Oiso: Outlier-Isolated Data Format for Low-Bit Large Language Model Quantization

Lancheng Zou, Shuo Yin, Mingjun Li, Mingzi Wang, Chen Bai, Wenqian Zhao, and Bei Yu, *Senior Member, IEEE*

***Abstract*—The scale of large language models (LLMs) has steadily increased over time, leading to enhanced performance in multimodal understanding and complex reasoning, but with significant execution overhead on hardware. Quantization is a promising approach to reduce computation and memory overhead for LLM deployment. However, maintaining accuracy and efficiency simultaneously is challenging due to the presence of outliers. Moreover, low-bit quantization tends to deteriorate accuracy due to its limited precision. Existing outlier-aware quantization/hardware co-design methods split the sparse outliers from the normal values with dedicated encoding schemes. However, such separation produces a nonuniform data format for normal values and outliers, leading to additional hardware design and inefficient memory access. This article presents an outlier-isolated data format (Oiso) for low-bit LLM quantization called Oiso. Oiso is a unified representation for both outliers and normal values. It isolates the normal values from the outliers, which can reduce the impact of outliers on the normal values during the quantization process. Taking advantage of the uniform format, Oiso arithmetic can be performed using a homogeneous computational unit, and Oiso values can be stored in a standardized format. Hierarchical block encoding with a sub-block alignment scheme is introduced to reduce the encoding cost and the hardware overhead. We introduce the Oiso architecture, equipped with Oiso processing elements and encoders tailored for Oiso arithmetic, realizing efficient low-bit LLM inference. Oiso quantization can push the limits of low-bit LLM quantization, and the Oiso accelerator outperforms the state-of-the-art outlier-aware accelerator design with 1.26× performance improvement and 25% energy reduction.**

***Index Terms*—Hardware acceleration, Neural network compression.**

I. Introduction

LARGE language models (LLMs) [1], [2], [3] have revolutionized various fields with superior performance in natural language processing tasks, such as translation, summarization and question answering. Despite the flashy performance, the deployment of LLMs faces many practical obstacles. One of the primary concerns is the considerable computational, memory, and energy costs of LLM training and inference, which increase with the growing size and complexity of mainstream LLMs. To illustrate, the GPT-3 model [1], comprising 175 billion parameters, necessitates 350 GB of memory for loading and requires a minimum of five A100-80G GPUs to perform inference. Moreover, it's estimated that ChatGPT service requires an average electricity consumption of 564 MWh per day [4].

Quantization [5], [6], [7], [8] is one of the model compression methods that transforms the model from the high-precision floating-point (FP) to low-precision data formats, e.g., integer. The compression of the model size and reduction of the computational cost it brings facilitate the acceleration of inference and savings in memory and energy [9], [10], [11], [12], [13]. Given LLMs' considerable computational and energy overheads, it is imperative to underscore quantization's pivotal role in ensuring their sustainability.

Architecture researchers have endeavored to address the outliers that emerge in the models with algorithm and hardware co-design [14], [15], [16], [17], [18], [19]. GOBO [14] and OLAccel [15] utilize the coordinate list to store the locations of sparse outliers. BiScaled-DNN [18] adopts block sparse indices format to index the outliers. Besides, DRQ [19] uses a direct bitmap for the outliers. The locations of the outliers are sparsely encoded, and they will be quantized into higher-precision data formats, such as INT8 and INT16. Olive [17] introduces an outlier-victim pair quantization which sacrifices the adjacent normal values to accommodate the outliers. For the Olive quantization framework, Flint [16] is leveraged for normal values, and an adaptive biased float is proposed for outliers.

Despite the ability of aforementioned outlier-aware architectures to effectively extract outliers from data distributions, the nonuniform data formats necessitate additional architectural overhead for codec and outlier computation. Due to the limited dynamic range of integers, integer-based quantization methods require additional encoding to extend their range to accommodate the intricate data distribution characteristics of LLMs. Furthermore, it is not prudent to tailor the architectural design to accommodate sparse outliers, as they comprise only a tiny fraction of the activation values (i.e., $< 0.1\%$). Moreover, they fixate on the outliers excessively, isolating them from the normal directly, which may result in the loss of potential opportunities to enhance precision and efficiency. For instance, further improvements to the model's accuracy can be achieved by implementing a methodology capable of optimizing both outliers and nonoutliers.

Received 16 February 2025; revised 19 May 2025; accepted 22 June 2025. Date of publication 1 July 2025; date of current version 22 January 2026. This work was supported in part by the Research Grants Council of Hong Kong, SAR, under Project CUHK14208021, and in part by the MIND Project under Grant MINDXZ202404. This article was recommended by Associate Editor N. K. Jha. *(Corresponding authors: Chen Bai; Wenqian Zhao.)*

Lancheng Zou, Shuo Yin, Mingjun Li, Mingzi Wang, Wenqian Zhao, and Bei Yu are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, SAR (e-mail: wqzhao@cse.cuhk.edu.hk).

Chen Bai is with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong (e-mail: cbai@link.cuhk.edu.hk).

Digital Object Identifier 10.1109/TCAD.2025.3585023

In order to circumvent the cumbersome encoding design to compensate for the dynamic range of integers, some researchers adopt block floating point (BFP) [20] for LLM quantization. It is suitable for LLM quantization due to its extensive dynamic range and hardware efficiency [21], [22], [23], [24]. BFP utilizes the largest exponent within a given block as the shared exponent. If an outlier is present within the block, its exponent is considerably larger than the other values. When other values are subjected to quantization, all the smaller values are likely to be transformed into 0, resulting in a significant loss of accuracy. Unlike traditional BFP, BiE [22] leverages the bi-shared exponents, one as the shared exponent of the normal part and the other as the shared exponent of the outlier part. Using two shared exponents enables the effective partitioning of more normal-outlier pairs, which can isolate the impact of outliers on normal value quantization, thereby reducing quantization errors. However, its excessive flexibility significantly increases the required hardware resources. Thus, the issue of hardware efficiency remains inadequately addressed, which hinders practical use.

According to the above analysis, we propose a novel outlier-isolated architecture for LLM quantization, designated as outlier-isolated data format (Oiso), which adopts *bi-shared exponents* per block as BiE [22]. A *hierarchical block encoding* with *sub-block alignment* scheme is introduced to reduce the hardware overhead caused by the bi-shared exponents. The block is divided into several sub-blocks of the same size. The top-level block encoding facilitates the extraction of bi-shared exponents for the entire block. The bottom-level block encoding determines the type of each sub-block, subsequently conducting quantization in accordance with top-level block information. Sub-block alignment dictates that elements within the same sub-block must belong to the same type (i.e., they must be all outliers or normal values). This way, the hardware efficiency is significantly improved with a slight but acceptable accuracy loss. Compared to BFP and BiE, Oiso provides more flexibility for design space exploration, allowing for a better tradeoff between accuracy and efficiency.

Oiso facilitates low-bit LLM quantization, attaining an admirable equilibrium between model accuracy and hardware efficiency. Its superiority can be attributed to the following reasons. First, the data format of Oiso leverages a *uniform representation* of both outlier and normal parts. In this case, it circumvents the additional architectural designs for handling outliers separately. Additionally, it eliminates the potential for unaligned memory access, which could otherwise be a hardware-unfriendly operation. Second, Oiso takes the opportunity to optimize both the outlier and the normal value to reduce the quantization error further. Besides handling the *explicit outliers* (with extremely large magnitudes) that are responsible for the failure of low-bit LLM quantization, Oiso is also able to optimize *implicit outliers* (relatively large values in the normal values) effectively. Third, Oiso can readily employ a *reduced bit width* due to its superior accuracy, consequently diminishing the overheads associated with computation and memory access.

As a plug-and-play post-training quantization (PTQ) method, the Oiso quantization framework obviates the necessity for fine-tuning and enables rapid deployment. It requires only the statistics on weights and activations from a small calibration dataset. Oiso's PTQ using 4-bit signed mantissa can achieve comparable accuracy with 8-bit Olive's PTQ results for OPT [2] and Llama2 [3]. We also describe integrating Oiso into existing accelerator architectures, e.g., systolic array in Google TPUs [13].

We make the following contributions in this article:

1) We introduce a novel Oiso called Oiso with a uniform representation of both the outlier and the normal value. The fundamental objective of Oiso is to isolate the normal values from outliers, thereby mitigating the influence of outliers on the quantization of normal values.
2) We propose a hierarchical block encoding method with a sub-block alignment scheme that aligns the elements in the sub-block to the same type to leverage the sparsity of outliers.
3) We investigate the methodology to identify outliers and find the potential to further reduce quantization error through leveraging implicit outliers.
4) We propose the Oiso processing element processing elements (PE) and encoder hardware implementation, which can be integrated into the existing architectures, e.g., a systolic array.
5) Benefiting from the effective quantization algorithm and architecture co-design, the experimental results show that Oiso can achieve resilient low-bit LLM quantization and outperform the-state-of-the-art accelerator Olive with 1.26× performance improvement and 25% energy reduction. Thus, the resilience of Oiso allows for a more general and practical deployment of low-bit LLM inference.

II. Background and Motivation

A. Challenges in LLM Quantization

Integer quantization maps high-precision numbers like IEEE single-precision FP (FP32) and half-precision FP (FP16) into discrete integer numbers like INT8 and INT4. It can increase the computational efficiency and reduce the memory footprint of the large-scale model for practical deployment. The typical INT8 quantization process can be formulated as follows:

$$\mathbf{X}_q = \left\lceil \frac{\mathbf{X}}{S} \right\rfloor, \quad S = \frac{\max(|\mathbf{X}|)}{2^{N-1} - 1} \tag{1}$$

where $\boldsymbol{X}$ is an FP32 matrix, $\boldsymbol{X}_q$ represents the quantized matrix in INT8, $\lceil \cdot \rfloor$ denotes the rounding function, e.g., rounding to the nearest scheme. S is the scaling factor, and N is the number of bits in quantized data format ($N = 8$ in this case).

