# SegTransformer: Enhancing Softmax Performance through Segmentation with a ReRAM-Based PIM Accelerator

Yu Chen Wang[1], Ing-Chao Lin[1], and Yuan-Hao Chang[2]
[1]National Cheng Kung University, [2]Academia Sinica
{p76111602, iclin}@gs.ncku.edu.tw, johnson@iis.sinica.edu.tw

***Abstract*—ReRAM-based Processor-In-Memory (PIM) architectures have demonstrated their ability to accelerate the matrix multiplication performed by the Transformer. However, these approaches often shift the performance bottleneck from the attention mechanism to the softmax computation. Moreover, data sharding, commonly employed for acceleration, hinters the Transformer's ability to find global softmax value. This paper proposes SegTransformer, a ReRAM-based PIM accelerator that improves softmax performance with high accuracy through segmentation techniques. Experimental results indicate that the SegTransformer significantly outperforms state-of-the-art Transformer accelerators.**

## I. Introduction

Previous studies on ReRAM-based PIM accelerators have demonstrated superior inference performance and energy efficiency compared to CMOS-based accelerators [1], [2]. Although specialized accelerators have been designed to enhance the self-attention mechanism, they still dominate the execution time in Transformers. Key challenges include: 1) Softmax computation still must wait for the entire input matrix to be ready, which reduces performance. 2) Data sharding, as used in TransPIM [3], reduces data movement but introduces accuracy loss by limiting attention computation to individual memory units, ignoring interactions between units. 3) The LUT-based softmax computation, as used in STAR [4], offers a single-cycle computation with high accuracy but requires a larger LUT to handle fractional values.

This paper proposes SegTransformer to address these issues. It introduces a segmented transmission module (STM) to divide data into segments, enabling pipelined softmax operations. An integrated softmax processing unit (ISPU) maps local to global softmax values using factors from the matching vector unit (MVU), ensuring stable accuracy despite data sharding. The MVU also includes a Rounder module to reduce unnecessary computations by rounding values to a specified precision.

## II. SegTransformer

### A. Overall Structure

Fig. 1 shows the proposed architecture, comprising three hardware modules: 1) STM, which includes circular buffers, 4 ReRAM crossbars, and a demux. 2) Integrated softmax processing unit (ISPU), which consists of a match vector unit (MVU), a softmax recalibrate unit (SRU), two buffers, two content-addressable memories (CAM) for search and subtraction, and a Look-up Table (LUT) for exponential operations. 3) Matrix multiplication module with buffers and a ReRAM crossbar for matrix multiplication.

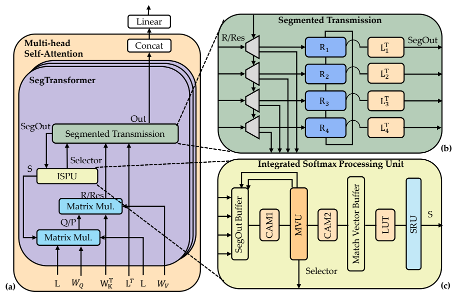

Fig. 1: The overall architecture of SegTransformer

### B. Segmented Transmission Module (STM)

To address challenge 1, we propose STM (in the top right in Fig. 1). An input vector $L$ of length $n$ generates $R$ through two matrix multiplications. $R$ is then segmented and stored in buffers $R_1$ to $R_4$ in STM (Fig. 2(a)). In Fig. 2(b), each buffer is multiplied by the corresponding $L_x^T$, producing intermediate segmented results, and then the generated $R$ is then cyclically shifted to the next buffer via circular buffers (Fig. 2(c)). If it reaches the last buffer, the segmented $R$ will be transferred back to the first buffer, repeating from Fig. 2(b). After 4 iterations, the results are generated in SegOut. The proposed STM improves resource utilization and reduces execution time.

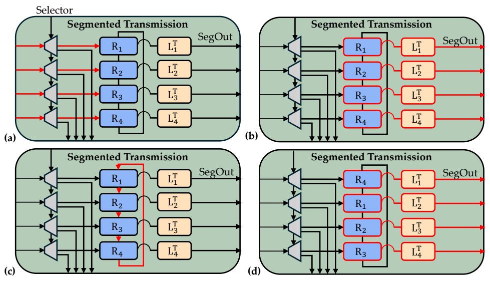

Fig. 2: The data flow of Segmented Transmission module

### C. Integrated Softmax Processing Unit (ISPU)

$$Softmax(x) = \frac{\left(e^{x_i - x_{LMs}} * e^{x_{LMs} - x_{GM}}\right)}{\sum_0^{n_S-1} e^{x_{LMs} - x_{GM}} * \sum_0^{l_S} e^{x_i - x_{LMs}}} \quad (1)$$

To address challenges 2 and 3, we propose a segmented softmax equation (Eq. 1), where $n$ is the token count, $x_i$ is the element value, $x_{GM}$ is the global row maximum, $x_{LM}$ is the local segment $s$ maximum, $n_S$ is the segment count, and $l_S$ is the segment length. We also propose ISPU, which computes softmax values with high accuracy through the following steps:

*1) FindMax:* in Fig. 3, the input sequence is stored in the SegOut Buffer and quantized❶. The quantized values are processed through CAM1 to find match vectors, which are sent to the Match Vector Array (MVA)❷. These vectors are used in an OR gate in the ISPU and then sent to the Local-Max Buffer❸. The FindMax module identifies the maximum value's position in the Local-Max Buffer❹, and the Local-Max vector is returned to the SegOut Buffer❻. The LocalMax vector is also sent to the GlobalMax Buffer for factor computations in later steps❺.

