HSCONN: Hardware-Software Co-Optimization of Self-Attention Neural Networks for Large Language Models

Siqin Liu
ls847719@ohio.edu
Ohio University
Athens, Ohio, USA

Prakash Chand Kuve
prakashchand.kuve@microchip.com
Microchip Technology Corporation
India

Avinash Karanth
ls847719@ohio.edu
Ohio University
Athens, Ohio, USA

ABSTRACT

Self-attention models excel in natural language processing and computer vision by capturing contextual information but encounter several challenges such as efficient data movement, quadratic computational complexity, and excessive memory accesses. Sparse attention techniques emerge as a solution, however, their irregular or regular patterns, coupled with costly data pre-processing, diminish their hardware efficiency. This paper introduces HSCONN, an energy-efficient hardware accelerator for self-attention, mitigating computational and memory overheads. HSCONN employs dynamic voltage and frequency scaling (DVFS) along with exploiting dynamic sparsity in matrix multiplication, thereby optimizing energy efficiency. The approach includes a row-wise pruning algorithm and independent voltage/frequency islands for processing elements, exploiting additional sparsity to reduce memory access and overall energy consumption. Experiments in natural language processing showcase HSCONN's remarkable speedups (1952×, 615×) and energy reductions (up to 820×, 113×) over CPU and GPU architectures. Compared to A3, SpAtten, and Sanger, HSCONN demonstrates superior speedup (1.71×, 1.25×, 1.47×) and higher energy efficiency (1.5×, 1.7×, 1.4×).

CCS CONCEPTS

• **Computer systems organization → Systolic arrays**; • **Hardware → Hardware accelerators**.

KEYWORDS

Large Language Models, Hardware and Software Codesign, Domain Specific Accelerator

ACM Reference Format:
Siqin Liu, Prakash Chand Kuve, and Avinash Karanth. 2024. HSCONN: Hardware-Software Co-Optimization of Self-Attention Neural Networks for Large Language Models. In *Great Lakes Symposium on VLSI 2024 (GLSVLSI '24), June 12–14, 2024, Clearwater, FL, USA*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3649476.3658709

1 INTRODUCTION

Transformer models have significantly improved the performance of Natural Language Processing (NLP) applications and have shown encouraging results in the field of Computer Vision (CV). The transformer model is distinct in its design as it relies solely on attention mechanisms as its fundamental building blocks, rather than the traditional models that employ recurrence or convolution. This revolutionary design has enabled the transformer and its variations to outperform traditional models in different NLP tasks such as machine translation, text classification, and text generation [3, 13].

Even though transformers have proven to be effective models for various NLP tasks, deploying them on devices with limited hardware resources continues to remain a challenge. This is because attention operations within the transformer model demand higher computation and significant memory accesses. In contrast to convolutional and recurrent neural networks that aggregate information locally, vanilla self-attention models calculate attention for every combination of queries and keys. While this approach provides a larger accessible context, it comes at a significant computational cost, which increases quadratically with the sequence length. For a single input containing 16K tokens, the computation of one self-attention module for the BERT-based model reaches 861.9×10^9 floating point operations per second (FLOPs).

Prior work has proposed to co-design attention algorithms and accelerator architectures to mitigate the attention overhead. For instance, A3 [4] leverages several approximation strategies to avoid computing near-zero scores to reduce the computational overhead. SpAtten [15] proposes a cascaded token pruning mechanism that progressively prunes unimportant tokens to reduce the overall complexity. However, these two solutions still have some drawbacks. Specifically, A3 needs to load all the data on-chip to perform approximate computation which does not reduce the off-chip DRAM accesses. The cascaded token pruning used in SpAtten successfully reduces both computation and DRAM access, but it is a coarse-grained strategy that does not support dynamic pruning for different attention heads.

Dynamic Voltage and Frequency Scaling, which is based on scaling the frequency and voltage, is a well-known energy management technique that trades off the processing speed with energy savings [8, 9]. Prior works have applied DVFS to all levels of the computing system - cores, caches, network, and memory - to maximize energy efficiency. Instead of explicitly addressing each zero-valued operation as in conventional ML accelerators, the DVFS technique inherently eliminates workload imbalance by adjusting the V/F of the processing elements (PEs) on a longer time scale to save energy. In extremely sparse workloads, the PEs can be power-gated i.e.

the power is completely cut off to reduce both dynamic and static power.

