Segment-Wise Accumulation: Low-Error Logarithmic Domain Computing for Efficient Large Language Model Inference

Xinkuang Geng, Yunjie Lu, Hui Wang, Honglan Jiang
Department of Micro-Nano Electronics, Shanghai Jiao Tong University, Shanghai, China
xinkuang@sjtu.edu.cn, stalker_lu@sjtu.edu.cn, ihuiwang@sjtu.edu.cn, honglan@sjtu.edu.cn

***Abstract*—Logarithmic domain computing (LDC) has great potential for reducing quantization errors and computational complexity in Large Language Models (LLMs). While logarithmic multiplication can be efficiently implemented using fixed-point addition, the primary challenge in multiply-accumulate (MAC) operations is balancing the precision of logarithmic adders with their hardware overhead. Through a detailed analysis of the errors inherent in LDC-based LLMs, we propose segment-wise accumulation (SWA) to mitigate these errors. In addition, a processing element (PE) is introduced to enable SWA in the systolic array architecture. Compared with the accumulation scheme devised for enhancing floating-point computing, the proposed SWA facilitates the integration into existing accelerator architectures, resulting in lower hardware overhead. The experimental results show that SWA allows LDC under low-precision configurations to achieve remarkable accuracy in LLMs, demonstrating higher hardware efficiency than merely increasing the precision of individual computations. Our method, while maintaining a lower hardware overhead than traditional LDC, achieves more than 13.9% improvement in average accuracy across multiple zero-shot benchmarks in LLAMA-2-7B. Furthermore, compared to integer domain computing, a logarithmic processing element array based on the proposed SWA yields reductions of 24.6% in area and 42.3% in power, while achieving higher accuracy.**

***Index Terms*—segment-wise accumulation, logarithmic domain computing, low-error, large language model**

I. INTRODUCTION

Large language models (LLMs) have achieved breakthrough success in natural language processing [1]–[3]. To meet the substantial storage and computation demands of LLMs, efficient deployment techniques have been extensively investigated both on servers and edge devices. As a commonly used model compression methodology, quantization does not alter the model architecture, while significantly reducing the resource overhead for deploying LLMs [4], [5]. By mapping the original floating-point data into a discrete number system with low representation precision, quantization significantly reduces the storage and computation requirements.

Considering that data in LLMs generally follow a long-tailed distribution, logarithmic quantization is more suitable than integer quantization [6], [7]. Furthermore, once the weights and activation values in LLMs are mapped to the logarithmic domain, the multiplication would be simplified to the fixed-point addition between their logarithmic encodings. However, addition in the logarithmic domain becomes a complex nonlinear operation, often approximated based on lookup tables (LUTs) [8], [9]. As the size of the LUT exponentially increases with the word length of the logarithmic domain, significant hardware overhead is necessary for high-precision applications [8].

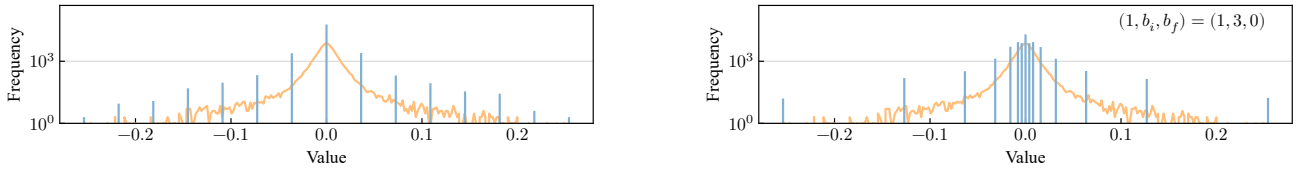
To circumvent the nonlinear operation in the logarithmic adder, [10] proposes a compact piecewise linear approximation (CPLA) based on Mitchell's algorithm [11] to perform conversions between binary and logarithmic numbers. Although some precision is traded for hardware efficiency, this method inevitably takes significant circuit resources due to the CPLA. In [7], [12], [13], certain power-of-two terms are precomputed, which is then used to convert the logarithmic domain addends to fixed-point numbers. Subsequently, a fixed-point adder is employed to sum up the converted addends. In this case, although the addition can be error-free, more efforts are needed for the accurate conversion and the high-bit-width fixed-point addition.