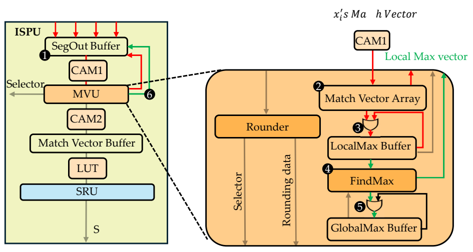

Fig. 3: The data flow of FindMax and how to find the maximum value by OR operation

*2) Subtraction and Rounding:* In Fig. 4, when the SegOut Buffer receives the match and LocalMax vectors, the values are fed into the MVU and rounded to one decimal place by the Rounder❷. The Rounder uses a counter to check whether all inputs are processed. Once done, the Selector signal directs the data to STM to generate the result❸. The rounded data are fed to CAM2 to obtain the match vector, which is then processed by LUT and stored in the Local Result Buffer❹. These values are summed in the Local Sum Buffer for later factor computation❺.

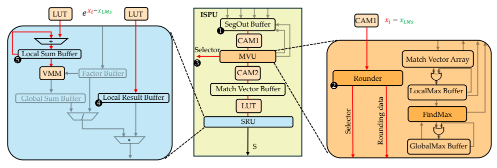

Fig. 4: The data flow of subtraction and rounding

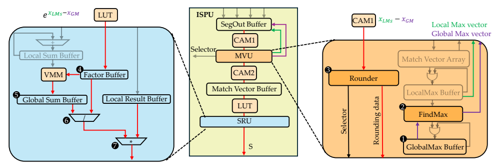

Fig. 5: The data flow of subtraction and rounding

*3) Softmax Recalibrate:* In Fig. 5, after completing local softmax computations in Step 2, the factor computation maps local results to global softmax. In the MVU module, each local maximum vector performs an OR operation with the GlobalMax buffer, and the FindMax module identifies the global maximum❶. The results pass through CAM1, Rounder❸, CAM2, and LUT, storing the factor in the Factor Buffer❹. Matrix multiplication with the Local Sum Buffer gives the denominator values for global softmax❺. Dividing by the denominator generates segment-specific factors❻, and multiplying each value by its factor gives the final result❼.

*4) Exponential Index Rounding:* In this step, exponential index rounding performs the rounding of ($x_i$- $x_j$) to one decimal place. This will restrict the value ranging from 0 to $-9.9$, corresponding to $e^0$ to $e^{-9.9}$ prestored in the LUT.

## III. Experimental Evaluation

### A. *Experimental Results*

In this experiment, we compare SegTransformer with ReTransformer [5] and STAR [4].

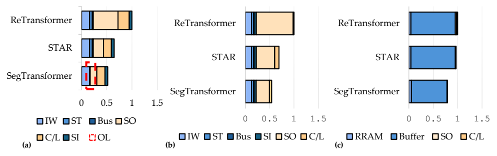

Fig. 6: The comparison of (a) latency consumption (b) power consumption (c)circuit area

Fig. 6 compares latency, power consumption, and circuit area with the state-of-the-arts. 1) Latency: Fig. 6(a) shows a 4x speedup in ST due to parallel computations in circular buffers. Our architecture reduces latency by 48.1% compared to ReTransformer [5] and 12.6% compared to STAR [4]. 2) Power Consumption: Fig. 6(b) indicates similar power values for IW, ST, and SI due to the use of four ReRAM crossbars for parallel operations. Our architecture reduces power consumption by 45.7% compared to ReTransformer and 15% compared to STAR. 3) Circuit Area: Fig. 6(c) demonstrates a 20% area reduction compared to both ReTransformer and STAR.

## Acknowledgment

This work was supported in part by the National Science and Technology Council of Taiwan (R.O.C) under grants 111-2221-E-001-013-MY3, 113-2223-E-001-001, and 113-2923-E-006-009.

## References

[1] H. Jin *et al.*, "Rehy: A reram-based digital/analog hybrid pim architecture for accelerating cnn training," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 11, pp. 2872–2884, 2021.

[2] L. Song *et al.*, "Pipelayer: A pipelined reram-based accelerator for deep learning," in *Proceedings of the 2017 IEEE international symposium on high performance computer architecture (HPCA)*, pp. 541–552, 2017.

[3] M. Zhou *et al.*, "Transpim: A memory-based acceleration via software-hardware co-design for transformer," in *HPCA 2022*, pp. 1071–1085.

[4] Y. Zhai *et al.*, "STAR: An efficient softmax engine for attention model with rram crossbar," in *Proceedings of the 2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–2, 2023.

[5] X. Yang *et al.*, "Retransformer: Reram-based processing-in-memory architecture for transformer acceleration," in *Proceedings of the 39th International Conference on Computer-Aided Design*, pp. 1–9, 2020.