AdaP-CIM: Compute-in-Memory Based Neural Network Accelerator using Adaptive Posit

Jingyu He, Fengbin Tu, Kwang-Ting Cheng, Chi-Ying Tsui
Department of Electronic and Computer Engineering
Hong Kong University of Science and Technology, Hong Kong SAR, China
{jheci, fengbintu, timcheng, eetsui}@ust.hk

***Abstract*—This study proposes two novel approaches to address memory wall issues in AI accelerator designs for large neural networks. The first approach introduces a new format called adaptive Posit (AdaP) with two exponent encoding schemes that dynamically extend the dynamic range of its representation at run time with minimal hardware overhead. The second approach proposes using compute-in-memory (CIM) with speculative input alignment (SAU) to implement the AdaP multiply-and-accumulate (MAC) computation, significantly reducing the delay, area, and power consumption for the max exponent computation. The proposed approaches outperform state-of-the-art quantization methods and achieve significant energy and area efficiency improvements.**

***Index Terms*—posit, quantization, compute-in-memory**

I. Introduction

Designing AI accelerators for large neural networks with limited on-chip hardware and memory resources presents a significant challenge. One approach to address this issue is to quantize weights into lower bit width formats with a wide dynamic range, such as AdaptivFloat [4], Posit [1], and outlier-aware quantization (OAQ) [3], [6]. However, these approaches have limitations, such as the need for offline fine-tuning and the requirement of dedicated hardware. Another approach involves the use of compute-in-memory (CIM). In the context of floating-point (FP) MAC operations in CIM, an input alignment unit is required to shift the fraction of the inputs with appropriate offsets, enabling the CIM to focus on integer MAC computations. However, conventional designs use a comparator tree (CT) to identify the maximum exponent, leading to long latency and high area and energy overheads.

To address these limitations, this study proposes two novel features: adaptive Posit (AdaP) and speculative alignment unit (SAU). First, AdaP is a novel format based on Posit, which dynamically adjusts to the data distribution at runtime with minimal hardware overhead. AdaP employs a fixed-length representation and chooses between two exponent encoding schemes for optimal runtime representation. Unlike regular Posit, where the variable-length regime may truncate the fraction, AdaP sets a predefined limit for the regime length. Second, the SAU regroups the exponents by their bit positions and performs bit-serial comparisons. The result of the MSB computation is then used to select the LSB output from a pair of pre-calculated values. This approach minimizes the critical path, significantly reducing the delay, area, and power consumption of the maximum exponent computation. By seamlessly integrating these two features, AdaP-CIM achieves a 4x higher memory density compared to FP32 at negligible accuracy loss and reduces the overall area and power consumption of the macro by 16.4% and 25.7%, respectively, compared to the baseline design.

II. Adaptive Posit Format

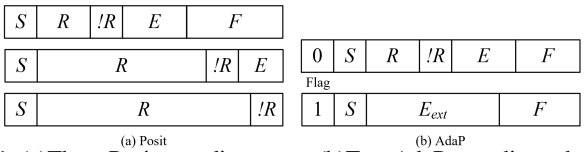

Fig. 1: (a)Three Posit encoding cases. (b)Two AdaP encoding schemes.

The conventional Posit format suffers from potential truncation of the exponent and fraction due to the length-variable regime, as shown in Fig. 1(a). Moreover, the decoding of the regime requires a leading zero counter (LZC) with a length of $n-1$-bit, which further exacerbates the area and power overhead of the CIM-based accelerator. Adaptive Posit (AdaP) reduces the LZC hardware overhead and extends the dynamic range using an intrinsic exponent extension. It introduces an additional parameter rs, which denotes the maximum run-length of the regime and distinguishes it from Posit. The condition $rs < n-1-es$ ensures that the fraction is not truncated and reduces the hardware overhead of the LZC. Given a number $AdaP(n, es, rs)$, the maximum regime value is determined as $k_{max} = rs - 1$. es and rs are determined by the dynamic range of the network parameters, and an encoding threshold T is set as $k_{max}2^{es} + e_{max} = (rs-1)2^{es} + 2^{es} - 1 = rs2^{es} - 1$. Fig. 1(b) illustrates the format of AdaP, which includes two encoding schemes indicated by the flag bit. In the first encoding scheme, when the effective exponent E is less than or equal to the threshold T, the encoding flag bit is set to 0, and the remaining fields are encoded in the same way as Posit. In the second encoding scheme, when E is greater than T, the encoding flag is set to 1, and an exponent extension, E_{ext}, is encoded in an unsigned integer format, followed by the fraction bits. For example, consider AdaP(8,1,3). If an effective exponent of 4 needs to be encoded, since $4 = (3-1)2^1 < T = (3-1)2^1 + 2^1 - 1 = 5$, the flag is set to 0, the regime is 111, and the exponent is 0. If an effective exponent of 6 needs to be encoded, since $6 > T = 5$, the flag is set to 1, and $E_{ext} = 6 - T = 1$.

