A Low Power Attention and Softmax Accelerator for Large Language Models Inference

Jeong-Hyun Kim[1], Chan-Hoon Kim[2], Soo-Min Rho[3], Ki-Seok Chung[4†]
Department of Electronic Engineering, Hanyang University
Seoul, Korea
{[1]hanagod2015, [2]kch1103, [3]smrho, [4]kchung}@hanyang.ac.kr
[†]Corresponding author.

***Abstract*—Transformer-based models, essential for high-performing Large Language Models (LLMs), surpass traditional Deep Neural Networks but require substantial computational resources. Therefore, more efficient transformer algorithms and accelerators are required to reduce the computational cost and power consumption of LLMs. We observed that as the sequence length increases, softmax operations, which are the key operation of the transformer self-attention mechanism, become the major bottleneck. In this paper, we propose Cross-Road Softmax, an optimized algorithm designed for the softmax operation within the attention layer, specifically tailored for inference in LLMs. Our software experiment was conducted on 8 Natural Language Processing benchmarks for evaluation. Furthermore, we design a Cross-Road Accel using the proposed Cross-Road Softmax that accelerates softmax function of the self-attention layer. We implement Cross-Road Accel in RTL and synthesize it with Synopsys Design Compiler using Nangate 15nm open cell library to obtain power and area statistics. In summary, on average, Cross-Road Accel achieves an approximately 3.5× increase in energy efficiency compared to state-of-the-art transformer accelerators.**

***Index Terms*—AI accelerator, Low Power Design, Algorithm-Hardware Co-Design, Transformer, Softmax, LLMs, NLP**

I. Introduction

Transformer-based Large Language Models (LLMs) [1] have shown breakthrough performance in various fields of Deep Neural Networks. One of the keys to the success is the self-attention mechanism. We observed that the computational cost and power consumption of the self-attention mechanism vary with sequence length. For lengths under 1K, linear operations dominate in the self-attention mechanism. However, with recent models supporting lengths over 1K, non-linear operations have become the primary bottleneck. We observe that as sequence length increases, the softmax computation becomes a critical bottleneck. To address this, we propose Cross-Road Softmax, which computes the softmax function using selective application of *one-hot encoding* and a *base-2* softmax method. Our One-Hot Softmax design leverages one-hot encoding to skip the softmax computation, utilizing softmax's tendency to assign dominant probabilities to outliers within a vector. This allows skipping the softmax operation altogether with an appropriate threshold, identified through experiments with well-known LLMs and tasks [2], [3]. For vectors that cannot skip the softmax operation, we introduce the Base-2 Softmax, which transforms the base to 2 for efficient computation. The Cross-Road Softmax combines both One-Hot Softmax and Base-2 Softmax methods. Moreover, we designed Cross-Road Accel, a hardware accelerator employing the Cross-Road Softmax method, to confirm the advantages of our proposed approach. We conducted software and hardware experiments to evaluate the methods and designs.
In summary, the contributions are listed as follows:

- We propose Cross-Road Softmax, a novel method for LLM inference that operates in dual modes: One-Hot Softmax and Base-2 Softmax, tailored to the characteristics of LLMs and softmax. Refer to Section III-A.
- We profiled well-known benchmarks to characterize softmax in LLMs and determine the optimal threshold for our method. Section III-B will address the details.
- We propose a hardware-friendly attention accelerator, Cross-Road Accel that includes Cross-Road Softmax. Compared with the SOTA accelerators, Cross-Road Accel achieves an average improvement of 3.5× in energy efficiency. Sections IV and V-C will address the details.

II. Background and Motivation

A. *Transformer and Attention Mechanism*

Transformer models mainly consist of stacked encoder or decoder blocks [1]. Each encoder block has two stages: the Multi-head Self-attention (MHA) layer and the Position-wise Feed-forward Network (FFN) layer. The MHA layer executes attention mechanism in parallel across multiple heads, each consisting of single-head attention blocks. These blocks generate Query (Q), Key (K), and Value (V) matrices by multiplying input tokens with pre-trained weights. The Scaled Dot-product Attention mechanism then calculates attention scores by multiplying Q with the transposed K, which are then normalized and converted into probabilities using the softmax function. Softmax is essential for transforming input values into probabilities.