In this paper, we propose **HSCONN**, a dynamic-voltage-and-frequency-scaling enabled hardware accelerator that simultaneously exploits sparse scores in attention models and explores the optimal dataflow for reduced data movement. In HSCONN, we dynamically sparsify the matrix multiplication in attention models based on a quantized approximation of the score matrix. The DVFS subsystem uses the score matrix as the workload estimation to adjust the voltage and frequency of the PEs to maximize energy efficiency. To bolster the performance of DVFS, we propose a row-wise pruning algorithm to further sparsify the entire row of the score matrix to reduce memory access and computation. In hardware, we arrange several PEs as an island with independent voltage and frequency domains to exploit this additional row-wise pruning proposed in HSCONN. The major contributions of this work are as follows:

(1) **Apply DVFS to HSCONN with Pruning Algorithm:** We propose a DVFS scheme for HSCONN by applying power-gating to PEs during periods of low computation to save static power and dynamically scaling voltage and frequency (V/F) during periods of medium to high computation to reduce dynamic energy consumption. We further develop a hardware and software co-design by customizing the algorithm with row-wise pruning, which favors the proposed DVFS-based hardware implementation without losing inference accuracy.
(2) **Dataflow Optimization:** We propose a customized dataflow wherein we unify the sparse dense matrix multiplication (SDMM) and sparse matrix multiplication (SpMM) in attention models under different computation stages. The dataflow eliminates sparsity decoding and memory transfer overhead by exploiting row-wise sparsity and maximizing the attention mask for the entire computation chain.
(3) **RTL Evaluation and Comparison:** We evaluate HSCONN on both NLP and ImageNet attention models over real-world datasets. We show that HSCONN achieves multiple orders of magnitude improvement in speedup and energy reduction over commodity CPUs/GPUs. HSCONN also outperforms state-of-the-art attention accelerators such as SpAtten, A3, and Sanger. We also evaluate different DVFS settings and granularities to explore the optimal hardware configuration.

To the best of our knowledge, this is one of the first attempts to implement DVFS in attention-based models by evaluating score sparsity for arbitrary dataflow patterns for both NLP and ImageNet applications to reduce both computational complexity and memory accesses.

2 BACKGROUND

2.1 Attention basics

Transformers have demonstrated leading-edge performance on a variety of NLP tasks [3, 6]. For a block of a transformer model, the input is a sequence of n vectors (tokens). Three linear projection weights project the input to Query, Key, and Value. Attention is then performed on these features to capture long-term dependencies of the input sequence. The h−th head computes the outputs as follows:

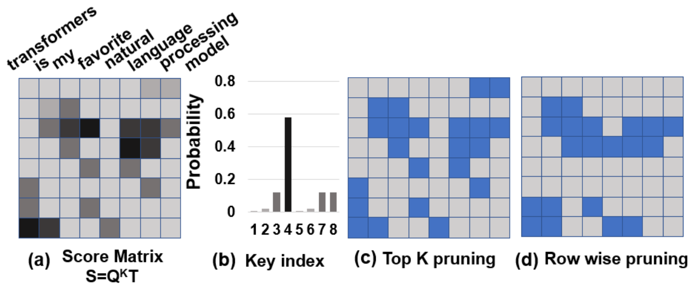

Figure 1: (a) Attention probability matrix. (b)The probability distribution is dominated by the query-key pair (my-favorite); (c) Top-k pruning zeros out elements that are valued below the threshold; (d) Row-wise pruning further trims the score matrix by eliminating entire rows.

$$Q_h, K_h, V_h = X \cdot W_Q,\ X \cdot W_K,\ X \cdot W_V$$
$$\text{Attention}(Q_h, K_h, V_h) = V_h \cdot \text{Softmax}\left(\frac{Q_h K_h^T}{\sqrt{d_h}}\right) \tag{1}$$