However, outliers give rise to an anomalously significant scaling factor in the context of LLM quantization. According to (1), this would result in most normal values being quantized to 0 following the scaling and rounding procedures, thus introducing significant quantization errors. Concurrently, clipping the sparse outliers will result in a significant accuracy degradation, which illustrates the importance of outliers in maintaining the functional integrity of LLMs [25]. Thus, it is essential to appropriately address both the outlier and the normal value for LLM quantization.

Moreover, although quantization-aware training (QAT) can achieve better quantization accuracy, PTQ is the prevailing

methodology for LLM quantization due to its efficiency. However, existing PTQ methods without any fine-tuning are only capable of maintaining accuracy at 8-bit [25], [26], [27], which poses a challenge of low-bit LLM PTQ.

B. Block Floating-Point

BFP [20], also known as microscaling data formats with integer version (MXINT) [28], is a particular variant of floating point. Different from FP, the elements within a block will share a common exponent. Thus, there is only one exponent in one block in BFP, which also reduces the exponents' memory footprint. The data format of BFP is illustrated in Fig. 1.

The advantages of BFP are twofold. First, BFP has a more extensive dynamic range than integers due to a shared exponent. Second, it is more efficient than FP due to the reduced bit width and the fact that BFP arithmetic is primarily low-overhead INT arithmetic, as opposed to the costly FP arithmetic. Given its advantages, BFP is widely explored for DNN model inference and training to benefit from its superiority [22], [29], [30], [31]. Additionally, some commercial hardware architectures also integrate BFP PE to accelerate DNN inference [32], [33], [34], [35], which substantiates the practical utility of BFP.

We can convert $\mathbf{X}$ in FP16 with N elements into $\mathbf{X}^{\text{BFP}}$ in the data format of BFP as follows:

$$2^{e_m}\left[(-1)^{s_0}m'_0, (-1)^{s_1}m'_1, \ldots, (-1)^{s_{N-1}}m'_{N-1}\right] \tag{2}$$

where

$$e_m = \max_i e_i, i \in 0, 1, \ldots, N-1 \tag{3}$$

$$m'_i = m_i >> (e_m - e_i), i \in 0, 1, \ldots, N-1 \tag{4}$$

where $>>$ is the right shift operation, s_i is the sign bit for each element, e_m denotes the shared exponent with E bits, m'_i represents the private mantissa of the ith elements with M bits.

The dot product is the basic computation in the general matrix multiplication (GEMM), and a significant portion of the computational workload in DNN can be attributed to GEMM operations. To show more clearly the efficiency of BFP arithmetic, we take the example of the dot product of BFP.

Given two blocks $\mathbf{X}_1^{\text{BFP}}$ and $\mathbf{X}_2^{\text{BFP}}$ with N elements in BFP format, the calculation will be donated as

$$2^{e_{m_1}+e_{m_2}}\sum_{i=0}^{N-1}\left((-1)^{s_{1,i}\oplus s_{2,i}}m_{1,i}\cdot m_{2,i}\right) \tag{5}$$

where e_{m_1} and e_{m_2}, are the shared exponent of $\mathbf{X}_1$ and $\mathbf{X}_2$, $\bigoplus$ is an XOR operation, $s_{1,i}$ and $s_{2,i}$ are the sign bit for each element, $m_{1,i}$ and $m_{2,i}$ are the private mantissa of each element. It is important to note that $m_{1,i}\cdot m_{2,i}$ is an integer multiplication operation, and the partial sum is calculated in integer addition. Consequently, the dot product of two blocks is more efficient than the counterpart of FP.

C. Motivation

1) Uniform Representation: Existing outlier-aware accelerators adopt different strategies to handle outliers.

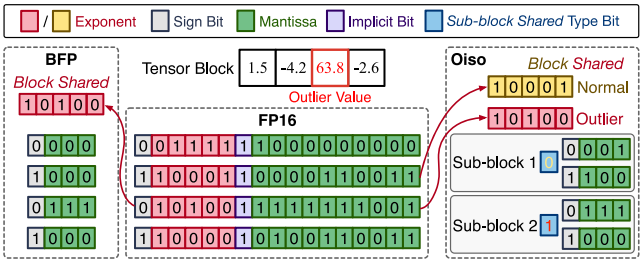

Fig. 1. Illustration of different numerical representation formats: FP16, BFP and Oiso.

OLAccel [15] and GOBO [14] leverage low-precision quantization for normal values and high-precision quantization for outliers with a coordinate list to memory the index of outliers. BiScaled-DNN [18] exploits different scaling factors for normal values and outliers and quantizes them into the same bit-width format. The quantized values are uniform, but the required block sparse index will lead to unaligned memory access. Moreover, the outliers in LLMs are unsuitable for using different scaling factors because outliers persist in fixed channels, as shown in Fig. 2. Per-channel activation quantization is challenging to map to hardware accelerators due to the unfriendly memory access, leading to severe performance degradation. DRQ [19] also utilizes different precision for outliers and normal values with an aligned bitmap to indicate outliers. Olive [17] employs Flint [16] for normal values quantization and adaptive floating point for outliers. If some layers are sensitive to quantization, it will increase the bit width, which can reduce the quantization error.

In conclusion, the nonuniform representations proposed by previous works decrease the hardware efficiency. On the one hand, the nonuniform representation requires additional architectural design to cope with codecs and computational units for different data formats. On the other hand, it will also lead to unfriendly memory access due to the variable-length data format and unsupported quantization process.

Thus, a uniform representation can be more efficient for outlier-aware LLM quantization. Oiso's design is based on the concept of uniformity, which addresses the shortcomings of previous works.

2) Leverage Implicit Outliers: The knock-on effect of nonuniform representations is that the opportunity is missed to leverage implicit outliers to further reduce quantization error and improve hardware efficiency. Prior outlier-aware quantization works find a significant threshold (e.g., 10 for the example in Fig. 2) to split the data into two parts: explicit outliers and normal values. However, the distribution of explicit outliers is sparse, and the additional architectural design may occupy over half of the total PE area [14], [15]. Thus, there is still plenty of room for more cost-effective design.

Concurrently, some contradictory scenarios may emerge. For instance, the threshold may be lowered, enabling implicit outliers to utilize high-precision quantization. It could enhance the model's accuracy and improve the utilization rate of outlier hardware. However, adopting a high-precision data format will

diminish hardware efficiency due to the increased bit width and computational cost.

The proposed Oiso circumvents this problematic situation by fully leveraging the potential of implicit outliers to enhance the precision of quantization accuracy with the help of uniform representation.

3) Limitations of Existing Block-Based Floating Point: The inherent limitations of integer quantization exacerbate the difficulties of low-bit LLM quantization. Some researchers have begun to seek new data formats to ease the burden of quantization algorithm design.

BFP represents a promising alternative for quantization, offering an extensive dynamic range and hardware efficiency. One representative configuration of BFP is BFP4, which packs 16 4-bit sign magnitude integers with one shared exponent to represent a 16-tuple vector. Nevertheless, the vanilla data format of BFP needs to be more flexible to facilitate further improvements in model accuracy. The accuracy will be drastically reduced when using a lower bit width for quantization.

OPAL [24] proposes an outlier-preserved BFP for LLM quantization. It preserves the outliers in bfloat16 and extracts tensor-wise and block-wise scales for shared exponent. However, the encoding has a high overhead because it requires scanning all the values in the tensor before determining the max exponent for tensor-wise scale and bias values for block-wise scale. The nonuniform data format will also lead to unaligned memory access and a lack of utilization of implicit outliers. BiE [22] introduces a variant of BFP where each block has two exponents to address the issues caused by outliers. Nevertheless, the hardware overhead is considerable due to its fine-grained outlier-normal encoding, preventing it from being efficiently implemented.

Therefore, a more flexible and efficient variant is needed to push the limits of BFP for more resilient LLM quantization.

III. Outlier-Isolated Data Format

This section commences with a theoretical analysis of how the accuracy of the vanilla BFP format can be improved. Then, we introduce the detailed information of our proposed Oiso with *hierarchical block encoding*. The sub-block analysis shows the potential using *sub-block alignment* to reduce the cost of the fine-grained outlier-normal encoding and corresponding hardware design. The data format of the proposed Oiso adheres to the design criterion for the uniform representation of both the outlier and the normal value. By isolating both *explicit outliers* and *implicit outliers*, the Oiso quantization framework can further reduce the quantization error.

A. Lesson Learned From BFP Quantization

The first question is: What can we do to further optimize the data format of BFP? Before delving into the design of the proposed data format, it is essential to grasp the characteristics of BFP and the associated computational error analysis.

A tensor $\mathbf{X}_q$ contains numerous blocks in BFP format quantized from FP. Assuming the rounding-to-nearest scheme

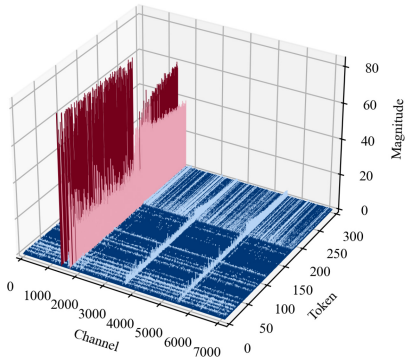

Fig. 2. Data distribution of the activation of one linear layer of the OPT-30B model. The normal distributions are in blue. The lines in red denote the outlier values with extremely large magnitudes.

is used during the quantization procedure, the resulting quantization error has a mean of zero, while the variance is calculated as follows [29]:

$$\sigma^2 = \frac{2^{-2M}}{12} \sum_{i=1}^{N_\gamma} p_{\gamma_i} 2^{2\gamma_i} \tag{6}$$

where $N_\gamma = 2^E$ represents the total number of the possible exponent values, γ_i is the ith exponent value, p_{γ_i} denotes the probability that γ_i is used as the shared exponent for the block, M and E are the bit widths of private mantissa and the shared exponent.

Thus, increasing the bit width of the private mantissa is the most straightforward approach to reduce the quantization error of BFP.