The objective of this work is to reduce the error under low bit-width logarithmic domain computing (LDC), making its accuracy acceptable in LLM applications while maintaining low hardware overhead. Through mathematical analysis, we identify that: 1. the error in logarithmic addition tends to increase with the magnitude of the sum; 2. LLMs often involve accumulation processes of long sequences, leading to substantial intermediate sums with high magnitude. We believe that this characteristic is the primary reason why rounding errors significantly impact the performance of low-bit-width LDC. Prior work demonstrates that floating-point addition exhibits similar error properties, and several methods focusing on reducing the errors by altering the order of summation have been proposed [14]. However, these approaches are primarily designed for high-precision operations on CPUs with high costs and are not viable to implement within existing accelerators for LLMs.

To address this challenge, we propose an enhancement scheme for individual processing elements (PEs), denoted as segment-wise accumulation (SWA), without modifying the data flow of the commonly used systolic array architecture. The proposed technique aims to maintain the magnitude of most

This work was supported by the National Natural Science Foundation of China under grant number 62374108.

978-3-9826741-0-0/DATE25/© 2025 EDAA



(a) Integer Quantization

(b) Logarithmic Quantization

Fig. 1: Quantization for the weights in the first transformer block in LLAMA-2-7B. The curve represents the distribution of the unquantized data. The vertical lines represent the frequency of the integer (left) and logarithmic (right) quantization results.

intermediate accumulation results at a relatively low level, thus mitigating the error associated with logarithmic domain addition.

The experimental results show that SWA allows LDC under low-precision configurations to achieve remarkable accuracy in LLMs, demonstrating higher hardware efficiency than merely increasing the precision of individual computations. Compared to the naive PE for LDC, our design achieves more than 22.5% improvement in average accuracy across 7 zero-shot benchmarks in LLAMA-2-7B [3]. Considering a 32×32 PE array, the proposed logarithmic accumulation scheme yields respectively 24.6% and 42.3% reductions in area and power over the conventional integer domain computing, and achieves higher accuracy in LLMs.

The rest of the paper is organized as follows. In Section II, some preliminary knowledge of LDC and the details of quantized LLMs are introduced. Section III we analyze the characteristics of the errors in logarithmic accumulation and the computing in LLM applications. In Section IV, SWA is proposed to suppress the rounding error; also, the corresponding PE architecture is devised to support SWA. In Section V, the performance of the SWA in LLMs as well as the hardware overhead of the PE array are evaluated. Finally, Section VI concludes the paper.

II. PRELIMINARIES

A. Logarithmic Domain Computing

In logarithmic domain, a real number x is encoded as $(X_{\text{sign}}, X_{\log})$, where X_{sign} indicates its sign by using 1 bit and $X_{\log}$ represents the logarithm of its absolute value $|x|$. This mapping can be expressed as

$$x = (-1)^{X_{\text{sign}}} \cdot 2^{X_{\log}}. \tag{1}$$

$X_{\log}$ is a fixed-point number with a b_i-bit integer part and a b_f-bit fractional part that determine the representation range and the resolution of the logarithmic domain, respectively. Along with the sign bit, we define $(1, b_i, b_f)$ as the encoding format of the logarithmic domain. Once the operands are encoded into the logarithmic domain, the multiplication $p = x \cdot y$ can be naturally transformed into a fixed-point addition as

$$2^{P_{\log}} = 2^{X_{\log}} \cdot 2^{Y_{\log}} \quad \rightarrow \quad P_{\log} = X_{\log} + Y_{\log}. \tag{2}$$

The sign of the product, P_{sign}, can be obtained through an XOR operation, which is not shown in (2). Although multiplication becomes very simple in the logarithmic domain, more efforts are required to implement addition $s = x + y$ as

$$\begin{aligned} 2^{S_{\log}} &= |2^{X_{\log}} \pm 2^{Y_{\log}}| \quad \rightarrow \\ S_{\log} &= \text{Max}(X_{\log}, Y_{\log}) + \phi_{\pm}(\Delta_{\log}), \end{aligned} \tag{3}$$

where $\phi_{\pm}(\Delta_{\log}) = \log_2(1 \pm 2^{-\Delta_{\log}})$ and $\Delta_{\log} = |X_{\log} - Y_{\log}|$. A common method to implement $\phi_{\pm}$ is to use $\Delta_{\log}$ to index its corresponding function value from LUTs that store all possible results for $\phi_{\pm}(\Delta_{\log})$. Approximations occur due to the limited precision values that the LUTs can handle. To store the precomputed function values into the space-limited LUTs, $\phi_{\pm}(\Delta_{\log})$ are rounded and preserved b_f fractional bits through $\mathcal{R}$ as

$$\mathcal{R}(x) = \lfloor x \cdot 2^{b_f} \rceil \cdot 2^{-b_f}, \tag{4}$$

where $\lfloor \cdot \rceil$ represents the nearest rounding operation. When considering the hardware implementation, the logarithmic domain addition would be expressed as

$$S_{\log} \approx \tilde{S}_{\log} = \text{Max}(X_{\log}, Y_{\log}) + \mathcal{R}(\phi_{\pm}(\Delta_{\log})). \tag{5}$$

The sign of the sum, S_{sign}, is set to that of the larger addend.