2.2 Motivation of DVFS dependent sparsity

Fig.1(a,b) depicts a probability matrix and essential indices derived from training the BERT-large model on the SQuAD-v1 dataset. We observe that a limited set of elements exhibits significant probabilities. Notably, in the annotated row, the distribution is overwhelmingly influenced by the probability of the pair <my, favorite>. By focusing solely on these pivotal query-key pairs during attention calculations and disregarding others, substantial computational resources can be conserved. Additionally, limiting the loading of selected keys and values onto the on-chip memory can effectively minimize total memory access requirements. Therefore, we can adopt the Top-K pruning algorithm [11] (as shown in Fig.1(c) to dynamically determine the sparsity pattern by analyzing the inputs. However, such irregular and dynamic sparsity deters efficient hardware implementations due to low computational intensity and hardware utilization. Fortunately, the DVFS technique effectively improves hardware energy efficiency by scaling down the supply voltage and corresponding frequency for sparse workloads and scaling up the voltage and frequency for dense workloads. This feature inherently addresses the workload imbalance problem without incurring complex reconfigurable hardware and datapath control. In HSCONN , we implement the DVFS scheme by grouping the PE array into PE islands with independent voltage and frequency supplies and feeding each PE island with a chunk of the sparse workloads. As a codesign of hardware and algorithm, HSCONN further improves the pruning algorithm (Fig.1(d) based on Top K pruning (Fig.1(c)) to better accommodate the DVFS-based architecture.

3 PROPOSED ARCHITECTURE

This section introduces the HSCONN architecture to support the proposed DVFS scheme. Our HSCONN mainly targets two design goals: (1) HSCONN supports arbitrary sparsity patterns with low control overhead while achieving high throughput. We achieve this goal by designing low-bit quantization of score estimation and implementing DVFS at PE-level granularity. (2) HSCONN is designed to be both computation and memory-efficient. Since data reuse and reduced memory accesses are essential to the accelerator efficiency, we integrate a customized dataflow that tightly fits within the DVFS design to minimize the data access from the on-chip buffer and maximize the data reuse in the PE array.

3.1 Microarchitecture

The proposed architecture overview of HSCONN is illustrated in Fig.2(a). HSCONN is composed of four functional units: score approximation, DVFS module, systolic PE array, and memory hierarchy. The score approximation (colored in grey) is designed for calculating the approximated score with the truncated bit width of two input operands (querries and keys). DVFS modules (colored in red) detect the runtime sparsity of the data stream and accordingly scale the frequencies and voltages of the PE array. PE array is constructed as a 16×16 systolic architecture and is mainly responsible for the multiplications and accumulations (MACs) of the attention algorithm. The softmax activation function is also implemented inside the PE for post-processing. Detailed design is elaborated in the PE microarchitecture as shown in Fig.2(b). We design a hierarchical memory design to (1) constrain the ASIC footprint with limited on-chip buffer size; and (2) maximize the data reuse in the PE array and the least expensive memory hierarchy (scratchpad) to reduce off-chip memory accesses.

3.2 Dataflow Exploration

The actual non-zero distribution can be arbitrary, leading to irregular data access of SDMM and SpMM operations. Besides, the sparsity occurs in the score matrix, which is both the output of SDMM operation and the input of SpMM operation, making it challenging to decode the sparsity to cooperate with dense queries, keys, and values. Our hardware design uses a unified dataflow, which takes the sparse score mask that indicates the sparsity pattern as input. This dataflow unifies the SDMM and SpMM operations by treating them as a chunk of unbalanced vector-vector multiplication (VVM). We distribute these workloads to the PE islands according to the row index. Then, the DVFS subsystem controls the voltage and frequency (V/F) supplies of each PE island based on the sparse information from the mask.