On the other hand, the quantization error is related to the shared exponent selection for each tensor block. The greater the value of the shared exponent γ_i and the higher the corresponding p_{γ_i}, the more significant the impact on quantization error. It also provides a rationale for vanilla BFP's suboptimal performance in LLM quantization, particularly in the presence of outliers. Consequently, how to reduce p_{γ_i} of relatively large γ_i is crucial in optimizing the data format of BFP.

In summary, the preceding analysis of quantization error demonstrates that we can increase the bit width of mantissa or decrease the frequency of large exponent as a shared exponent. Nevertheless, increasing the bit width of the mantissa is antithetical to the fundamental objective of quantization, as minimizing the bit width of the compressed model is always preferable. Therefore, the latter option is the more viable choice. Our proposed Oiso data format can achieve this objective by splitting the elements into two groups, i.e., the outlier group and the normal group, which will be detailed in Section III-C.

B. Sub-Block Alignment Analysis

Another essential problem is determining the granularity of the outlier/normal group. BiE [22] utilizes the finest granularity so that each value in the block can be freely assigned to one group based on the threshold value. In this case, every value should have one extra bit to indicate the type. The quantized format utilizes increased average bit width and

TABLE I
PROPORTION OF OUTLIER AND NORMAL TYPES IN THE EXAMPLE ACTIVATION TENSOR OF OPT-30B [2]

Sub-block Size	Normal	Outlier
1	99.96%	0.04%
2	99.92%	0.08%
4	99.85%	0.15%
8	99.70%	0.30%
16	99.40%	0.60%

introduces considerable hardware overhead to accommodate its excessive flexibility.

Inspired from model pruning [36], [37], [38], [39], [40], some insignificant values can be pruned, which can achieve negligible loss compared to the full precision model. Moreover, the extreme outliers are sparse, and the distribution of the other part is usually flat. Therefore, eliminating the necessity for fine-grained encoding can reduce the memory and architectural design overhead.

A sub-block is defined as a subset of the block comprising N consecutive elements, and there is no overlap between the sub-blocks. Therefore, if the block size is N_B, there will be a total of $N_{\text{SB}} = N_B/N$ sub-blocks in the block. Sub-block alignment assigns all the elements in the sub-block to be the same type. Furthermore, if at least one element within the sub-block is identified as an outlier, the sub-block is identified as an outlier. Otherwise, it is normal. In this manner, only the normal values adjacent to the outliers are relinquished.

We conduct a sub-block alignment analysis using a case study of one activation tensor in OPT-30B model [2]. The block size is 16, and the sub-block size varies from 1 to 16. We use a threshold value of $\mu+3\sigma$, where μ is the mean value and σ is the standard variance of the tensor. If the magnitude of the value exceeds the threshold, the value will be seen as an outlier.

When the sub-block size is 1, which is also the finest-grained type encoding, the extreme outliers only account for 0.04%, as shown in Table I. As the sub-block size increases, more normal values will inevitably be compromised. Nevertheless, this will only have a negligible impact on a tiny proportion (<1%), with most normal values remaining unaffected.

C. Oiso Encoding

Based on the insights gained from the BFP quantization error and sub-block alignment analysis, we detail our proposed Oiso data format with hierarchical block encoding.

Unlike BFP, Oiso can isolate the normal values from the outliers by introducing an extra shared exponent. The two shared exponents are responsible for the normal values and outliers in the block, respectively. Therefore, the top-level block encoding starts by extracting the block's shared exponents, i.e., the maximum exponents of the outlier and normal parts. All the elements in the block share the shared exponents globally.

In the bottom-level block encoding, an identifier that indicates whether the elements belong to the outlier or normal type must be introduced. The bottom-level block encoding is at sub-block granularity to reduce the cost of memory storage and hardware design. 1-bit type bit is adopted for every sub-block as the identifier. The average bit width used per element for encoding in the block can be defined as follows:

$$bw = \frac{2 \times E + N_{\text{SB}}}{N_B} + M + 1 \tag{7}$$

where E is the bit width of the exponent, and $2\times E$ means Oiso has two shared exponent per block. M represents the bit width of mantissa and 1 for the sign bit. N_B is the block size, and N_{SB} is the number of sub-blocks. Because each sub-block shares a type bit, the N_{SB} here denotes the number of shared type bits for the whole block. For instance, we quantize a block in FP16 into Oiso data format, $E = 5$, $N_B = 16$, and $M = 3$ for the quantized mantissa. If the finest-grained encoding is used, the average bit width will be 5.625. However, vanilla BFP only uses 4.3125-bit on average. Accordingly, the utilization of the finest-grained encoding necessitates the allocation of an additional 30% of memory resources. Reducing the granularity of the bottom-level block encoding, e.g., setting N_{SB} to 4, can decrease the memory overhead by 13%.

Algorithm 1 Oiso Encoding With Sub-Block Alignment

1: **Input:** FP block $\boldsymbol{x} = \{x_1, \ldots, x_{N_B}\}$, threshold T, block size N_B, sub-block size N.
2: **Output:** Type bits $\boldsymbol{t} = \{t_1, \ldots, t_{N_B/N}\}$, private mantissa $\boldsymbol{m}_q = \{m_q^1, \ldots, m_q^{N_B}\}$, normal/outlier shared exponent e_n/e_o.
3: Number of sub-blocks $N_{\text{SB}} = N_B/N$;
4: **for** $i = 0 \rightarrow N_{\text{SB}}$ **do**
5: each sub-block $\boldsymbol{x}_i = \{x_{i\cdot N+1}, \ldots, x_{i\cdot N+N}\}$ in $\boldsymbol{x}$;
6: **if** $\exists x \in \boldsymbol{x}_i, |x| > T$ **then**
7: $t_{i+1} \leftarrow 1$;
8: **else**
9: $t_{i+1} \leftarrow 0$;
10: **end if**
11: **end for**
12: $e_n \leftarrow \max\{e_i : x_i \in \boldsymbol{x} \wedge |x_i| \le T\}$;
13: $e_o \leftarrow \max\{e_i : x_i \in \boldsymbol{x} \wedge |x_i| > T\}$;
14: **for** each $x_i \in \boldsymbol{x}$ **do**
15: **if** x_i belongs to an outlier sub-block **then**
16: $m_q^i \leftarrow m_i >> e_o - e_i$;
17: **else**
18: $m_q^i \leftarrow m_i >> e_n - e_i$;
19: **end if**
20: **end for**

The Oiso encoding algorithm is demonstrated in Algorithm 1. $\boldsymbol{x}$ is a block of FP numbers, e_i and m_i are each element's original exponent and mantissa. The threshold value T is a tensor-wise setting. Thus, its hardware overhead is negligible. The sub-block alignment algorithm is from line 5 to line 11: if there exists one element whose magnitude exceeds the threshold value T, the sub-block type will be an outlier. Otherwise, it is normal. The top-level block encoding identifies the max exponents of the two parts as their shared exponents (lines 12 and 13). After that, the mantissa will be shifted accordingly, and the amount of the shift is related to the type of elements. It should be noted that the mantissa includes the implicit bit during the shifting operation, as Fig. 1 shows. Finally, the shifted mantissa will be truncated to get the output mantissa $\boldsymbol{m}_q$ (lines 14–20) based on the quantization configuration of mantissa. The encoding algorithm can be

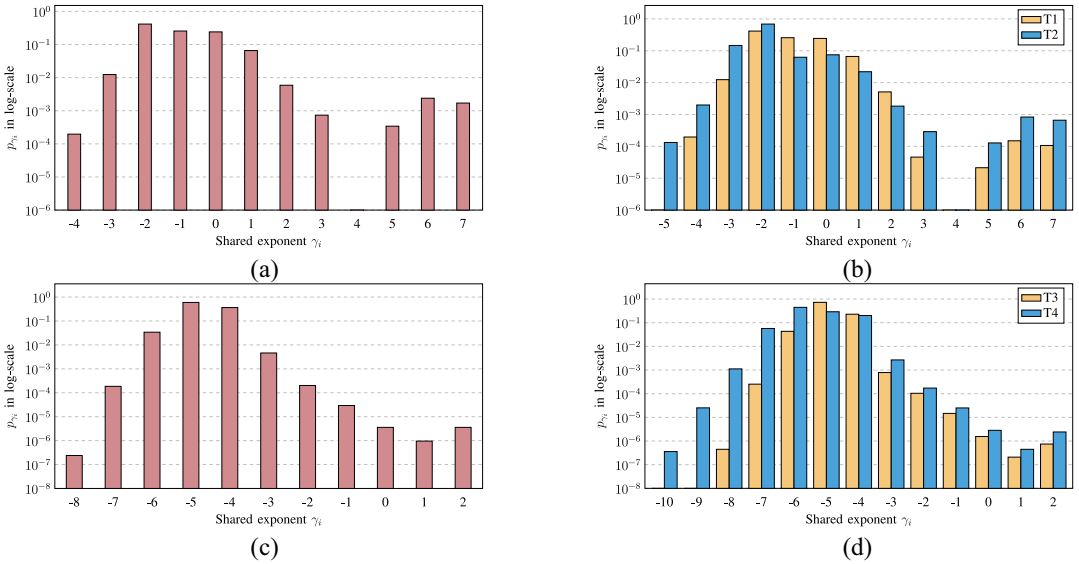

Fig. 3. Distributions of p_{γ_i} of the activation and weight tensor of one Linear Layer in OPT-30B. p_{γ_i} is represented in log-scale for better illustration. (a) (c) shows the original shared exponents' distribution of the activation and weight in BFP. (b) (d) Oiso can decrease the p_{γ_i} of the relatively large γ_i. In particular, $T_1 > T_2$ and $T_3 > T_4$. The results of MSE demonstrate the potential of leveraging the implicit outliers to reduce the quantization error. (a) Distribution of the shared exponents of the activation in BFP format, the MSE is 0.0055. (b) Distribution of the shared exponents of the activation in Oiso using threshold T1 and T2, the MSE is 0.0052 and 0.0051, respectively. (c) Distribution of the shared exponents of the weight in BFP format, and the MSE is 2.5630e-06. (d) Distribution of the shared exponents of weight in Oiso using threshold T3 and T4, the MSE is 2.0266e-06 and 1.6093e-06, respectively.

implemented efficiently in both hardware and software. In particular, the Oiso encoder eliminates the need for FP multiplication compared to integer-based quantization and can be integrated into the activation unit.