B. LLM Quantization and Inference

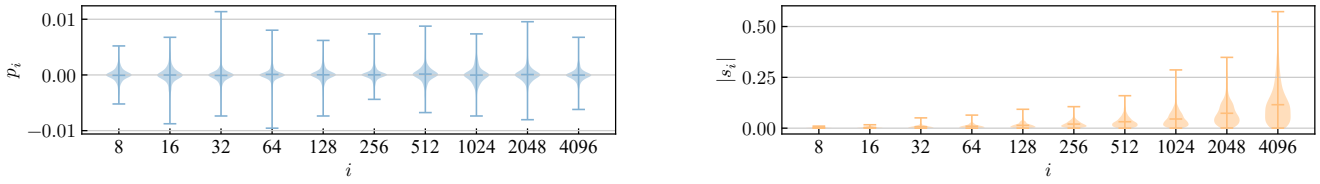
Quantization selects the closest element from a given set Q containing discrete numbers to approximate the original high-precision value as

$$\hat{x} = \arg\min_{q} |s_x q - x| \approx \frac{x}{s_x}, \ q \in Q, \tag{6}$$

where x is the original value, $\hat{x}$ is the quantized result, and the scale factor s_x adjusts the elements in Q to fit the range of x.

For integer quantization (also known as symmetric uniform quantization), Q_{int} consists of integers. For logarithmic quantization, $Q_{\log}$ contains powers of two terms and zero. Notably, when scaling the two sets of quantization points in Q_{int} and $Q_{\log}$ to the same range, $Q_{\log}$ exhibits higher resolution around zero than Q_{int}, which is consistent with the long-tailed distribution of data in LLMs, as shown in Fig. 1.

The primary operations in LLMs can be categorized as matrix multiplications, which produce elements in the output matrix by computing the inner products of the corresponding vectors from the input matrices. In quantized LLMs, elements from one vector share the same scale factor, which enables



(a) Distribution of p_i.

(b) Distribution of $|s_i|$.

Fig. 2: Distributions of the product to be accumulated, p_i, and the intermediate accumulation result, $|s_i|$, at various accumulation stages. The input vectors are randomly extracted from the first transformer block in LLAMA-2-7B.

multiply-accumulate (MAC) operations to be performed all on the set Q as

$$\hat{y}_k \approx \frac{1}{s_y} \sum s_a \hat{a}_i s_w \hat{w}_i = \frac{s_a s_w}{s_y} \sum \hat{a}_i \hat{w}_i. \tag{7}$$

Compared to the quantized inputs $\hat{a}_i$ and $\hat{w}_i$, the intermediate accumulation results have a higher bit width. Therefore, in an accelerator, frequent access to partial sums is memory inefficient. In this paper, we assume the output-stationary dataflow [15] is employed. This means that the intermediate accumulation result is held in the register until the final result is generated. The result is then quantized to a lower bit width before being sent to memory.

III. Computing Error Analysis

A. Logarithmic Domain Computing Characteristics

In the logarithmic domain, multiplication is error-free and can be executed by simply adding the two logarithmic operands. As per (3) and (5), the absolute error caused by logarithmic addition can be expressed as

$$\varepsilon_{\log} = |\tilde{S}_{\log} - S_{\log}| = |\mathcal{R}(\phi_{\pm}(\Delta_{\log})) - \phi_{\pm}(\Delta_{\log})|. \tag{8}$$

According to (4), $\varepsilon_{\log}$ can be further expressed as

$$\varepsilon_{\log} = |\underbrace{\lfloor t \cdot 2^{b_f} \rceil - t \cdot 2^{b_f}}_{\kappa}| \cdot 2^{-b_f}, \tag{9}$$

where $t = \phi_{\pm}(\Delta_{\log})$. Obviously, the nearest rounding operation $\lfloor \cdot \rceil$ produces errors ranging from -0.5 to 0.5. Since t can always be considered uniform at a small scale (2^{-b_f}), the error κ follows a uniform distribution from -0.5 to 0.5, which results in $\mathbb{E}(|\kappa|) = 0.25$. Thus, the expected absolute error due to the approximation can be calculated as

$$\mathbb{E}(\varepsilon_{\log}) = \mathbb{E}(|\kappa|) \cdot 2^{-b_f} = 2^{-b_f-2}. \tag{10}$$