As shown in Fig.3, the complete dataflow is divided into three stages, where Stage 1 implements SDMM of $I \times W_K$, W_Q, and W_V; Stage 2 implements the SpMM of $K \times Q$; Stage 3 is responsible for the SpMM of $S \times V$. In Stage 1, the query, key, and value matrices are computed by multiplying the input matrix with the pre-trained weights. In this step, each row of the input matrix is horizontally mapped to one PE island, while the column of the weight matrix is vertically delivered. The computation of the first step is completely dense and all PE islands are configured with the highest V/F to achieve the highest throughput. PEs within an island compute different attention heads in parallel. In Stage 2, the key and query matrices generated from the last stage are quantized by 4 bits and are multiplied to obtain the approximate score matrix, which is further used to develop the attention mask by the sparsity detector module. As shown in the middle part of Fig.3, the masked SpMM of key and query matrices are mapped onto the PE islands with only one line VVM is completely avoided. Therefore, the corresponding PE island 2 is power-gated to save power and the remaining islands share the same DVFS setting. In Stage 3, the score matrix is multiplied by the value matrix, of which the SpMM is the most complex compared to the previous two stages. We demonstrate three scenarios in this example: (1) The computation of completely zero-valued rows of S (row 2) is blocked and the corresponding PE islands are power gated as in Stage 2. (2) Row 1 of S remains dense and is mapped to PE island 1, which is supplied with the highest V/F. (3) For the rest rows, depending on the sparsity ratio, the PE islands are scaled accordingly with different V/F. PE islands may work asynchronously with different V/F settings in this stage, which incurs extra synchronization costs. However, the DVFS control logic alleviates this problem to some extent that higher sparse workloads with fewer computations are always configured to lower V/F islands, making limited buffer sizes sufficient to accommodate these asynchronous outputs in PEs. What's more, the final output is sent directly back to the global buffer to avoid synchronization.

3.3 DVFS Design

The sparsity of attention occurs in the intermediate matrix (scores) while the input matrices are still dense (queries, keys, values). The irregular sparsity makes it hard to leverage the parallelism of the systolic array, as systolic arrays exhibit highly structured data access. To effectively exploit the sparsity, we propose a DVFS scheme to exploit the sparsity without specifying fixed sparsity patterns and inherently eliminate workload imbalance.

Each DVFS model consists of one inactive state (power-gated) and four active states. In an inactive state, the voltage supply to the specific PE and its outgoing interconnection is reduced to 0 V with no clock applied to the PE. V/F pair for one PE may switch for each epoch. After testing several epoch sizes, we set 50 cycles for a relatively balanced design point [5, 18]. PE in an active state can operate in any one of the four available voltage levels. These voltage and frequency pairs are commonly configured in DVFS-supported processors or accelerators [16]. Due to the highly sparse nature of attention, we introduce a baseline mode that is maintained constantly at a high voltage level and update the threshold values to fit in the attention workload distribution as shown in Table 1.

As shown in Table 1, the power-gating model maximizes static power reduction by assigning more PEs to the power-gating mode. The power-saving model maximizes dynamic power reduction and assigns the highest portion of workloads (10-70%) to the lowest V/F mode (0.8V/2.75ns). The booster model, on the contrary, operates the PEs at the highest voltage whenever the workload percentage is higher than 40% to enable the highest computation performance. The Vanilla model is based on a moderate strategy that evenly allocates the workloads to all V/F modes.

3.4 Row-wise pruning algorithm

Our HSCONN 's row-wise pruning algorithm aims to not only alleviate the costly quadratic computational complexity of the number of input tokens in self-attention blocks but also bolster the hardware efficiency of computing the irregular sparse workloads. Although our proposed DVFS-based hardware can flexibly process