III. Speculative Input Alignment

For CIM to support FP MAC [5], a comparator tree (CT) is used to determine the maximum exponent E_{max} among the

979-8-3503-4859-0/DATE24/© 2024 EDAA

exponents of the inputs. Subsequently, each input's exponent offset $E_{max} - E_i$ is calculated to shift the input's fraction before sending it into the CIM for computation. The CT-based design requires $N-1$ 4-bit comparators (FP8) and the critical path comprises $\log_2 N$ comparators and multiplexers, incurring significant area and energy overhead as the number of inputs increases. Unlike the CT approach, which compares all bits of the two inputs simultaneously at each node, the speculative Alignment Unit (SAU) compares two exponents in a bit serial manner. Fig. 2 illustrates an example implementation using 6 bits. The inputs (E or E_{ext}) are grouped by their bit positions, and $E_i[j]$ stands for the j^{th} bit of the i^{th} input, while $Emax$ represents the maximum input. An OR-gate tree is used to generate $E_{max}[5]$ based on the MSBs of all inputs. Simultaneously, two OR-gate trees are used to speculatively compute potential results at $E_{max}[4]$ for both cases when $E_{max}[5] = 0$ or $E_{max}[5] = 1$. If $E_{max}[5] = 1$, the 4^{th} significant bit of all inputs are ANDed with its MSB before being sent to the OR tree of that particular bit location. Once the upper-bit computation is completed, the result is then used to select the correct output of the next lower bits. The critical path of the SAU occurs when $E_{max}[5] = 1$, and includes 1 AND gate, $\log_2 N$ OR gates, and $\log_2 rs + es$ muxes.

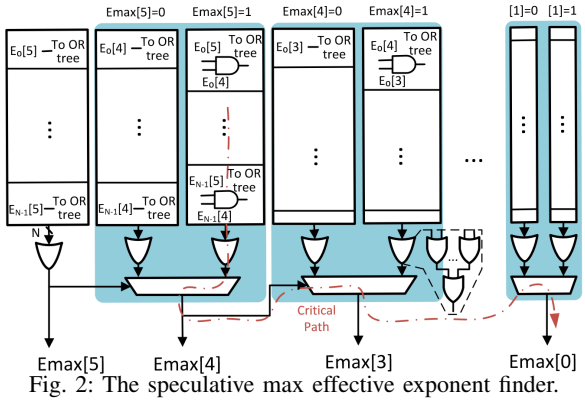

Fig. 2: The speculative max effective exponent finder.

IV. Experimental Results

We evaluate the AdaP-CIM in terms of accuracy, performance, power, and area (PPA). Table I compares the accuracy of three quantization formats at different bit widths using the BERT-Base-Uncased model and three GLUE datasets. It is important to note that AdapF's exponent bias of the activation is fine-tuned offline on the test dataset, which may not provide a fair comparison with other quantization schemes. AdaP exhibits more robustness at low precision compared to the other formats. Specifically, at 7-bit, AdaP introduces only a 0.013 lower Matthew correlation compared to the FP32 baseline, while the Posit experiences a non-negligible 0.03 drop on CoLA dataset due to insufficient maximum effective exponent and truncated fraction bits. At 9-bit, AdaP achieves the same matched accuracy as the FP32 baseline, offering a 3.56× higher storage density to alleviate the memory wall issue without accuracy loss. Overall, the E_{ext} encoding scheme of AdaP is more accurate than Posit and AdapF for large numbers, attributed to a larger maximum effective exponent and reduced clipping.