Note that $\varepsilon_{\log}$ represents the absolute error in the logarithmic encoding. As per (8), the corresponding real error ε is

$$\varepsilon = |2^{\tilde{S}_{\log}} - 2^{S_{\log}}| = 2^{S_{\log}} \cdot |2^{\pm\varepsilon_{\log}} - 1|. \tag{11}$$

Here, we perform a Taylor expansion of $2^{\pm\varepsilon_{\log}}$ at zero as

$$2^{\pm\varepsilon_{\log}} = 1 + \ln 2 \cdot (\pm\varepsilon_{\log}) + \frac{1}{2!} \cdot \ln^2 2 \cdot \varepsilon_{\log}^2 + \cdots. \tag{12}$$

Considering that the error $\varepsilon_{\log}$ is generally small enough, we only remain the first two terms in the expansion. Then the real error ε can be approximated as

$$\varepsilon \approx 2^{S_{\log}} \cdot |\ln 2 \cdot (\pm\varepsilon_{\log})| = \ln 2 \cdot 2^{S_{\log}} \cdot \varepsilon_{\log}. \tag{13}$$

As per (10), the expected real error can be expressed as

$$\mathbb{E}(\varepsilon) \approx \ln 2 \cdot 2^{S_{\log}} \cdot \mathbb{E}(\varepsilon_{\log}) = \ln 2 \cdot 2^{-b_f-2} \cdot |s|, \tag{14}$$

where s is the real sum result. For a certain resolution of the logarithmic domain, i.e., fixed b_f, the expected real error of logarithmic addition is proportional to the magnitude of the sum.

B. LLM Computing Characteristics

A vector inner product in quantized LLMs can be viewed as an accumulation process as

$$s_i = s_{i-1} + a_i \cdot w_i = s_{i-1} + p_i. \tag{15}$$

In this case, the two addends of the addition are not equivalent in status: s_{i-1} is the result of the previous accumulation, while p_i is independent of s_{i-1}. As shown in Fig. 2 (a), the distributions of p_i are similar across different MAC index i. Here, we assume that the products at different accumulation steps are independently and identically distributed. According to the central limit theorem, the sum of n such products follows a normal distribution as

$$s_n = \sum_{i=1}^{n} p_i \sim \mathcal{N}(n\mu, n\sigma^2), \tag{16}$$

where μ and σ^2 are the mean and variance of the individual product, respectively. $|s_n|$ follows a folded normal distribution [16], whose mean is given by

$$\mathbb{E}(|s_n|) = \sqrt{\frac{2n}{\pi}}\sigma e^{-\frac{n}{2}\frac{\mu^2}{\sigma^2}} + n\mu \,\mathrm{erf}(\sqrt{\frac{n}{2}}\frac{\mu}{\sigma}), \tag{17}$$

The numerical analysis indicates that the statistic increases with n. In other words, theoretically, as the vector length increases, the magnitude of the accumulation result $|s_n|$ also becomes larger from a statistical perspective. Fig. 2 (b) shows the distribution of $|s_i|$ at different accumulation steps in the vector inner products in LLMs. It indicates that the magnitude of the accumulation result grows with the vector length, which is consistent with the results of theoretical analysis.

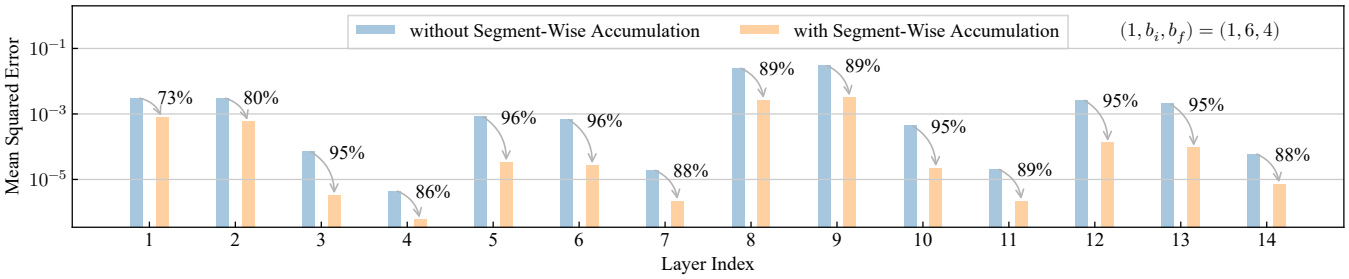


Fig. 3: Mean squared errors (logarithmic scale) of different layers in the first two transformer blocks in LLAMA-2-7B with or without segment-wise accumulation. The length of each segment is set to 128.

