: An improved binding operation for deep symbol-like processing in neural networks," *Neural computation*, vol. 31, no. 5, pp. 849–869, 2019.

[12] Y. Tay *et al.*, "Long range arena: A benchmark for efficient transformers," in *International Conference on Learning Representations*, 2020.

[13] M. O'Connor, N. Chatterjee, D. Lee, J. Wilson, A. Agrawal, S. W. Keckler, and W. J. Dally, "Fine-grained dram: Energy-efficient dram for extreme bandwidth systems," in *2017 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2017, pp. 41–54.

[14] G. C. Cardarilli *et al.*, "A pseudo-softmax function for hardware-based high speed image classification," *Scientific reports*, vol. 11, no. 1, p. 15307, 2021.

[15] K. Chen *et al.*, "Cacti-3dd: Architecture-level modeling for 3d die-stacked dram main memory," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2012, pp. 33–38.

[16] N. Nangia and S. Bowman, "Listops: A diagnostic dataset for latent tree learning," in *Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, 2018, pp. 92–99.

[17] D. R. Radev *et al.*, "The acl anthology network corpus," *Language Resources and Evaluation*, vol. 47, pp. 919–944, 2013.

[18] A. Maas *et al.*, "Learning word vectors for sentiment analysis," in *Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 2011, pp. 142–150.