D. Quantization Framework

Our Oiso quantization framework targets PTQ for LLMs without any retraining or fine-tuning to achieve rapid deployment of LLM quantization. Oiso quantizes both activations and weights in low-bit format to reduce the memory footprint and computational cost. Compared to vanilla BFP quantization, we provide more flexible quantization parameters, namely the sub-block size and tensor-wise threshold values, to further reduce the quantization error.

The selection of sub-block size will affect not only the quantization accuracy but also the hardware efficiency, which we will be discussed later. And the choice of threshold is crucial at the algorithmic level. It will determine whether the superiority of the Oiso data format can be fully leveraged.

Existing works employ mean squared error (MSE) minimization framework to determine the quantization parameters by gradient descent or searching algorithm [17], [22], [26], [27], [41]. Oiso quantization framework also adopts the MSE minimization algorithm to determine the threshold values for each activation and weight tensor.

The majority of existing studies use a quite large threshold value to split the tensor into two groups. Consequently, the remaining 99% of normal values cannot undergo any optimization. In other words, the potential for leveraging *implicit outliers* is overlooked. Based on the quantization error analysis in Section III-A, the normal-normal sub-block can also be optimized by isolating relatively large values to prevent relatively small values from losing precision.

In Fig. 3, we evaluate the effect of isolating the normal values from outliers on the activation and weight of one Linear Layer in the OPT-30B model [2]. Compared to BFP, Oiso can effectively reduce the impact of outliers on the normal values. As shown in Fig. 3(b) and (d), the p_{γ_i} are reduced compared with Fig. 3(a) and (c). First, we employ two threshold values, T_1 and T_3, for activation and weight Oiso quantization. $T_1 = \mu_1 + 3 \times \sigma_1$ and $T_3 = \mu_2 + 3 \times \sigma_2$, where μ_1 and μ_2 denote the mean values of the activation and the weight, σ_1 and σ_2 represent the standard variance of the two tensors, respectively. Applying T_1 and T_3 isolates the explicit outliers, reducing the quantization error compared to BFP. Nevertheless, this approach is suboptimal because the explicit outliers are sparse, neglecting the vast optimization space of normal values. We further employ T_2 and T_4 as the optimal threshold values for activation and the weight, which are much smaller than T_1 and T_3. By using decreased threshold values, although p_{γ_i} of the large exponent slightly increases (e.g., $\gamma_i = 3 \sim 7$ in the activation and $\gamma_i = -3 \sim 2$ in the weight), p_{γ_i} of the intermediate values, which accounts for the vast majority of value distribution, is significantly reduced (e.g., $\gamma_i = -1 \sim 2$ in the activation and $\gamma_i = -5 \sim -4$ in the weight). The optimized MSE indicates that the exploitation of implicit outliers is an effective way to further improve Oiso's quantization performance.

IV. Oiso Architecture

Based on our proposed Oiso quantization format, we build an efficient hardware accelerator to perform Oiso arithmetic. In this section, we will introduce the architecture design of Oiso accelerator. We will present detailed hardware design of 4-bit Oiso PE and encoder, along with a data-packing strategy for efficiently storing the quantized tensors in Oiso format.

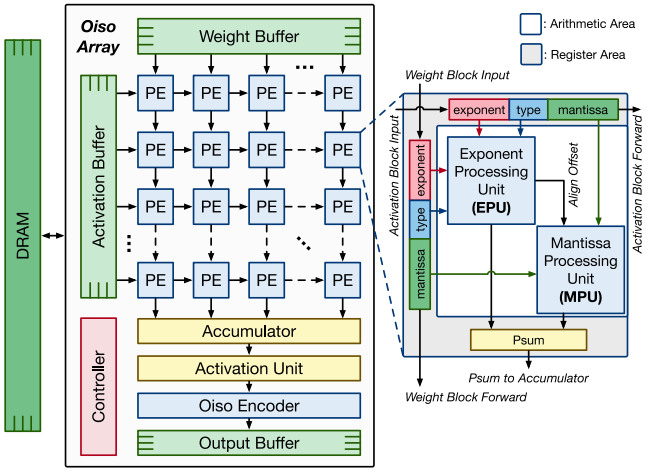

Fig. 4. Architecture overview of Oiso accelerator.

A. Architecture Overview

The Oiso architecture, as illustrated in Fig. 4, involves a 2-D PE array, an activation/weight buffer, an accumulation unit, an output buffer, and a controller. Oiso accelerator adopts systolic array-based dataflow paradigm [13], enabling efficient hardware implementation for Oiso arithmetic, while fully exploiting memory bandwidth. Weight tensors will be first quantized and stored in the DRAM. The quantized tensors with reduced precision will then be loaded and packed into the activation/weight buffer. The PE array supports the Oiso arithmetic efficiently. We allocate a 16-unit-wide multiply-accumulate (MAC) for each PE to handle the activation and weight blocks computation. We employ an output-stationary data flow in which activations and weights flow through the array. The partial sum of each PE will be normalized, aligned, and then accumulated.

B. Oiso PE Design

Oiso PE consists of an exponent processing unit (EPU) and a mantissa processing unit (MPU), as shown in Fig. 5, which is tailored for Oiso arithmetic.

A thorough understanding of Oiso's dot product is necessary before diving into the PE design. The dot product of two blocks $\boldsymbol{x}_a$ and $\boldsymbol{x}_w$ in Oiso format can be formulated as

$$E_{\max} = e_o^a + e_o^w \tag{8}$$

$$e_i^a = e_n^a \cdot \left(1 - t_i^a\right) + e_o^a \cdot t_i^a, i \in [1, N_{\mathrm{SB}}] \tag{9}$$

$$e_i^w = e_n^w \cdot \left(1 - t_i^w\right) + e_o^w \cdot t_i^w, i \in [1, N_{\mathrm{SB}}] \tag{10}$$

$$p_i = \sum_{j=1}^{N} m_{i,j}^a \cdot m_{i,j}^w \tag{11}$$

$$\mathrm{psum} = 2^{E_{\max}} \sum_{i=0}^{N_{\mathrm{SB}}} p_i >> \left(E_{\max} - e_i^a - e_i^w\right) \tag{12}$$

where e_o and e_n are the shared exponents for the outliers and the normal values. t_i represents the type bit of the i-th sub-block. N_{SB} and N denote the number of sub-blocks and the number of elements in a sub-block. $m_{i,j}$ is the j-th signed mantissa in the i-th sub-block.

Compared to the dot product of BFP in (5), the product of the mantissa may correspond to different exponent sums due to the presence of two shared exponents during the dot product of Oiso. Thus, before summing up the mantissa products, a shift operation is required to align all the products to ensure that they will be accumulated correctly. During the alignment process, the maximum exponent sum serves as the target. The products with smaller corresponding exponent will be aligned according to the difference between their exponent and the target exponent. After that, the aligned products will be summed up.

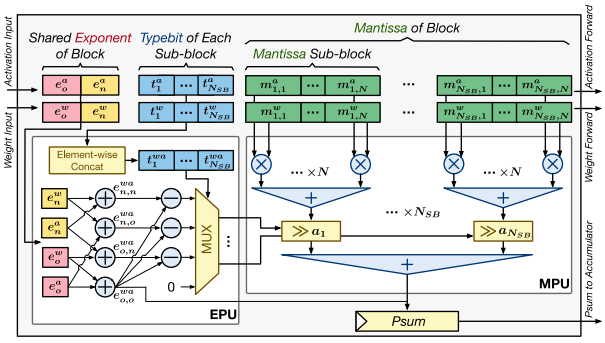

Fig. 5. Oiso PE design including an EPU and a MPU.

EPU is utilized to determine the alignment offset. There are four exponent sums because each block has two shared exponents. Thus, EPU first calculates all the exponent sums using four adders. The corresponding alignment offset will then be obtained by calculating the difference with the maximum exponent sum $E_{\max}$. There is no need to build a comparator tree to determine the maximum exponent sum. Because e_o will always be no less than e_n, and $e_o^a + e_o^w$ will therefore always be no less than the remaining three exponent sums. Besides, if there is no outlier in the block, the Oiso encoder will assign e_o with e_n to ensure e_o is always no less than e_n. The type bits of the sub-block are concatenated to form the selection signal, which determine the sub-block's alignment offset. If the concatenated type bits $t_i^{wa} = 00_2$, the alignment offset will be $E_{\max} - e_n^w - e_n^a$; if $t_i^{wa} = 01_2$, the offset will be $E_{\max} - e_n^w - e_o^a$; if $t_i^{wa} = 10_2$, the offset will be $E_{\max} - e_o^w - e_n^a$; otherwise, the offset is 0.

MPU is responsible for the compute-intensive MAC operations in the Oiso arithmetic. It is equipped with 16 4-bit multipliers to obtain the mantissa products. Following the hierarchical block structure, MPU has two levels of adder trees. The adder tree operates by grouping input operands in pairs and processing them through multiple levels of adders, with each level reducing the number of operands by half. This iterative reduction continues until a single output sum is produced. In the first level, N_{SB} adder trees will accumulate the partial product sum of each sub-block. Then the sub-block partial sums will then be aligned with the shift factors obtained from the EPU. The second-level adder tree sums up the aligned sub-block partial sums to output the final partial sum of the block.