$$softmax(x_i) = \left(\frac{e^{x_i}}{\sum_{i=0}^{k-1} e^{x_i}} \right) \tag{1}$$

As shown in Equation 1, the softmax function normalizes the exponentials of inputs to compute probabilities. It is essential in the attention mechanism, converting the correlation between Q and K into a probability matrix, known as the

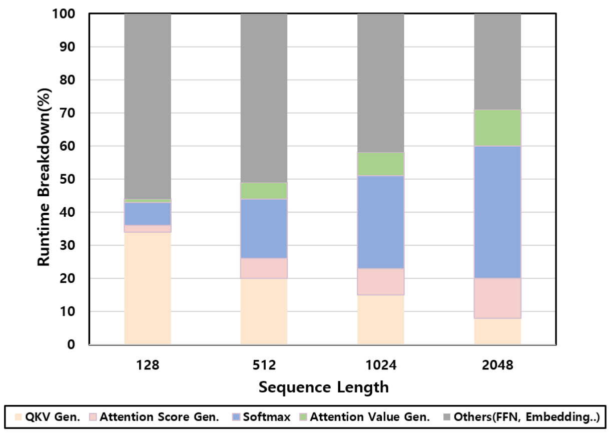

Fig. 1. Runtime breakdown of Transformer operations according to sequence length.

attention distribution. The final result of attention is obtained by multiplying the distribution with the V matrix, known as the context. Finally, all heads are concatenated to match the input dimension of the MHA layer.

B. Motivations

Runtime breakdown of Transformer layers: In the earlier LLMs, the token lengths were less than 1K in most cases. When the token length is less than 1K, the computational load for QKV Generation and FFN operations except for attention is dominant in most LLMs. However, the number of parameters and the max sequence lengths that LLMs should be able to support increase rapidly. As the max sequence length that can be handled by an LLM far exceeds 1K, the load of attention has grown dramatically. Therefore, it becomes very crucial to accelerate the attention computation. To understand the computational bottleneck among all components with the attention layer, performance profiling was carried out. Our profiling results confirmed that as the sequence length increases, the computational load of the softmax function within the attention layer dramatically increases. Figure 1 depicts that the computational load of softmax in LLMs is increasing. The x-axis represents sequence length, while the y-axis denotes the runtime breakdown.

Limitations of conventional hardware: Therefore, many studies have attempted to implement efficient and fast hardware to compute the softmax function. However, since softmax has non-linear features that divide the power of an irrational number e, it is challenging to implement it in hardware. As a result, many existing hardware architectures utilize LUT or CORDIC to implement the exponential function. However, the LUT method requires substantial memory for high precision, which consumes more power than logic circuit implementation. The CORDIC method is quite accurate when the iteration count is high. On the other hand, the latency increases sharply as the iteration count gets higher. In this paper, we propose a novel method of computing the softmax function.

III. Cross-Road Accel

A. Cross-Road Softmax

We introduce Cross-Road Softmax, a novel method of computing softmax. In LLMs inference, we only need the relative probability from the softmax output, so exact probability is unnecessary if one value is significantly larger than the others. In the proposed Cross-Road Softmax, the softmax results are generated from either of the following two modules.

The **One-hot Softmax** module outputs a vector with a single 1 indicating the highest value, simplifying softmax computation using one-hot encoding. This method is used when there's a significant gap between the highest and second-highest values, determined by a pre-defined threshold. Applying one-hot encoding improves performance and reduces power consumption by eliminating the need for full softmax computation. Therefore, setting an appropriate threshold is crucial for optimal performance and efficiency. On the other hand, if the difference between the highest and second values does not surpass the threshold, One-Hot Softmax will not be operated. Instead of One-Hot Softmax, the **Base-2 Softmax** module will be run at that point. The Base-2 Softmax module is designed based on a *hardware-friendly* approach by transforming the nonlinear function e^x in the softmax. The entire process of the Cross-Road Softmax is outlined in Algorithm 1, and the architecture is depicted in Figure 3.