IV. METHODOLOGY

A. Segment-Wise Accumulation

As per the analysis presented in Section III, when accumulating a long sequence, the magnitude of the summation gradually increases, which produces an increased error in LDC. We believe that this is the underlying cause of the high error exhibited by LDC in LLMs.

The error in floating-point addition is also proportional to the magnitude of the result, as the rounding error in the mantissa is scaled by the exponent. Previous works have focused on reducing the error of floating-point accumulation in scenarios requiring extremely high precision, no matter the cost. For instance, some techniques minimize the intermediate results during accumulation by reordering the sequence [14]. These methods are not viable in large-scale computations, due to their associated high overheads in computations.

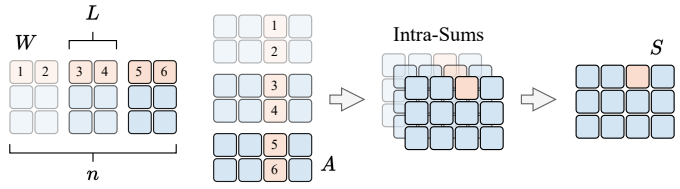


Fig. 4: Segment-wise accumulation with the local length of L.

To reduce the accumulation error in the logarithmic domain while maintaining high hardware efficiency, we propose segment-wise accumulation (SWA). As shown in Fig. 4, the long vectors in the two input matrices are divided into several segments, and the inner product for each segment is computed to obtain the corresponding intra-sum independently. Then the final accumulation result can be generated by adding up the intra-sums. SWA can be formulated as

$$s_n = \sum_{j=1}^{n/L} \sum_{i=1}^{L} a_{(j-1)L+i} \cdot w_{(j-1)L+i}. \tag{18}$$

This technique ensures that the intra-sum within each segment remains relatively small, thereby significantly reducing the accumulation error. Although the accumulation of the intra-sums from each segment may incur large errors, they constitute only a minor portion of the total additions, provided that the length of each segment is not excessively short.

To verify the effectiveness of the proposed SWA, we apply it to implement the inference of the 8-bit quantized LLAMA-2-7B. The mean squared errors (MSEs) between the original accurate outputs and the approximate outputs are then computed. The approximate outputs are produced using LDC with or without SWA. The output MSEs of 14 matrix multiplications in the first two transformer blocks in LLAMA-2-7B are shown in Fig. 3. It can be seen that SWA effectively suppresses the error caused by LDC. Compared to the case without using SWA, the MSEs for most operations are reduced by over 90%.

B. Hardware Design

In prior works, to enhance the speed of individual inner products, partial computation tasks are distributed across multiple CPU cores for parallelism and followed by an accumulation of the results [17]. However, when focusing on matrix multiplication that involves a large number of vector inner products, exploiting parallelism within individual inner products is less efficient. Furthermore, in the context of LLM accelerators that commonly feature systolic array architectures, it is not feasible to distribute the computation of one inner product across different processing elements (PEs) during matrix multiplication. Such a distribution scheme would lead to the failure of data reuse along both dimensions of the two-dimensional PE array. Although the use of the tree-structure accumulation generally used in convolutional neural networks allows for implicit segmentation of the vectors, it is less favorable than systolic array architecture for LLMs due to the long length of the vectors. Moreover, the high complexity of logarithmic domain addition makes the situation worse.

In summary, current architectures for accelerating the inference of LLMs fail to efficiently perform SWA while maintaining computational efficiency. To address this challenge, we propose an enhancement solely to a commonly used PE to reduce the error in LDC, leveraging the locality of the intra-sums.

Fig. 5 depicts the original architecture of the commonly used PE, where the input data flow into the PE from the left and top, and are then stored into W Reg and A Reg. After performing multiplication, the product is accumulated to S Reg 0. Once

the final result is obtained, S Reg 0 offloads the accumulation result sequentially.