C. Encoder

The Oiso data encoder is designed to efficiently convert FP values into the Oiso format, ensuring correctness and hardware efficiency. Based on the Oiso encoding algorithm

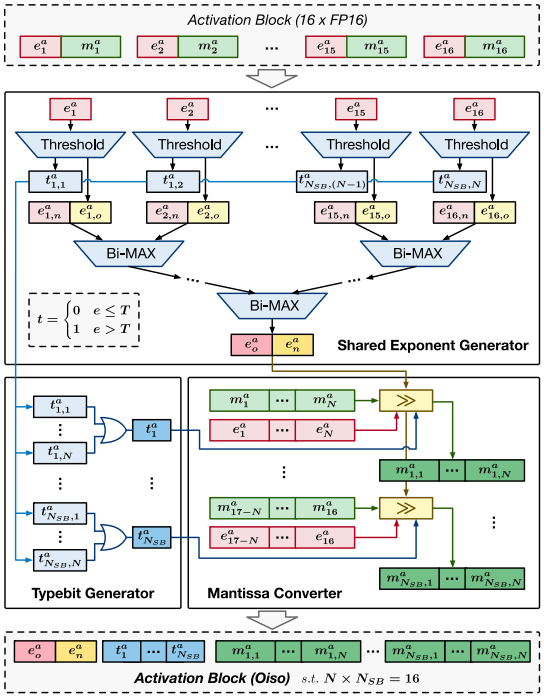

Fig. 6. 4-bit Oiso encoder architecture consists of a shared exponent generator, a typebit generator and a mantissa converter.

described in Algorithm 1, the hardware implementation is shown in Fig. 6. The encoder consists of three key submodules: the shared exponent generator, the typebit generator, and the mantissa converter. These modules work together to process the input FP vector and produce the Oiso-encoded outputs in a hierarchical block encoding structure. The input to the encoder is a vector of 16 FP values, which first pass through the threshold units for element-wise comparison with the tensor-wise threshold T. This comparison determines the type of each element as a normal value or an outlier. The resulting binary type bits are grouped into sub-blocks by the typebit generator, which performs an OR operation across each sub-block to encode a single type bit t_i for the group. This sub-block alignment reduces encoding granularity and simplifies hardware implementation, while maintaining high efficiency. The shared exponent generator extracts the maximum exponents for both normal values (e_n) and outliers (e_o) using a bi-comparator tree, which is a hierarchical structure that efficiently identifies the maximum exponents for normal values (e_n) and outliers (e_o) in parallel, ensuring low-latency extraction for shared exponent generation. These shared exponents are then used by the mantissa converter, which aligns the mantissas of all elements with their respective shared exponents through a series of efficient right-shift operations. The aligned mantissas are truncated to the target bit width (e.g., 4-bit signed mantissa) to complete the encoding.

D. Memory Layout of Oiso Values

We introduce a storage format for Oiso values with 4-bit signed mantissa, which is compatible with existing memory subsystems. Oiso value is composed of three parts: shared exponents, type bits for sub-blocks, and 4-bit signed mantissa for each element. Quantizing FP16 values into Oiso format, the exponent field is 10 bits $(5 + 5)$. The total bits of the signed mantissas are 16×4. Moreover, the bit number used for type depends on the number of sub-blocks N_{SB}. There is a mantissa memory that only stores the 64-bit mantissa block. If $N_{\mathrm{SB}} = 1, 2, 4$, we will pack the type bits with the shared exponents and pad zeros to form a 16-bit number. Otherwise, we will store the exponents and type bits separately.

V. Evaluation

In this section, we evaluate the quantization accuracy using the Oiso quantization framework. The hardware efficiency of Oiso architecture is also evaluated in terms of performance, energy consumption and area overhead.

A. Methodology

1) Quantization Framework: We implement the Oiso quantization framework using PyTorch [42] to simulate its hardware behavior. We evaluate OPT-{6.7B, 13B, 30B} [2], Llama2-{7B, 13B} [3] and Llama3.1-8B [43] which are the representative open-source LLMs with different scales from Huggingface [44].

We compare Oiso with existing quantization works, including SmoothQuant (SQ) [26], Olive [17], BFP [21] and BiE [22]. SQ [26] is the state-of-the-art PTQ method for LLM, which mitigates the outliers by an equivalent scaling between activations and weights. BFP [21] investigates several blocked-based data formats for LLM quantization, demonstrating that BFP is the most promising one to maintain the balance between accuracy and efficiency. BiE [22] further improves the accuracy of BFP by introducing an extra shared exponent for LLM quantization. Olive [17] proposes an outlier-victim pair encoding for LLM quantization, which also supports mixed precision computations to exploit the potential of lower-precision quantization for some layers.

We access the quantized models across a range of natural language processing tasks, including language modeling, multiple choice and common sense reasoning: Wikitext2 [45], LAMBADA [46], PIQA [47], ARC_easy [48], COPA [49] and QNLI [50]. It enables the evaluation of the model's comprehensive capabilities after quantization. The quantization evaluation framework is built based on lm-eval-harness [51].

2) Accelerator Implementation: We implement the Oiso encoder and the PE described in Section IV with Chisel [52] and synthesize them using Synopsys Design Compiler [53] with 7nm FinFET technology library [54] to estimate the area, latency and power of the hardware implementations. We develop a cycle-accurate simulator to evaluate Oiso's end-to-end performance. CACTI [55] is employed to estimate the latency and power of on-chip buffers.

We compare Oiso with AdaptivFloat [56], BiE [22], and Olive [17] in terms of performance and energy consumption. AdaptivFloat [56] introduces an adaptive FP that dynamically determines the tensor-wise exponent bias, and describes the design of the proposed hybrid float-integer PE. BiE [22] utilizes the finest-grained type encoding, and the proposed architecture supports its arithmetic. Olive [17] transforms

TABLE II
QUANTIZATION ACCURACY OF OISO COMPARED WITH BASELINES ON OPT MODELS. THE EVALUATION BENCHMARKS INCLUDE ARC_EASY (ARC_E), COPA, QNLI, PIQA AND LAMBADA

Model	Method	Arc_e	COPA	QNLI	PIQA	LAMBADA
$OPT_{6.7B}$	FP16	65.61	81.00	50.89	76.28	67.69
	SQ8	64.81	80.00	50.67	76.50	68.62
	Olive8	65.19	81.00	50.87	75.68	66.99
	SQ6	59.55	82.00	51.38	71.93	64.16
	BFP4	61.83	82.00	51.67	72.69	58.12
	BiE4	61.87	81.00	53.73	74.43	61.79
	Olive4	41.29	63.00	49.30	60.28	4.39
	Oiso4	62.29	83.00	53.14	73.83	60.37
OPT_{13B}	FP16	67.09	86.00	57.20	75.90	68.72
	SQ8	66.54	84.00	56.14	75.30	68.41
	Olive8	66.67	85.00	56.76	76.22	68.60
	SQ6	54.00	63.00	49.81	68.28	46.19
	BFP4	65.40	84.00	54.18	74.70	67.07
	BiE4	66.84	83.00	56.31	74.97	68.19
	Olive4	38.30	58.00	48.84	57.40	6.31
	Oiso4	65.61	87.00	54.75	74.27	67.94
OPT_{30B}	FP16	69.95	82.00	51.86	77.58	71.40
	SQ8	69.61	84.00	52.52	77.75	71.71
	Olive8	70.16	82.00	52.43	77.37	71.22
	SQ6	41.29	66.00	51.44	57.94	0.54
	BFP4	67.68	81.00	51.13	76.77	70.39
	BiE4	69.32	81.00	52.30	76.77	71.59
	Olive4	33.04	53.00	52.19	55.55	0.56
	Oiso4	68.10	79.00	53.18	76.82	70.64

TABLE III
QUANTIZATION ACCURACY OF OISO COMPARED WITH BASELINES ON LLAMA-2 AND LLAMA-3 MODELS. THE EVALUATION BENCHMARKS INCLUDE ARC_EASY (ARC_E), COPA, QNLI, PIQA AND LAMBADA

Model	Method	Arc_e	COPA	QNLI	PIQA	LAMBADA
$Llama2_{7B}$	FP16	76.35	86.00	49.95	78.07	73.92
	SQ8	73.61	90.00	55.15	75.90	70.83
	Olive8	73.11	90.00	57.86	76.88	70.21
	SQ6	66.58	80.00	52.11	72.52	56.61
	BFP4	70.24	81.00	54.66	74.97	67.65
	BiE4	74.37	84.00	50.41	77.31	72.66
	Olive4	37.33	72.00	50.67	57.56	10.38
	Oiso4	74.12	86.00	53.27	76.77	72.04
$Llama2_{13B}$	FP16	79.55	90.00	49.53	79.11	76.71
	SQ8	77.95	90.00	54.68	77.42	72.89
	Olive8	73.06	86.00	49.46	77.58	72.00
	SQ6	70.79	83.00	51.13	74.48	60.86
	BFP4	74.24	87.00	49.62	77.31	75.37
	BiE4	77.82	88.00	49.64	78.35	75.41
	Olive4	28.45	65.00	50.05	52.94	0.16
	Oiso4	77.61	87.00	49.61	77.91	75.20
$Llama3.1_{8B}$	FP16	81.48	88.00	49.77	80.14	75.84
	SQ8	81.65	89.00	49.73	79.39	74.81
	Olive8	81.73	88.00	50.70	80.09	75.41
	SQ6	66.88	82.00	49.94	74.59	60.86
	BFP4	77.36	85.00	50.69	77.58	69.42
	BiE4	78.54	85.00	50.41	78.35	71.30
	Olive4	30.09	59.00	50.30	55.28	4.52
	Oiso4	76.68	84.00	50.85	78.08	71.01

the encoded values into unified exponent-integer pairs and implements the MAC unit for their computations.

B. Accuracy Results

We evaluate the performance of Oiso on LLM quantization. Since LLM's memory overhead is already considerable, QAT, which requires retraining to recover the quantization error, is not suitable for LLM quantization. We employ the PTQ setting for LLM quantization, which is much more efficient than QAT. We randomly sample 128 data from the Pile dataset [57] for calibration during the PTQ process to determine the quantization parameters.