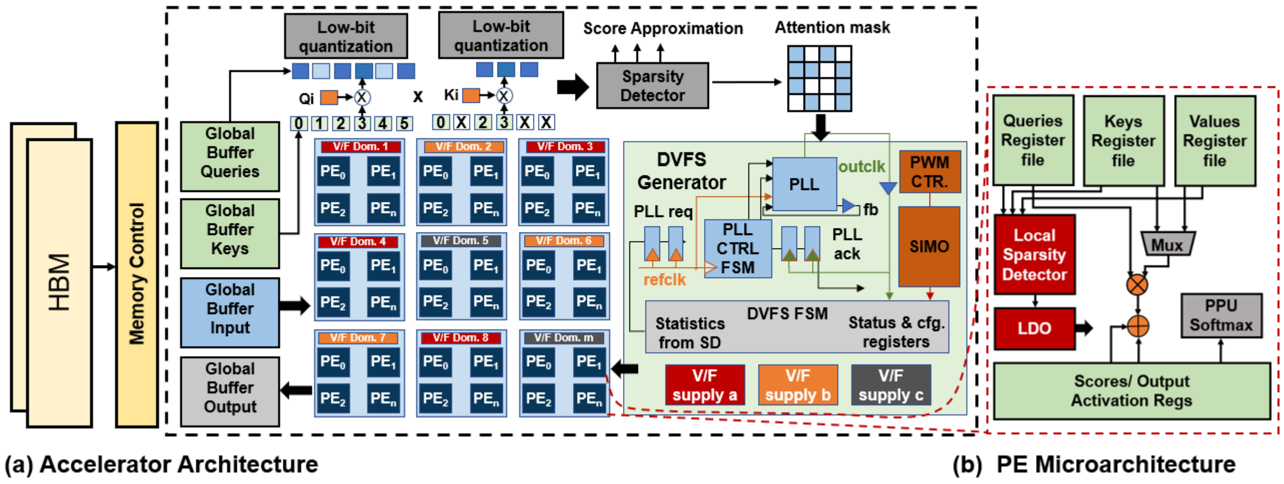

Figure 2: (a) An overview of HSCONNarchitecture consisting of off-chip HBM, low-bit score approximation, systolic PE array, and DVFS controller and generator. (b) Microarchitecture of the PE.

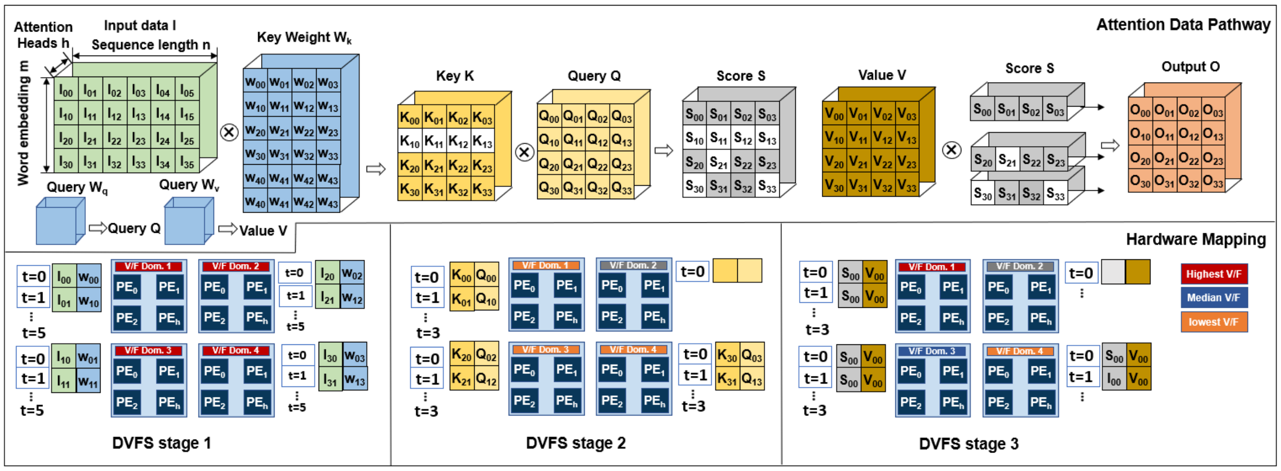

Figure 3: A walkthrough example shows the workflow of HSCONN divided into three major steps, where the first step implements SDMM of $I \times W_K$, W_Q, and W_V; stage 2 implements the SpMM of $K \times Q$; the third step is responsible for the SpMM of $S \times V$. The exponential operation and softmax activation are implemented in place for each PE. DVFS is guided by the approximated score attention mask and scales the voltage and frequency for each PE in stage 2, and scaled accordingly in other stages.

Table 1: Workload Distribution Among Different Models and DVFS modes.