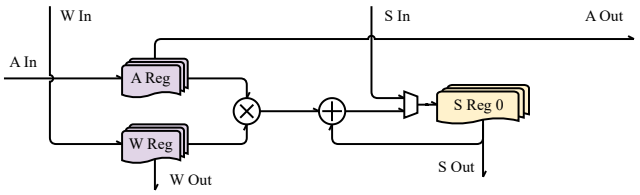


Fig. 5: Naive PE architecture.

Fig. 6 shows the proposed PE to support SWA, where two operation modes are available for accumulation. When performing a vector inner product for each segment, the product is accumulated to S Reg 0 as the naive PE does. After completing the computation for the current segment and generating the intra-sum, an additional clock cycle is used to accumulate the intra-sum from S Reg 0 into S Reg 1. In this cycle, the elements in W Reg and A Reg must be retained to ensure that the input data flow in the PE array is not affected, and S Reg 0 is cleared to zero to prepare for the next segment. Additionally, in the cycle for accumulating the intra-sum of the final segment, S Reg 0 would retain the result instead of being cleared. This guarantees the final result to be offloaded using the same register as the naive PE.

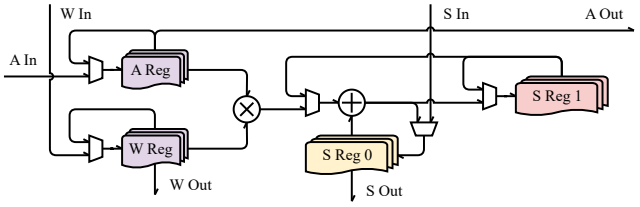


Fig. 6: PE architecture for segment-wise accumulation.

V. Evaluation

A. Experimental Setup

The proposed segment-wise accumulation scheme is evaluated based on 7 common sense reasoning benchmarks and 1 language modeling benchmark (perplexity) across OPT-350M [2] and LLAMA-2-7B [3]. The considered LLMs are quantized to 8 bits; tensor-wise quantization is employed, i.e., the elements within the same tensor share the same scale factor, resulting in minimal memory and hardware overheads. In the context of logarithmic quantization, the bit-width configuration $(1, b_i, b_f)$ is set to $(1, 5, 3)$ to balance between representation range and resolution. To simulate the accuracy of LLMs with approximate logarithmic adders, we rewrite the quantized operators in PyTorch using CUDA C++, and replace the addition operations with the functions that mimic the approximate hardware behavior.

A scheme denoted as Kulisch [7], [12], [13] is considered as an alternative MAC architecture in logarithmic quantization. This method shifts the precomputed powers of two to convert logarithmic domain products into fixed-point numbers and then performs accumulation in the fixed-point domain. Given that this technique does not introduce any errors during accumulation, it serves as a reference for the upper precision bound of LDC. The precision of the precomputed values and the bit width of the fixed-point adder are configured as those used in [12]. Moreover, CPLA [10] is considered for approximately implementing the logarithmic addition. Unlike the Kulisch method, CPLA produces the sum in logarithmic encoding. Integer domain computing is also considered for comparison, which employs symmetric uniform quantization [4].

The proposed SWA suppresses the errors without altering the adder itself; instead, it focuses on the enhancement at the PE architecture level. Therefore, we measure the circuit characteristics of a 32×32 PE array for evaluating the hardware efficiency of the proposed scheme. The PE array is designed using Verilog HDL and synthesized using Synopsys Design Compiler to generate the netlist and area information. Based on the randomly extracted data during the inference process of LLAMA-2-7B, the waveform files are generated by Verilog Compile Simulator and then used to assess the power consumption by PrimeTime PX. All experiments are conducted using a 28 nm CMOS technology with a supply voltage of 0.9 V and a clock frequency of 500 MHz.

B. Results and Analysis

Table I lists the accuracy and the perplexity of the 2 LLMs across different computing schemes. The hardware characteristics of the PE arrays using different computing schemes are shown in Table II. For the naive LDC, $(1, b_i, b_f)$ is set to $(1, 6, 4)$ and $(1, 6, 5)$ to evaluate the performance under different precision configurations. For the LDC using SWA, due to the additional hardware overhead introduced by the enhancement to the PE architecture, we assess two formats with lower precision, $(1, 6, 3)$ and $(1, 6, 4)$, to ensure a comparable area and power. Notably, in LDC with SWA, the length of each segment is set to 128. This means that after performing 128 original MAC operations, each PE would take one clock cycle to accumulate the intra-sum. Compared to the straightforward implementation, our approach results in an increase of 0.78% in the number of clock cycles, which is deemed negligible.