SQ [26] is the state-of-the-art INT8 PTQ method, and we use its INT8 and INT6 configurations for comparison. We also compare with 8-bit/4-bit Olive [17] for PTQ of LLMs. In addition, BFP4 [21] and BiE4 [22] which have 4-bit signed mantissa (i.e., 1s 3m) are used as baselines. Our proposed Oiso4 has a 4-bit signed mantissa (i.e., 1s 3m), and its sub-block size is 4. We use this quantization configuration to evaluate quantized models on the classification and language modeling tasks. Further analysis of sub-block size and mantissa precision is also conducted.

1) Classification Tasks: Classification tasks include LAMABADA [46], PIQA [47], ARC_easy [48], COPA [49], QNLI [50], which are used to verify the model's capability on tasks such as multiple choice and commonsense reasoning.

As Table II shows, our Oiso4 can achieve negligible accuracy loss on OPT-{6.7B, 13B, 30B} models compared to the original models. SQ and Olive can preserve the accuracy at 8-bit. However, 6-bit SQ can not maintain the accuracy, particularly on OPT-{13B, 30B} models. For 4-bit Olive, the accuracy of the quantized models significantly degraded, especially on Arc_easy and LAMBADA tasks. The average accuracy of Oiso is comparable to that of the BiE and is even higher on OPT-13B, which demonstrates the redundancy of the finest-grained type encoding. We can leverage the sub-block to reduce the hardware overhead. Oiso reduces the impact of outliers on the normal values, enabling it to surpass the BFP. Similar conclusions can also be obtained in evaluating the Llama2-{7B, 13B} models, which demonstrate the generalization of the proposed methodology, as shown in Table III. It can be seen that Oiso4 outperforms Olive8 on Llama2-13B with much smaller bit width.

Language Modeling Task. Language modeling task is critical to evaluate the quality of the sentences generated by LLMs. Perplexity is employed as the evaluation metric for language modeling. It is defined as a measurement of a language model's uncertainty or confusion when attempting to predict the subsequent token in a sequence. A language model's perplexity is derived from the likelihood that it assigns to the actual sequence of tokens in the dataset. Thus, a smaller perplexity means the quality is better.

Table IV shows different quantization methods evaluated on OPT and Llama2 models using WikiText2 [45]. Oiso4 can maintain accuracy with a 4-bit signed mantissa. It is noteworthy that Oiso4 can exceed the capabilities of 8-bit Olive on Llama2-{7B, 13B} and outperform 8-bit SQ on Llama2 models and OPT-13B. Compared to BFP, Oiso demonstrates enhanced resilience in LLM quantization, particularly on Llama-7B. It prevents a notable reduction in the language modeling capacity of LLM after quantization.

2) Sub-Block Size and Precision Effects: Oiso has the flexibility to adjust the sub-block size compared to BFP [20] and BiE [22]. An experiment is conducted to demonstrate the impact of varying sub-block sizes (e.g., 1, 2, 4, 8, 16) on quantization performance. Moreover, we explore Oiso's potential for lower-precision quantization, i.e., Oiso3 with 3-bit signed mantissa (1s 2m).

The average accuracy over the classification tasks and perplexity of WikiText2 on OPT-{6.7B, 13B} with different

TABLE IV
LANGUAGE MODELING PERFORMANCE ON QUANTIZED OPT, LLAMA-2 AND LLAMA-3 MODELS

Method	Llama2-7B	Llama2-13B	Llama3.1-8B	OPT-6.7B	OPT-13B	OPT-30B
FP16	5.31	4.73	6.05	10.64	9.91	9.34
SQ8	6.93	5.94	6.18	11.33	12.79	9.35
Olive8	6.77	5.58	6.23	10.83	10.03	9.42
SQ6	10.38	8.28	9.49	13.16	13.75	82.54
BFP4	7.56	5.14	7.53	11.98	11.03	9.93
BiE4	5.65	5.02	6.83	11.50	10.37	9.46
Olive4	1e6	2e3	135.95	29.55	18.46	58.14
Oiso4	5.87	5.12	7.43	11.79	10.71	9.71

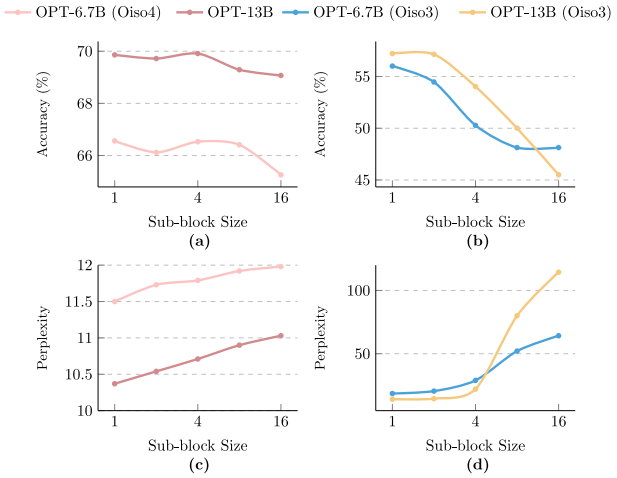

Fig. 7. Average accuracy on Arc_easy, COPA, QNLI, PIQA and LAMBADA and the perplexity on WikiText2 versus sub-block size and mantissa bit width on OPT-6.7B and OPT-13B models.

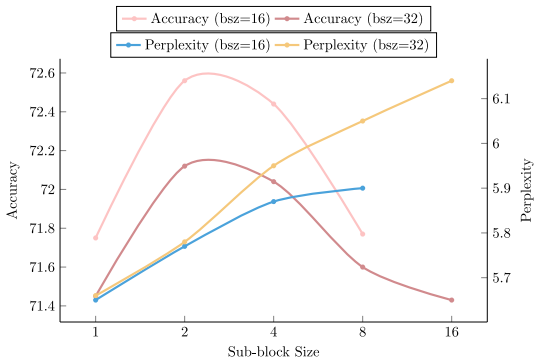

Fig. 8. Average accuracy and perplexity versus sub-block size and block size on LLaMA-2 7B.

sub-block sizes and precision are shown in Fig. 7. The quantization performance generally decreases with an increase in the sub-block size and a decrease in the precision. In classification tasks, Oiso with a 4-bit mantissa and sub-block sizes of 1 and 4 can outperform other sub-block sizes. Moreover, when the sub-block size exceeds 4, the language modeling capability will significantly degrade under the 3-bit configuration.

The former is likely because the larger the sub-block size, the more challenging it becomes to isolate normal values from outliers effectively. The latter can be elucidated through (6). In the case of BFP, the quantization error is observed to decrease as the mantissa bit width increases.

It is noteworthy that the advantages of the proposed Oiso data format become more apparent under the lower-bit setting. This is because, at lower mantissa precision, the normal values are more susceptible to the influence of the outliers, i.e., they are more likely to be quantized to zero. For instance, to avoid a significant loss of precision for the normal values (i.e., directly quantized to 0), the exponent value of outliers within a given block must not exceed the exponent value of a normal value by more than 2. This requirement becomes even more rigorous at a lower precision setting, and the difference in the exponent can not exceed 1.

We further investigate the impact of block size and sub-block size settings on quantization performance, as shown in Fig. 8. In previous experiments, the block size is consistently set to 16. Here, we increase the block size to 32. To highlight the effect of the bi-exponent used, we excluded cases where the block size and sub-block size are equal. Generally, a larger block size tends to result in higher quantization error, as more normal values are influenced by outlier values. On the other hand, a smaller sub-block size enhances the ability to isolate the influence of outlier values.

C. Area, Performance and Energy Analysis

We implement Oiso PE array and encoder and compare with previous works, including Olive [17], BiE [22] and AdaptivFloat [56]. Olive supports mixed-precision computations. Thus, we use its mixed-precision quantization method to obtain an accuracy similar to Oiso for an iso-accuracy comparison. We use the 8-bit AdaptivFloat and BiE4 (i.e., 1s 3m) for comparison. For our proposed Oiso, we employ Oiso4 (i.e., 1s 3m), and its sub-block size is 4 to better balance accuracy and efficiency.

1) Performance: Oiso can achieve better end-to-end latency due to its reduced precision and hardware overhead, as shown in Fig. 9. For OPT-6.7B and OPT-13B, Olive can obtain a similar performance compared to Oiso. However, Olive can only use limited low-precision quantization for the rest of the model, so performance can not be significantly improved. Benefiting from the reduced hardware overhead, Oiso can surpass BiE with more PE units under the same area constraint. Overall, Oiso can achieve 1.26×, 1.54×, and 1.85× faster inference over Olive, BiE, and AdaptivFloat, respectively.

We further conduct a timing analysis on Oiso and BiE architectures. The maximum operating frequency $f_{\max}$ is measured for each design, which is the highest clock frequency at which a digital system can operate reliably without violating

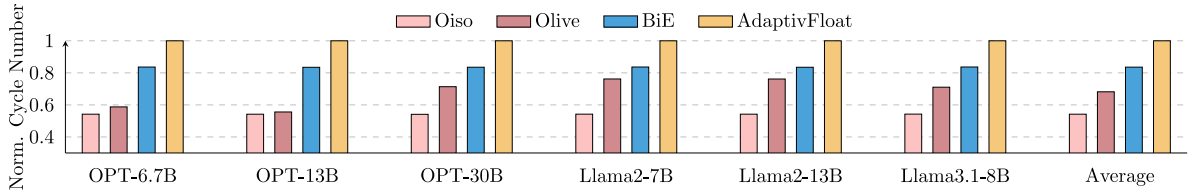

Fig. 9. Comparison of the normalization latency in different accelerators.