DVFS Models	power-gating	0.8V/ 2.75ns	1.0V/ 2.25ns	1.2V/ 1.8ns
Baseline	0%	/	/	>0%
Power-Saving	<10%	10-85%	85-90%	>90%
Vanilla	<10%	10-30%	30-55%	>55%
Booster	<10%	10-45%	45-50%	>50%

the imbalanced workloads by scaling the voltage and frequency, fixed patterning of the sparsity still contributes to reduced runtime controlling overheads. To generate the desired sparse mask patterns, we first extract the attention mask by forwarding the pre-trained models on all training samples, and then perform top-k pruning according to a retrainable pruning threshold θ. Then, we select only the score matrix of high value by pruning the remaining entire rows of the matrix if the total number of non-zero-valued elements is less than the threshold α. Such pruning will generate a binary mask (as shown in Fig.1(d)), where a number of sparse rows become completely zero and the remaining rows become moderately denser after retraining to maintain accuracy.

The runtime DVFS controlling with row-wise pruning preprocessing is described in Algorithm 1. For a given averaged and normalized attention map A extracted from a pre-trained attention

Algorithm 1 Implementation of DVFS with row-wise pruning.

```
Input:
    Key, Query, Value tensors $K, Q, V \in \mathbb{R}^{nm}$
    pruning threshold $\theta$, row-wise augmentation threshold $\alpha$
    Current DVFS island index $h$
Output:
    Voltage and Frequency state for each PE island.
 1: Initialize start voltage 1.2V and frequency state
    Quantized approximation of the score matrix $S'$
 2: $S'$ = softmax(SDMM($quant4bit(K), quant4bit(Q)$))
    Attention mask $AM \in \mathbb{R}^{nm}$ with thresholding
 3: $AM_{ij} = 0\ if\ S'_{ij} < \theta\ else\ 1$
    Row-wise pruning augmentation
 4: $AM_i = 0\ if\ \sum_j AM_{ij} < \alpha\ else\ unchanged$
    Attention computation with DVFS control
 5: for $islands\_index\ h$ do
 6:     $V_h = 1.2V$ for all islands
        set up highest voltage mode in stage 1
 7:     $K, Q, V = SDMM(I, (W_K, W_Q, W_S)$
 8:     if $AM_{ij} = 0$ for all j then
 9:         power-gate current PE island
10:     else
11:         $V_h = 1.0V$
12:     end if
13:     $S = softmax(SpMM(K, Q))$ using mask $AM$
        set up moderate voltage mode in stage 2
14:     $O = SpMM(S, V))$ using mask $AM$
15:     if $AM_{ij} = 0$ for all j then
16:         power-gate current PE island
17:     end if
18: end for
```

model on all training samples, we prune and reorder it into row-dense, row-sparse, or row-empty patterns. In the inference phase, the sparse mask is generated at runtime by approximating the score matrix values and used by the DVFS subsystem to allocate workloads, power gate idle islands, and scale the voltage and frequency. Specifically, we first generate the attention mask $AM \in \mathbb{R}^{nm}$ based on the quantized matrix multiplication of K and Q (Line 1-3), where the pruning threshold is obtained through pre-training and remains constant in this phase (Line 3). After the irregular sparse pattern has been identified, we regularize it by zeroing out the entire row if the total number of non-zero valued elements in the approximated score matrix is less than the pre-trained row-wise pruning threshold α (Line 4). Guided by the augmented attention mask, the DVFS subsystem scales the voltage and frequency of each PE island, breaking it down into three stages (Lines 5-18).

4 SIMULATION AND PERFORMANCE

4.1 Simulation Setup

Benchmarks: We evaluate our method on BERT [3], GPT-3 [2], and large language model LLamA [12]. For NLP models, we select tasks Stanford Question Answering Dataset SQuAD [10], GLUE [14], MNLI [17], and commonsense reasoning benchmarks PISCO [1], BoolQ, and SIQA. All the models are trained on the publicly released pre-train weights with default training parameters. We then modify the code to support the proposed dynamic sparse pattern.