Algorithm 1: Cross-Road Softmax

```
1  Input: $x$ $(Input\ vector,\ INT16)$
2  Output: $dist$ $(Distribution)$
3  Parameter: $k$ $(\#\ of\ elements\ in\ the\ vector\ x)$,
4             $th$ $(Threshold\ value)$
5  $1^{st}, 2^{nd} \leftarrow Detect(x)$
6  if $1^{st} - 2^{nd} \geq th$ then
7      // One-Hot Softmax operation
8      $dist[index(1^{st})] \leftarrow 1.0$
9      $dist[index(else)] \leftarrow 0.0$
10 else
11     // Base-2 Softmax operation
12     $dist \leftarrow \left(\frac{2^{\mathrm{round}(x_i \cdot 1.4375)}}{\sum_{k=0}^{i-1} 2^{\mathrm{round}(x_i \cdot 1.4375)}}\right)$
13 end
```

B. Determining the Threshold

As detailed in Section III-A and lines 7-9 of Algorithm 1, One-Hot Softmax uses a threshold to skip computations. The threshold value is crucial, relying on a dominant probability in the input vector. Even a small difference between the highest and second-highest values impacts the probability distribution. We experimented with BERT-Tiny and 10% samples from GLUE benchmarks [2] to determine the ratio of vectors exceeding the threshold and skipping softmax for power efficiency. We compared naive softmax and tested thresholds from 1 to 4 to find the optimal value.

TABLE I
ACCURACY FOR VARIOUS THRESHOLDS ON BERT-TINY USING 10% OF THE GLUE BENCHMARK SAMPLES.

Threshold	Average Ratio over Threshold (%)	Average Accuracy
$th = 1$	43.87	**0.624**
$th = 2$	32.27	0.646
$th = 3$	15.73	0.645
$th = 4$	5.49	0.644
baseline	0	0.641

The range of the experimental group is justified because the threshold value as 0 makes the algorithm meaningless. Moreover, the value of 5 leads to a near-zero threshold exceeding the ratio, substantially reducing the softmax's acceleration. Table I represents the relation between the ratio and the accuracy drop according to threshold value in LLMs. In Table I, setting the threshold value to 1 results in a significant accuracy drop due to excessive skipping of necessary softmax computations. However, with a threshold value of 2, approximately 32.27% of the vectors can skip softmax computation without significant accuracy loss. As a result, 2 is the best threshold in the case of Table I. In general cases, it is also necessary to apply heuristic methods to determine the optimal threshold value.

IV. HARDWARE IMPLEMENTATION

A. Overall Architecture

Figure 2 illustrates the overall system architecture and Cross-Road Accel. As depicted in Figure 2, we implement a hardware system named **Core Wrapper**, composed of Direct Memory Access (DMA), SRAM, Core Controller, Scheduler, and Cross-Road Accel. Within the Core Wrapper, **Core Controller** is responsible for managing and controlling the system state and **Scheduler** facilitates read/write data from/to the SRAMs, which have a capacity of 16KB. The **Cross-Road Accel** receives Q, K, V and valid signals from the Scheduler as inputs and accelerates the Scaled Dot-Product Attention operation. In Cross-Road Accel, two **8x8 Systolic Arrays** that utilize the output stationary are implemented. The systolic arrays are pipelined to improve the computation throughput. Additionally, the **Dispense Controller Module** pre-processes the output of Cross-Road Softmax, while the **Post Processing Module** post-processes the final output, the context.

B. Cross-Road Softmax Implementation

The **Cross-Road Softmax** module consists of four sub-blocks: Detect Score, Row-Wise Top-2, One-Hot Softmax, and Base-2 Softmax. The **Detect Score** block and **Row-Wise Top-2** block are responsible for preprocessing in the Cross-Road Softmax module. **One-Hot Softmax** activates upon receiving the flag from Row-Wise Top-2. It assigns 1.0 (16'hffff) as the attention distribution (AD) to the index stored in the Idx Pointer, representing the largest value, while all other outputs are set to 0 (16'h0000). This operation assigns maximum probability to the largest input's index and zero to others, completing in one cycle. On the other hand, the **Base-2 Softmax** block operates by storing the results of shift&add and bit slice operations in a buffer for the divide operation. These operations are performed in parallel, maintaining a difference of an 8x8 size from the PE array. When the Base-2 Softmax flag is generated, a division operation is performed, requiring cycles equal to the maximum sequence length (e.g., 128 cycles for a sequence length of 128). However, if the One-Hot Softmax flag is on, the Base-2 Softmax skips the division and flushes the buffer to save energy.