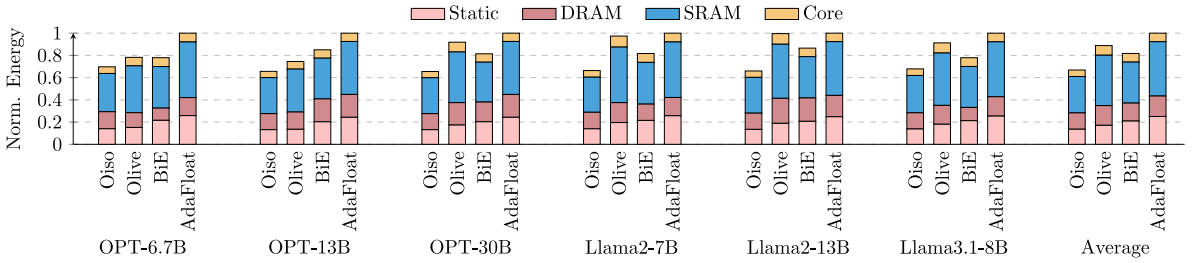

Fig. 10. Comparison of the normalization energy consumption in different accelerators.

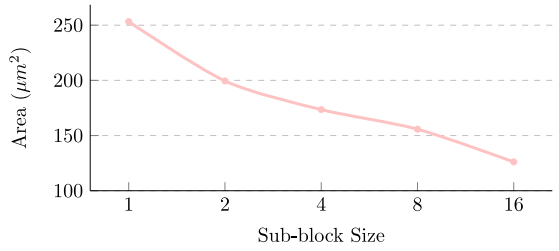

Fig. 11. Oiso4 PE area versus sub-block size.

timing constraints. The results show that for Oiso, $f_{\max} = 699$ MHz, while for BiE, $f_{\max} = 641$ MHz. Due to the reduced hardware complexity, $f_{\max}$ of Oiso achieves a 9% improvement compared to BiE.

2) Energy: Fig. 10 illustrates the breakdown of the energy consumption of each accelerator, including static energy and dynamic energy (DRAM, SRAM, and core). SRAM and DRAM consume more than half of the total energy, which demonstrates the significance of the lower-bit quantization. Oiso can achieve the lowest average bit LLM quantization, which can significantly reduce the energy consumption of DRAM and SRAM. Olive fails to achieve low-bit quantization on OPT-30B, Llama2-7B, and Llama2-13B. Thus, it can not further reduce the energy consumption caused by SRAM and DRAM; its energy consumption is even close to AdaptivFloat. Oiso can reduce the energy by 25%, 18%, and 33% over Olive, BiE, and AdaptivFloat.

3) Area: We first analyze the effect of sub-block size on the PE area. As Fig. 11 illustrates the PE area will decrease with the increased sub-block size. Finest-grained type encoding will lead to significant hardware overhead when dealing with shared exponent selection, alignment and so on. Introducing sub-block alignment avoids those by sharing the same type in the sub-block. When the sub-block size is 4, it can reduce the area overhead by 31%.

The area breakdown of Oiso architecture is shown in Table V. Compared to BFP PE arrays, Oiso requires only additional architectural design for the PEs and encoders; the rest is identical. As for the Oiso PE array, it will occupy about 95% of the total area. However, compared to the on-chip buffer, the area overhead is insignificant. In our previous performance and energy analysis, we size the other accelerators to a similar area as Oiso.

TABLE V
AREA BREAKDOWN OF OISO

Module	Number	Area (mm^2)	Ratio (%)
Oiso4 PE (173.43μm^2)	256	0.0444	94.9%
Encoder (152.92μm^2)	16	0.0024	5.1%

VI. RELATED WORKS AND DISCUSSION

A. LLM Quantization

The emergence of the sparse outliers in activations makes LLM quantization a challenging endeavor when the size of the model exceeds 6.7B [25], [58]. To tackle the outliers, LLM.int8() [25] proposes a mixed-precision quantization scheme that preserves the outliers in FP16 and quantizes the remaining part into INT8. SQ [26], Outlier Suppression+ [27] and OmniQuant [59] mitigate the outliers in activations using an equivalent scaling transformation between activations and weights. GPTQ [60] and AWQ [61] are weight-only quantization frameworks that can reduce the memory footprint of the weights, but the computations are performed in high-precision data format. Binarization represents an extreme form of compression for LLMs, utilizing a single bit to represent quantized value, thereby pushing the boundaries of LLM quantization [62], [63], [64]. To restore the performance of quantized models, several parameter-efficient fine-tuning methods have been proposed [65], [66]. For instance, IR-QLoRA [65] has been introduced to improve the accuracy of quantized LLMs by leveraging LoRA with a focus on information retention, resulting in significant performance enhancements for quantized LLMs.

B. Block Floating-Point

BFP is a popular numerical representation method used in digital signal processing and deep neural network [20], [28], [67], [68]. Researchers introduce different methods to modify the BFP data format to improve the quantization accuracy. AFP [69] proposes an adaptive BFP in which every element has a 3-bit private exponent bias. BiE [22] achieves lossless LLM quantization by introducing an extra shared exponent. OPAL [24] extracts the tensor-wise shared exponent and a block-wise exponent bias. Although they can achieve better accuracy than the vanilla BFP, the hardware overhead and encoding efficiency must be considered better.

Olive [17] proposes an outlier-victim encoding (OVP) which is also inspired by pruning. However, the effect and granularity are different from our proposed sub-block alignment. OVP will only pair up two neighbors. Moreover, it will sacrifice victims to accommodate the outliers, and the victims will become the outlier identifiers. The sub-block alignment can select a different pairing size based on the sub-block size. The normal values in sub-blocks are sometimes sacrificed entirely. It is because we leverage not only explicit outliers but also implicit outliers. The difference between the exponent values of the implicit outliers and the exponents of the normal values is not very large, so in this case, some of the normal values in the sub-block are not completely discarded. In this work, the sub-block size is chosen statically. A dynamic sub-block size is not preferred because it would introduce additional encoding and hardware overhead. Specifically, if different sub-block sizes were used during the inference process, additional control logic would be required to manage this, resulting in increased complexity and resource consumption.

VII. Conclusion

This article introduces a novel Oiso data format for low-bit LLM quantization. With a uniform representation of both outliers and normal values, Oiso circumvents the additional hardware designs to tackle with outliers separately and inefficient memory access. By isolating both explicit outliers and implicit outliers, Oiso quantization can further improve the quantization accuracy. Exploiting the sparsity of explicit outliers, we introduce a hierarchical block encoding method with sub-block alignment to encode the type of the values. We propose an architectural implementation tailored for Oiso arithmetic, realizing efficient low-bit LLM inference. The experimental results demonstrate that Oiso quantization achieves negligible quantization loss for low-bit LLM quantization. The Oiso accelerator outperforms the state-of-the-art outlier-aware accelerator Olive with 1.26× performance improvement and 25% energy reduction.

References

[1] T. B. Brown et al., "Language models are few-shot learners," in *Proc. 34th NeurIPS*, 2020, pp. 1877–1901.

[2] S. Zhang et al., "OPT: Open pre-trained transformer language models," 2022, *arXiv:2205.01068*.

[3] H. Touvron et al., "Llama 2: Open foundation and fine-tuned chat models," 2023, *arXiv:2307.09288*.

[4] A. de Vries, "The growing energy footprint of artificial intelligence," *Joule*, vol. 7, no. 10, pp. 2191–2194, 2023.

[5] B. Jacob et al., "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE CVPR*, 2018, pp. 2704–2713.

[6] Y. Cai, Z. Yao, Z. Dong, A. Gholami, M. W. Mahoney, and K. Keutzer, "ZeroQ: A novel zero shot quantization framework," in *Proc. IEEE/CVF CVPR*, 2020, pp. 13169–13178.

[7] Y. Li et al., "BRECQ: Pushing the limit of post-training quantization by block reconstruction," in *Proc. ICLR*, 2021, pp. 1–16.

[8] X. Wei, R. Gong, Y. Li, X. Liu, and F. Yu, "QDrop: Randomly dropping quantization for extremely low-bit post-training quantization," in *Proc. ICLR*, 2022, pp. 1–19.

[9] Y. Chen et al., "DaDianNao: A machine-learning supercomputer," in *Proc. 47th IEEE/ACM MICRO*, 2014, pp. 609–622.

[10] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan. 2017.

[11] H. Sharma et al., "Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural network," in *Proc. ACM/IEEE 45th ISCA*, 2018, pp. 764–775.

[12] J. Albericio et al., "Bit-pragmatic deep neural network computing," in *Proc. 50th IEEE/ACM MICRO*, 2017, pp. 382–394.

[13] N. P. Jouppi et al., "In-datacenter performance analysis of a tensor processing unit," in *Proc. 44th ISCA*, 2017, pp. 1–12.

[14] 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 *Proc. 53rd IEEE/ACM MICRO*, 2020, pp. 811–824.

[15] E. Park, D. Kim, and S. Yoo, "Energy-efficient neural network accelerator based on outlier-aware low-precision computation," in *Proc. ACM/IEEE 45th ISCA*, 2018, pp. 688–698.

[16] C. Guo et al., "ANT: Exploiting adaptive numerical data type for low-bit deep neural network quantization," in *Proc. 55th IEEE/ACM MICRO*, 2022, pp. 1414–1433.

[17] C. Guo et al., "OliVe: Accelerating large language models via hardware-friendly outlier-victim pair quantization," in *Proc. 50th ISCA*, 2023, pp. 1–15.

[18] S. Jain, S. Venkataramani, V. Srinivasan, J. Choi, K. Gopalakrishnan, and L. Chang, "BiScaled-DNN: Quantizing long-tailed datastructures with two scale factors for deep neural networks," in *Proc. 56th ACM/IEEE DAC*, 2019, pp. 1–6.

[19] Z. Song et al., "DRQ: Dynamic region-based quantization for deep neural network acceleration," in *Proc. ACM/IEEE 47th ISCA*, 2020, pp. 1010–1021.

[20] B. Rouhani et al., "Pushing the limits of narrow precision inferencing at cloud scale with Microsoft floating point," in *Proc. 34th NeurIPS*, 2020, pp. 10271–10281.

[21] C. Zhang, J. Cheng, I. Shumailov, G. Constantinides, and Y. Zhao, "Revisiting block-based quantisation: What is important for sub-8-bit LLM inference?" in *Proc. Conf. Empir. Methods Nat. Lang. Process.*, 2023, pp. 9988–10006.