By applying SWA, the errors in logarithmic computations are notably reduced. For the case where $b_f = 4$, the proposed SWA results in higher accuracy and lower perplexity in both of the evaluated LLMs. Specifically, SWA respectively improves the average accuracy on OPT-350M and LLAMA-2-7B by 3.8% and 22.5%, indicating a more pronounced benefit in larger models. Although SWA incurs additional hardware overhead under the same b_f, it allows LDC under low-precision configuration ($b_f = 3$) to achieve better performance than the naive LDC with high-precision configuration ($b_f = 5$). When considering this case, applying SWA to LDC leads to an average accuracy improvement of 13.9% and a perplexity reduction of 185.0, while simultaneously achieving reductions in area, power, and delay by 16.35%, 6.78%, and 10.68%, respectively. It can

TABLE I: Zero-shot accuracy (higher is better) of 7 common sense reasoning benchmarks and perplexity (lower is better) of 1 language modeling benchmark on OPT-350M and LLAMA-2-7B across 6 computing schemes.

Quant.	MAC	b_f	RTE	COPA	HellaSwag	PIQA	WinoGrande	ARC-E	ARC-C	**Average ↑**	**WikiText-2 ↓**
OPT-350M	Full-precision	-	52.0	72.0	32.0	64.4	52.3	44.1	20.8	**48.2**	**22.0**
Int-8	Integer	-	52.3	68.0	31.7	65.1	52.7	43.6	20.1	**47.7**	**22.4**
Log-8	Kulisch [12]	-	52.7	70.0	32.0	64.3	52.1	43.7	21.0	**48.0**	**22.1**
	CPLA [10]	5	52.0	68.0	61.5	30.2	50.4	40.6	21.4	**46.3**	**46.5**
	LDC [9]	4	51.6	63.0	29.5	57.7	52.2	37.2	19.5	**44.4**	**75.4**
		5	52.3	71.0	31.7	63.4	50.5	42.0	20.1	**47.3**	**24.3**
	LDC with SWA	3	56.3	70.0	31.2	62.7	49.5	41.4	20.4	**47.3**	**26.4**
		4	52.7	70.1	32.1	64.4	51.9	44.1	20.9	**48.2**	**22.8**
LLAMA-2-7B	Full-precision	-	63.2	87.0	57.6	78.3	67.2	69.3	40.0	**66.0**	**5.5**
Int-8	Integer	-	54.9	77.0	46.0	68.8	61.6	62.5	32.4	**57.6**	**10.0**
Log-8	Kulisch [12]	-	59.2	87.0	56.6	77.6	68.0	68.9	40.1	**65.4**	**5.6**
	CPLA [10]	5	46.2	63.0	51.0	25.7	51.9	25.9	23.7	**41.1**	**77,180.8**
	LDC [9]	4	53.8	56.0	25.6	51.2	51.4	25.2	22.6	**40.8**	**359,121.2**
		5	56.3	67.0	29.9	56.7	50.6	30.8	21.8	**44.8**	**192.9**
	LDC with SWA	3	54.5	81.0	49.9	72.3	59.4	61.3	32.8	**58.7**	**7.9**
		4	58.8	85.0	55.1	76.6	63.1	65.6	39.0	**63.3**	**5.9**

TABLE II: Area, power and delay for 32×32 PE arrays.

Quant.	MAC	b_f	Area	Power	Delay
			mm^2	mW	ns
Int-8	Integer	-	0.353	314.8	1.364
Log-8	Kulisch [12]	-	0.405	454.1	1.727
	CPLA [10]	5	0.317	253.3	1.868
	LDC [9]	4	0.258	163.6	1.395
		5	0.318	194.7	1.517
	LDC with SWA	3	0.266	181.5	1.355
		4	0.319	217.5	1.503

be seen that our method demonstrates higher efficiency than merely increasing the precision of individual computations.

Compared to the Kulisch method presented in [12], although SWA effectively reduces the errors in LDC, it incurs a certain degree of precision loss due to the inherent approximation. Table I indicates that, in the smaller model OPT-350M, LDC with the proposed SWA under $b_f = 4$ exhibits comparable performance to the Kulisch method across various benchmarks. In the larger model LLAMA-2-7B, the impact of the approximation becomes more pronounced; but even in this case, it only shows an average accuracy degradation of 2.2%, which can be considered acceptable, especially when compared with the over 20% accuracy degradation observed in the naive LDC. Moreover, since the Kulisch method involves format conversion and high-precision fixed-point addition, it incurs higher hardware overhead, as shown in Table II. Conversely, our design under $b_f = 4$ can be regarded as more hardware-efficient, offering reductions of 21.23% in area, 52.10% in power, and 12.97% in delay.