TABLE I: BERT Comparison with other Quantization Formats

	CoLA Matt. Corr.[a] ↑ @ FP32=0.586			STS-B Pear. Corr.[b] ↑ @ FP32=0.858			MNLI Mat. Acc.[c] ↑ @ FP32=0.846		
Bits	**Posit**	**AdapF**	**AdaP**	**Posit**	**AdapF**	**AdaP**	**Posit**	**AdapF**	**AdaP**
7	0.546	0.569	**0.573**	0.834	0.841	**0.849**	0.816	0.829	**0.835**
8	0.563	0.574	**0.581**	0.839	0.848	**0.852**	0.834	**0.841**	0.840
9	0.571	0.582	**0.584**	0.846	**0.856**	0.855	0.841	0.842	**0.846**

[a]Matthews correlation. [b]Pearson correlation. [c]Matched accuracy.

Table II shows the synthesized area and power breakdown of an AdaP-CIM macro and a baseline (BL) design using TSMC 28 nm PDK. The BL design [2] uses the same SRAM-based CIM macro for Posit-based MAC operations with a CT-based IAU. The macro size is 256×64, and the clock frequency is 250MHz. OC & Shift refer to the offset calculation and shifters. The AdaP format reduces the area and energy overhead of the decoders by limiting the run length of the regime field. As a result, the AdaP decoders account for only 22.53% and 10.29% of the total area and power, respectively, compared to 33.01% and 18.66% of the BL design. In addition, the SAU significantly reduces the area and power percentage of the alignment unit to 1.02% and 0.63% of the total area and power, respectively, compared to 3.04% and 15.17% of the BL, respectively. Overall, the proposed design reduces the overall area and power consumption of the macro by 16.4% and 25.7%, respectively, compared to the BL design.

TABLE II: Area and Power Breakdown

Module	**Area (um^2)**	**Per.**	**Power (uW)**	**Per.**
256x AdaP DEC	**15866**	**22.53%**	**1793**	**10.29%**
SAU	**719**	**1.02%**	**110**	**0.63%**
OC & Shift	4835	6.87%	2220	12.74%
SRAM-CIM	49000	69.58%	13299	76.34%
AdaP-CIM tot.	**70420**		**17422**	
256x Posit DEC	27784	33.01%	4376	18.66%
CT-based IAU	2559	3.04%	3557	15.17%
Baseline tot.	84177		23452	

V. Acknowledgement

This research was supported by ACCESS – AI Chip Center for Emerging Smart Systems, sponsored by InnoHK funding, Hong Kong SAR.

References

[1] J. L. Gustafson and I. T. Yonemoto. Beating floating point at its own game: Posit arithmetic. *Supercomputing frontiers and innovations*, 4(2):71–86, 2017.

[2] C.-T. Lin, D. Wang, B. Zhang, G. K. Chen, P. C. Knag, R. K. Krishnamurthy, and M. Seok. Dimca: An area-efficient digital in-memory computing macro featuring approximate arithmetic hardware in 28 nm. *IEEE Journal of Solid-State Circuits*, 2023.

[3] E. Park, D. Kim, and S. Yoo. Energy-efficient neural network accelerator based on outlier-aware low-precision computation. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pages 688–698. IEEE, 2018.

[4] T. Tambe, E.-Y. Yang, Z. Wan, Y. Deng, V. J. Reddi, A. Rush, D. Brooks, and G.-Y. Wei. Algorithm-hardware co-design of adaptive floating-point encodings for resilient deep learning inference. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.

[5] F. Tu, Y. Wang, Z. Wu, L. Liang, Y. Ding, B. Kim, L. Liu, S. Wei, Y. Xie, and S. Yin. Redcim: Reconfigurable digital computing-in-memory processor with unified fp/int pipeline for cloud ai acceleration. *IEEE Journal of Solid-State Circuits*, 58(1):243–255, 2022.

[6] A. H. Zadeh, I. Edo, O. M. Awad, and A. Moshovos. Gobo: Quantizing attention-based nlp models for low latency and energy efficient inference. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 811–824. IEEE, 2020.