TABLE II
ACCURACY COMPARISON BETWEEN THE SW IMPLEMENTATION AND OUR HW IMPLEMENTATION

GLUE Task	BERT-TINY (Base)	BERT-TINY (Ours)	BERT-LARGE (Base)	BERT-LARGE (Ours)
COLA	0	0	0.62	0.608
MNLI	0.65	0.659	0.861	0.862
MNLI-MM	0.66	0.663	0.863	0.863
MRPC	0.665	0.683	0.857	0.858
QNLI	0.773	0.77	0.926	0.926
RTE	0.588	0.592	0.693	0.7
SST-2	0.809	0.799	0.93	0.93
STS-B	0.533	0.529	0.888	0.887
WNLI	0.436	0.478	0.45	0.42

V. EVALUATION

A. Experimental Setup

Two types of evaluation were conducted: (1) **accuracy degradation** of the hardware implementation compared to software and (2) **hardware performance** in terms of latency, power consumption, and area. To evaluate accuracy degradation, we implemented two well-known LLM models, BERT-Tiny with 2 layers and BERT-Large [3] with 24 layers. The implementation of Cross-Road Softmax uses PyTorch, Hugging Face, CUDA, and an NVIDIA GeForce RTX 3090 GPU. Accuracy measurements were also performed using PyTorch. Cross-Road Accel was implemented using Verilog HDL, with functional verification conducted using Xilinx's Vivado XSIM and APIs. Similar to previous research in transformers, our design predominantly involves multiply-accumulate (MAC) operations. Therefore, we calculated the throughput, commonly named effective throughput, by measuring the number of cycles for MAC operations [4].

B. Software Evaluation

Experiments using eight tasks from the GLUE Benchmark [2] show that the hardware implementation of Cross-Road Softmax results in negligible accuracy drops or slight improvements, as shown in Table II. For the MNLI task, the largest in GLUE, there is no accuracy drop or up to a 3% improvement in the BERT-TINY and BERT-LARGE [3] models.

C. Hardware Evaluation

The hardware implementation was synthesized using Synopsys Design Compiler with the Compile-Ultra option and the NanGate 15nm cell library. Cross-Road Accel was evaluated on throughput, power consumption, and circuit area. Throughput, defined as total MAC operations divided by

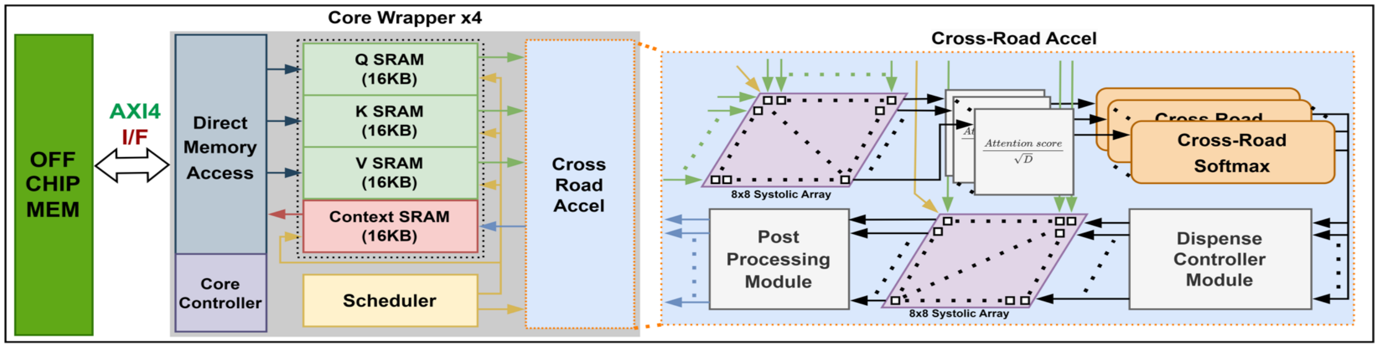

Fig. 2. Architecture of the overall system and Cross-Road Accel.