Platforms for comparison: We compare our framework with modern hardware accelerators, including cloud GPU (NVIDIA Tesla V100), and commodity CPU (Intel Xeon I7 4770). We measure the performance of GPUs using PyTorch with cuBLAS, and CPUs using PyTorch MKL. To ensure a fair comparison, we evaluated our

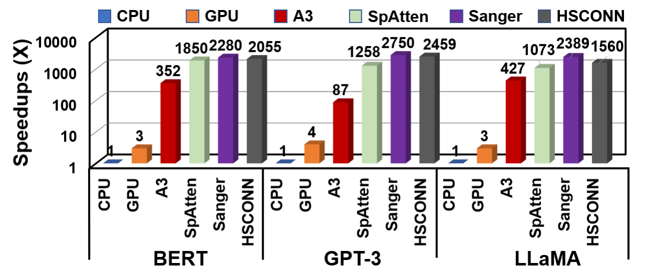

Figure 4: The normalized inference speedups (w.r.t. CPU) achieved by our HSCONN framework over three SOTA transformed accelerators.

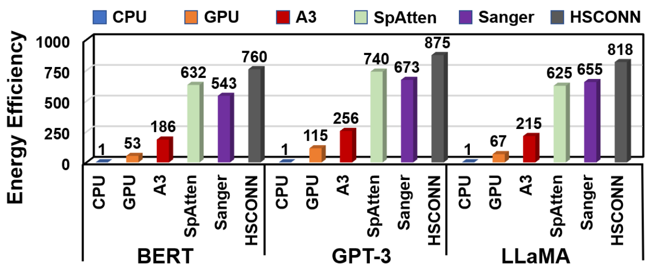

Figure 5: The normalized energy efficiency improvement (w.r.t. CPU) achieved by our HSCONN framework over three SOTA transformed accelerators.

approach against the state-of-the-art sparse attention accelerators, such as A3 [4], SpAtten [15], and Sanger [7], by scaling their number of multipliers to a 64x64 processing element array at a 1 GHz frequency. We considered both pruning techniques and architectural design in our comparison.

4.2 Simulation Results

Comparison with CPUs and GPUs. Fig.4 shows the speedup of HSCONN over Intel Xeon CPU and V100 GPU on 5 language processing tasks, e.g. MNLI, QNLI, RTE, SQuAD, CLOTH. For BERT, GPT-3, and LLamA models, on average, HSCONN achieves 685×, 614.75×, 520× speedup over GPU, and 2055×, 2459×, 1560× speedup over Intel Xeon CPU. For energy efficiency shown in Fig.5, HSCONN is 11.3×, and 113× better compared to GPU and Xeon CPU. The higher speedup comes from the reduced computation, memory accesses, and the optimized DVFS of HSCONN that can effectively leverage the sparsity with arbitrary patterns. HSCONN processes 64 queries and 64 keys in parallel, which implies that the computing resources are fully utilized in these tasks.

Comparison with Other Accelerators: We evaluate the sparsity design of HSCONN with four state-of-the-art sparse attention accelerators as configured in Table 2. The simulation results of the performance comparisons are included in Fig.4 and Fig.5. HSCONN shows the highest computation saving at the sparsity level thanks to our fine-grained DVFS technique and the corresponding architecture. In comparison, A3 introduces a pre-processing step and uses a rough sparsity prediction technique which hurts model accuracy under aggressive pruning. SpAtten uses a coarse-grained approach

Table 2: Comparison with sparse attention accelerators.

Acc.	A3	SpAtten	Sanger	HSCONN
Sparsity Design	candidate approxmation	Top-K pruning	block pruning	row wise pruning
Comp. Engine	16x16 dot	16x32 matrices mul.	64x16 systolic array	64x64 DVFS array
Tech.	40nm	55nm	55nm	40nm
Memory	64 KB	2 MB	512 KB	512 KB
Area	2.08mm2	1.55mm2	16.9 mm2	12.53 mm2
Power	0.115W	3.82W	2.76W	1.08W
Throughput	221 GOP/s	360 GOP/s	529 GOP/S	613 GOP/S

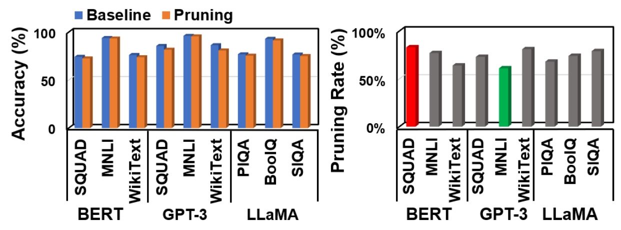

Figure 6: Accuracy before and after pruning-aware fine-tuning (left). Runtime pruning rate with top-k pruning and row-wise enhancement (right).

for attention pruning where it prunes entire columns and rows progressively. Such structural constraint limits the level of sparsity it can exploit in a similar way to traditional weight pruning.