[22] L. Zou, W. Zhao, S. Yin, C. Bai, Q. Sun, and B. Yu, "BiE: Bi-exponent block floating-point for large language models quantization," in *Proc. 41st ICML*, 2024, pp. 62978–62992.

[23] N. Trukhanov and I. Soloveychik, "Accurate block quantization in LLMs with outliers," 2024, *arXiv:2403.20137*.

[24] J. Koo, D. Park, S. Jung, and J. Kung, "OPAL: Outlier-preserved microscaling quantization accelerator for generative large language models," in *Proc. 61st ACM/IEEE DAC*, 2024, pp. 1–6.

[25] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, "LLM.int8(): 8-bit matrix multiplication for transformers at scale," 2022, *arXiv:2208.07339*.

[26] 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 *Proc. 40th ICML*, 2023, pp. 38087–38099.

[27] X. Wei et al., "Outlier suppression+: Accurate quantization of large language models by equivalent and optimal shifting and scaling," 2023, *arXiv:2304.09145*.

[28] B. D. Rouhani et al., "Microscaling data formats for deep learning," 2023, *arXiv:2310.10537*.

[29] Z. Song, Z. Liu, and D. Wang, "Computation error analysis of block floating point arithmetic oriented convolution neural network accelerator design," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 816–823.

[30] S. Q. Zhang, B. McDanel, and H. Kung, "FAST: DNN training under variable precision block floating point with stochastic rounding," in *Proc. IEEE HPCA*, 2022, pp. 846–860.

[31] Y.-C. Lo and R.-S. Liu, "Bucket getter: A bucket-based processing engine for low-bit block floating point (BFP) DNNs," in *Proc. 56th Annu. IEEE/ACM MICRO*, 2023, pp. 1002–1015.
[32] *d-Matrix Jayhawk II Architecture*, d-Matrix, Santa Clara, CA, USA, 2024.
[33] *Tenstorrent Grayskull Architecture*, Tenstorrent, Toronto, ON, Canada, 2024.
[34] *Tenstorrent Wormhole Architecture*, Tenstorrent, Toronto, ON, Canada, 2024.
[35] *AMD Ryzen AI 300 Series Architecture*, AMD, Santa Clara, CA, USA, 2024.
[36] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2016, *arXiv:1510.00149*.
[37] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1–9.
[38] E. Frantar and D. Alistarh, "SparseGPT: Massive language models can be accurately pruned in one-shot," in *Proc. 40th ICML*, 2023, pp. 10323–10337.
[39] X. Ma, G. Fang, and X. Wang, "LLM-pruner: On the structural pruning of large language models," in *Proc. 37th NeurIPS*, 2023, pp. 21702–21720.
[40] M. Sun, Z. Liu, A. Bair, and J. Z. Kolter, "A simple and effective pruning approach for large language models," in *Proc. ICLR*, 2024, pp. 1–23.
[41] R. Banner, Y. Nahshan, and D. Soudry, "Post training 4-bit quantization of convolutional networks for rapid-deployment," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 7950–7958.
[42] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. 33rd NeurIPS*, 2019, pp. 1–12.
[43] A. Grattafiori et al., "The Llama 3 herd of models," 2024, *arXiv:2407.21783*.
[44] T. Wolf et al., "Transformers: State-of-the-art natural language processing," in *Proc. Conf. Empir. Methods Nat. Lang. Process., Syst. Demonstr.*, 2020, pp. 38–45.
[45] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," 2016, *arXiv:1609.07843*.
[46] D. Paperno et al., "The LAMBADA dataset: Word prediction requiring a broad discourse context," 2016, *arXiv:1606.06031*.
[47] Y. Bisk, R. Zellers, J. Gao, and Y. Choi, "PIQA: Reasoning about Physical Commonsense in Natural Language," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 7432–7439.
[48] P. Clark et al., "Think you have solved question answering? Try arc, the AI2 reasoning challenge," 2018, *arXiv:1803.05457*.
[49] M. Roemmele, C. A. Bejan, and A. S. Gordon, "Choice of plausible alternatives: An evaluation of commonsense causal reasoning," in *Proc. AAAI Spring Symp. Ser.*, 2011, pp. 90–95.
[50] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," 2019, *arXiv:1804.07461*.
[51] L. Gao et al. "A framework for few-shot language model evaluation." 2023. [Online]. Available: https://zenodo.org/records/10256836
[52] J. Bachrach et al., "Chisel: Constructing hardware in a scala embedded language," in *Proc. DAC*, 2012, pp. 1216–1225.
[53] (Synopsys, Sunnyvale, CA, USA). *Synopsys Design Compiler: Concurrent Timing, Area, Power, and Test Optimization*. (2024). [Online]. Available: https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/dc-ultra.html
[54] L. T. Clark et al., "ASAP7: A 7-nm FinFET predictive process design kit," *Microelectron. J.*, vol. 53, pp. 105–115, Jul. 2016.
[55] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi, "Cacti 6.0: A tool to model large caches," *HP Lab.*, vol. 27, p. 28, Apr. 2009.
[56] T. Tambe et al., "Algorithm-hardware co-design of adaptive floating-point encodings for resilient deep learning inference," in *Proc. 57th ACM/IEEE DAC*, 2020, pp. 1–6.
[57] L. Gao et al., "The pile: An 800GB dataset of diverse text for language modeling," 2020, *arXiv:2101.00027*.
[58] Z. Yao, R. Y. Aminabadi, M. Zhang, X. Wu, C. Li, and Y. He, "ZeroQuant: Efficient and affordable post-training quantization for large-scale transformers," in *Proc. 36th NeurIPS*, 2022, pp. 27168–27183.
[59] W. Shao et al., "OmniQuant: Omnidirectionally calibrated quantization for large language models," 2024, *arXiv:2308.13137*.
[60] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, "GPTQ: Accurate post-training quantization for generative pre-trained transformers," 2023, *arXiv:2210.17323*.
[61] J. Lin et al., "AWQ: Activation-aware weight quantization for LLM compression and acceleration," 2024, *arXiv:2306.00978*.
[62] W. Huang et al., "BiLLM: Pushing the limit of post-training quantization for LLMs," in *Proc. 41st ICML*, 2024, pp. 20023–20042.
[63] H. Qin et al., "BiBERT: Accurate fully binarized BERT," in *Proc. ICLR*, 2022, pp. 1–24.
[64] Z. Li et al., "ARB-LLM: Alternating refined binarizations for large language models," in *Proc. ICLR*, 2025, pp. 1–13.
[65] H. Qin et al., "Accurate LoRA-finetuning quantization of LLMs via information retention," in *Proc. 41st ICML*, 2024, pp. 41498–41516.
[66] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient finetuning of quantized LLMs," in *Proc. 37th Adv. Neural Inf. Process. Syst.* 2023, pp. 10088–10115.
[67] A. V. Oppenheim and C. J. Weinstein, "Effects of finite register length in digital filtering and the fast fourier transform," *Proc. IEEE*, vol. 60, no. 8, pp. 957–976, Aug. 1972.
[68] X. Lian, Z. Liu, Z. Song, J. Dai, W. Zhou, and X. Ji, "High-performance FPGA-based CNN accelerator with block-floating-point arithmetic," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 8, pp. 1874–1885, Aug. 2019.
[69] T. Yeh, M. Sterner, Z. Lai, B. Chuang, and A. Ihler, "Be like water: Adaptive floating point for machine learning," in *Proc. 39th ICML*, 2022, pp. 25490–25500.

Lancheng Zou received the B.E. degree in electronic information engineering from Wuhan University, Wuhan, China, in 2023. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.

His research interests include machine learning for electronic design automation, efficient deep neural network, and hardware/software co-design.

Shuo Yin received the B.Eng. degree in computer science and engineering from Beihang University, Beijing, China, in 2022. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.

His research interests include programming language, compiler design for hardware, and large scale GPU parallelization for EDA.

Mingjun Li received the B.E. degree in automation science and electrical engineering from Beihang University, Beijing, China, in 2021. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, supervised by Prof. B. Yu.

His research interests include energy efficient computing, algorithm-architecture co-design, and electronic design automation.

Mingzi Wang received the B.E. degree from the School of Computer Science, Beijing University of Posts and Telecommunications, Beijing, China, in 2022, and the M.S. degree from the Tsinghua Shenzhen International Graduate School, Tsinghua University, Beijing, in 2025. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.

His research interests include machine learning in electronic design automation, efficient AI, and design space exploration.

Chen Bai received the B.E. degree in software engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 2020, and the Ph.D. degree in computer science and engineering from The Chinese University of Hong Kong, Hong Kong, in 2024.

He is currently a Postdoctoral Fellow with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong. His research interests include computer architecture and electronic design automation.

Prof. Bai received the William J. McCalla Best Paper Award from ICCAD 2021 and the Best Paper Award Nomination from ISPD 2024.

Wenqian Zhao received the B.Sc. and Ph.D. degrees in computer science and engineering from The Chinese University of Hong Kong, Hong Kong, in 2019 and 2024, respectively.

His research interests include machine learning for VLSI design automation and hardware-aware deep-learning acceleration.

Bei Yu (Senior Member, IEEE) received the Ph.D. degree from The University of Texas at Austin, Austin, TX, USA, in 2014.

He is currently a Professor with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.

Prof. Yu received 11 Best Paper Awards from ICCAD 2024, 2021, and 2013, IEEE TSM 2022, DATE 2022, ASPDAC 2021 and 2012, ICTAI 2019, Integration, the VLSI Journal in 2018, ISPD 2017, SPIE Advanced Lithography Conference 2016, six ICCAD/ISPD contest awards, IEEE CEDA Ernest S. Kuh Early Career Award in 2021, DAC Under-40 Innovator Award in 2024, and the Hong Kong RGC Research Fellowship Scheme Award in 2024. He has served as the TPC Chair of ACM/IEEE Workshop on Machine Learning for CAD, and in many journal editorial boards and conference committees.