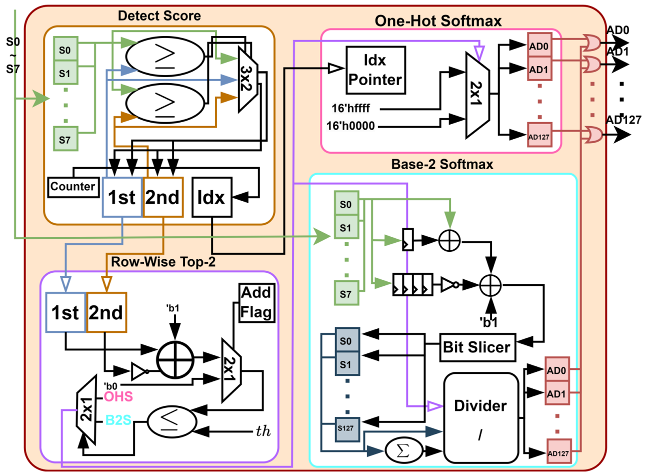

Fig. 3. Architecture of the Cross-Road Softmax module.

processing time [4], is approximately 1.29 TOPS at 2GHz. Synopsys Primetime verified no timing violations at this frequency. Cross-Road Accel consumes 283.47mW of power, and it achieves a throughput of 1.29TOPS, and it occupies an area of 37728um^2. Cross-Road Accel achieved an average energy efficiency of approximately 3.5× better than that of well-known previous works [5]–[7]. Evaluation results are summarized in Table III.

TABLE III
COMPARISON OF PRIOR TRANSFORMER ACCELERATORS

	SpAtten [5]	ELSA [6]	FACT [7]	Ours
Library[nm]	40	40	28	**15**
Frequency[GHz]	1	1	0.5	**2**
MAC Units	1024	528	-	**512**
Power[mW]	2600	969	333.07	**283.47**
Throughput[TOPs]	1.61	1.09	0.928	**1.29**
Energy Efficiency[TOPs/W]	0.62	1.12	4.38	**4.55**

VI. CONCLUSION

In this paper, we propose Cross-Road Accel, which significantly reduced the computational cost of the softmax operation in self attention mechanism. The key to reducing computational cost is to selectively skip the softmax computation under certain conditions. For necessary softmax computations, we propose a more efficient hardware design. We experimented with Cross-Road Softmax in LLMs, but our method can be applied to any models and tasks utilizing softmax. In conclusion, the proposed hardware design, Cross-Road Accel achieves 3.5× better energy efficiency on average compared to existing SOTA transformer accelerators.

ACKNOWLEDGMENT

This work was supported by Institute of Information communications Technology Planning Evaluation (IITP) grant funded by the Korea government(MSIT) (No. 2022-0-00153, Development of O-RU for in-building based on AI network management using beamforming path)

REFERENCES

[1] Ashish Vaswani et al. “Attention is All you Need,” In Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pages 5998–6008.
[2] Alex Wang et al. “GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding,” In Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, pages 353–355, November 2018. Association for Computational Linguistics. URL https://aclanthology.org/W18-5446, doi: 10.18653/v1/W18-5446.
[3] Jacob Devlin, Ming-wei Chang, Kenton Lee, and Kristina Toutanova “BERT:Pre-training of Deep Bidirectional Transformers for Language Understanding,” *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, vol. 1. pp.4171–4186, 2019.
[4] Chang Gao, Daniel Neil, Enea Ceolini, Shih-Chii Liu, and Tobi Delbruck. “DeltaRNN: A Power-efficient Recurrent Neural Network Accelerator,” In Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA ’18), pages 21–30, 2018. Association for Computing Machinery. ISBN 9781450356145. doi: 10.1145/3174243.3174261.
[5] 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, March 2021. doi: 10.1109/HPCA51647.2021.00018.
[6] Tae Jun Ham et al. “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, 2021. doi: 10.1109/ISCA52012.2021.00060.
[7] Yubin Qin et al. “FACT: FFN-Attention Co-optimized Transformer Architecture with Eager Correlation Prediction.” In Proceedings of the 50th Annual International Symposium on Computer Architecture (ISCA ’23), Orlando, FL, USA, 2023. Association for Computing Machinery. ISBN 9798400700958. doi: 10.1145/3579371.3589057.