As shown is Fig.4, HSCONN outperforms A3, SpAtten, and Sanger with 1.71×, 1.25, and 1.47× speedup respectively. SpAtten and Sanger apply fixed computing modules, limiting the pattern of sparsity strictly. While HSCONN is composed of a DVFS-based systolic array with more flexibility in supporting sparsity, allowing higher effective throughput. The SpAtten attention model employs a cascaded token-pruning technique to eliminate tokens that are considered unimportant, based on accumulated attention probabilities among layers. However, this method of pruning can lead to significant accuracy loss without retraining, due to its coarse-grained approach.

Accuracy and Pruning Ratio Exploration: We first evaluate the performance (accuracy) of the proposed row-wise turning algorithm by exploring different parameters of the pruning ratio up to 16×. We estimate the pruning ratio and accuracy of each configuration. Since this exploration only involves inference on test sets, it takes several minutes to hours to finish. We plot all the exploration results in Fig.6. The optimal configuration should have both a high pruning ratio and accuracy. We chose the configuration with the highest pruning ratio and negligible accuracy loss (within 0.5%) as our best configuration for each task.

5 CONCLUSIONS

In this paper, we propose an energy-efficient and high throughput accelerator HSCONN, that leverages the DVFS-based PE islands hardware design to support dynamic sparsity and improve energy efficiency for the attention mechanism. We further sparsify the attention by the proposed row-wise pruning algorithm and then the dynamic voltage and frequency scaling subsystem use the sparse mask to save power consumption. Extensive experiments on both NLP and CV benchmarks demonstrate that HSCONN achieves distinct speedups and energy reduction over state-of-the-art attention accelerators.

ACKNOWLEDGMENTS

This research was partially supported by NSF grants CCF-1703013, CCF-1901192, CCF-1936794, CCF-2324645, and CCF-2311544. We sincerely thank the anonymous reviewers for their excellent feedback.

REFERENCES

[1] Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 7432–7439.

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[4] 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. 2020. A^3: Accelerating attention mechanisms in neural networks with approximation. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 328–341.

[5] Weixiong Jiang, Heng Yu, Jiale Zhang, Jiaxuan Wu, Shaobo Luo, and Yajun Ha. 2020. Optimizing energy efficiency of CNN-based object detection with dynamic voltage and frequency scaling. *Journal of Semiconductors* 41, 2 (2020), 022406.

[6] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942* (2019).

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

[8] Seyed Morteza Nabavinejad, Hassan Hafez-Kolahi, and Sherief Reda. 2019. Coordinated DVFS and Precision Control for Deep Neural Networks. *IEEE Computer Architecture Letters* 18, 2 (2019), 136–140.

[9] Seyed Morteza Nabavinejad, Sherief Reda, and Masoumeh Ebrahimi. 2022. Coordinated Batching and DVFS for DNN Inference on GPU Accelerators. *IEEE Transactions on Parallel and Distributed Systems* 33, 10 (2022), 2496–2508. https://doi.org/10.1109/TPDS.2022.3144614

[10] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250* (2016).

[11] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2021. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics* 9 (2021), 53–68.

[12] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[14] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461* (2018).

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

[16] Qiang Wang and Xiaowen Chu. 2020. GPGPU performance estimation with core and memory frequency scaling. *IEEE Transactions on Parallel and Distributed Systems* 31, 12 (2020), 2865–2881.

[17] Qizhe Xie, Guokun Lai, Zihang Dai, and Eduard Hovy. 2017. Large-scale cloze test dataset created by teachers. *arXiv preprint arXiv:1711.03225* (2017).

[18] Zheqi Yu, Pedro Machado, Adnan Zahid, Amir M Abdulghani, Kia Dashtipour, Hadi Heidari, Muhammad A Imran, and Qammer H Abbasi. 2020. Energy and performance trade-off optimization in heterogeneous computing via reinforcement learning. *Electronics* 9, 11 (2020), 1812.