Compared with the CPLA method, it can be predicted that LDC with SWA exhibits a larger area under the same b_f according to Table II. However, our design under $b_f = 3$ achieves particularly better performance compared to CPLA under $b_f = 5$. In this case, LDC with SWA results in reductions of 16.09% and 28.40% in area and power, respectively.

As the most commonly used computing scheme, integer domain computing exhibits suboptimal performance compared to our method under 8-bit tensor-wise quantization in LLMs, due to its inefficient uniform quantization. In LLAMA-2-7B, even at the low precision ($b_f = 3$), LDC with SWA outperforms integer domain computing, demonstrating an 1.1% average improvement in accuracy and a 2.1% reduction in perplexity. Concurrently, our method achieves reductions of 24.65% in area and 42.34% in power.

VI. CONCLUSION

In this paper, we propose segment-wise accumulation to suppress the errors due to approximate logarithmic domain computing in LLM applications. The key to SWA is to reduce the expected magnitude of the intermediate sums by breaking the long accumulation chain. This allows for more addition operations to be performed in the circumstance where logarithmic domain computing exhibits lower errors. To integrate the proposed technique into existing systolic array architectures without altering the data flow or introducing extra memory accesses, a PE with dual accumulation routes is proposed. The experimental results show that SWA significantly enhances the performance of approximate logarithmic domain computing, achieving lower hardware overhead and higher accuracy compared to the commonly used integer domain computing.

REFERENCES

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[2] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin *et al.*, "Opt: Open pre-trained transformer language models," *arXiv preprint arXiv:2205.01068*, 2022.

[3] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[4] M. Nagel, M. Fournarakis, R. A. Amjad, Y. Bondarenko, M. Van Baalen, and T. Blankevoort, "A white paper on neural network quantization," *arXiv preprint arXiv:2106.08295*, 2021.

[5] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han, "SmoothQuant: Accurate and efficient post-training quantization for large language models," in *International Conference on Machine Learning*. PMLR, 2023, pp. 38 087–38 099.

[6] E. H. Lee, D. Miyashita, E. Chai, B. Murmann, and S. S. Wong, "LogNet: Energy-efficient neural networks using logarithmic computation," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 5900–5904.

[7] S. Vogel, M. Liang, A. Guntoro, W. Stechele, and G. Ascheid, "Efficient hardware acceleration of cnns using logarithmic data representation with arbitrary log-base," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2018, pp. 1–8.

[8] I. Kouretas, C. Basetas, and V. Paliouras, "Low-power logarithmic number system addition/subtraction and their impact on digital filters," *IEEE Transactions on Computers*, vol. 62, no. 11, pp. 2196–2209, 2012.

[9] B. Parhami, "Computing with logarithmic number system arithmetic: Implementation methods and performance benefits," *Computers & Electrical Engineering*, vol. 87, p. 106800, 2020.

[10] W. Zhang, X. Geng, Q. Wang, J. Han, and H. Jiang, "A low-power and high-accuracy approximate adder for logarithmic number system," in *Proceedings of the Great Lakes Symposium on VLSI 2024*, 2024, pp. 125–131.

[11] J. N. J. Mitchell, "Computer multiplication and division using binary logarithms," *Ire Transactions on Electronic Computers*, vol. EC-11, no. 4, pp. 512–517, 1962.

[12] J. Johnson, "Rethinking floating point for deep learning," *arXiv preprint arXiv:1811.01721*, 2018.

[13] Y. Wang, D. Deng, L. Liu, S. Wei, and S. Yin, "Pl-npu: An energy-efficient edge-device dnn training processor with posit-based logarithm-domain computing," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 10, pp. 4042–4055, 2022.

[14] N. J. Higham, "The accuracy of floating point summation," *SIAM Journal on Scientific Computing*, vol. 14, no. 4, pp. 783–799, 1993.

[15] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," *ACM SIGARCH computer architecture news*, vol. 44, no. 3, pp. 367–379, 2016.

[16] M. Tsagris, C. Beneki, and H. Hassani, "On the folded normal distribution," *Mathematics*, vol. 2, no. 1, pp. 12–28, 2014.

[17] J. Demmel and H. D. Nguyen, "Fast reproducible floating-point summation," in *2013 IEEE 21st Symposium on Computer Arithmetic*. IEEE, 2013, pp. 163–172.