OA-LAMA: An Outlier-Adaptive LLM Inference Accelerator with Memory-Aligned Mixed-Precision Group Quantization

Huangxu Chen*†, Yingbo Hao*‡, Yi Zou‡, Xinyu Chen§†
†The Hong Kong University of Science and Technology (Guangzhou), ‡South China University of Technology
hchen499@connect.hkust-gz.edu.cn, 202410193423@mail.scut.edu.cn, zouyi@scut.edu.cn, xinyuchen@hkust-gz.edu.cn

***Abstract*—Large language models (LLMs) face significant deployment challenges due to their substantial memory and computational demands. While low-precision quantization offers a promising solution, the presence of activation outliers severely degrades model accuracy. Existing approaches either compromise hardware efficiency through misaligned memory access or sacrifice quantization granularity through rigid bit-width allocation, particularly when handling non-uniform tensor distributions across and within layers. This paper presents a hardware-software co-designed framework resulting in an outlier-adaptive LLM inference accelerator with memory-aligned mixed-precision group quantization, named OA-LAMA. The framework comprises three key innovations: First, an outlier-adaptive memory-aligned mixed-precision group (OAMAG) format with a novel outlier reordering technique is proposed to preserve accuracy while maintaining DRAM-aligned memory access. Second, a distribution-aware group allocation strategy is proposed to address inter-layer outlier ratio variance. Finally, we design the OA-LAMA hardware architecture with a three-level accumulation architecture and timing-balanced processing elements to support the OAMAG format efficiently. Evaluations demonstrate that OA-LAMA achieves better accuracy than state-of-the-art 4-bit quantization methods while delivering 1.21–3.09× performance improvement and 1.35–2.47× energy efficiency gains over leading LLM accelerators. OA-LAMA establishes new Pareto frontiers in accuracy-efficiency co-optimization for LLM inference. OA-LAMA is open-sourced at https://github.com/CLab-HKUST-GZ/ICCAD25_OA-LAMA.git.**

***Index Terms*—Large Language Model (LLM), Hardware-Software Co-Design, Accelerator, Mix-Precision Group Quantization.**

I. Introduction

Large language models (LLMs) have demonstrated remarkable performance across a wide range of tasks, including natural language understanding [6] and generation [7], [8]. However, their efficient deployment remains challenging due to substantial memory consumption and computational demands. For instance, DeepSeek-R1 [9], with 671 billion parameters, requires 1,342 GB of memory in FP16 precision, far exceeding the 192 GB memory capacity of the latest NVIDIA B200 GPU [10], let alone other resource-constrained scenarios. Among existing optimization techniques, quantization has emerged as one of the most effective methods for reducing the inference cost of LLMs [11], [12]. Traditionally, neural network quantization can be categorized into two approaches: Post-Training Quantization (PTQ) [1]–[5], [11], [13]–[20] and Quantization-Aware Training (QAT) [11], [21]–[23]. While QAT generally achieves superior performance, retraining LLMs is often impractical in real-world deployment due to their massive scale and prohibitive computational overhead [2]. Consequently, PTQ is more widely adopted in practice.

The inference process of LLMs consists of two phases: the prefill phase and the decoding phase. The prefill phase is primarily compute-bound, while the decoding phase with small batch sizes is mainly memory-bound [24]. Current quantization approaches for LLMs can be categorized into three types: (1) **Weight-only quantization:** It primarily addresses memory-bound General Matrix-Vector Multiply (GEMV) operations in the decoding phase [13], [17], [23]. (2) **Weight-Activation quantization:** It simultaneously addresses both memory-bounded GEMV operations in the decoding phase and compute-bound General Matrix Multiply (GEMM) pressure during the prefill phase [1], [3]–[5], [14], [15], [19], [20]. (3) **Key-Value (KV) cache quantization:** It targets the decoding phase for long-sequence or large-batch scenarios [25], [26], where the memory overhead from KV cache can exceed that of the model weights themselves [27]. The KV cache can be conceptually regarded as a specialized form of activation that persists across sequential decoding steps. Unlike transient activations in conventional forward passes, the KV cache retains computed key-value states from previous tokens to enable efficient autoregressive generation. Among these, weight-activation quantization provides the most comprehensive benefits, as it simultaneously targets memory-bound and compute-bound bottlenecks. In this work, we target weight-activation quantization while also supporting KV cache quantization to achieve comprehensive optimization benefits.

Despite recent progress, achieving accurate and efficient low-bit quantization (e.g., 4-bit quantization) remains challenging due to the presence of activation outliers [12]. These rare but large-magnitude values disproportionately impact scaling factors, causing significant quantization errors for normal values in the same group. While group-wise quantization [3], [13], [17] and mixed-precision techniques [2], [16], [21], [28] have been proposed to address this, they often suffer from precision-efficiency trade-offs, hardware overhead from FP16 scaling, and misalignment with DRAM memory structures.

To understand the design trade-offs in existing work, we summarize key characteristics of representative quantization approaches in Table I. Most notably, methods such as OPAL [1] and Oltron [5] preserve outlier accuracy using high-bit-width formats like BF16 or FP12, but incur substantial memory and hardware costs. Others, like Atom [3] and Tender [4], use fixed bit-width allocations or 1-bit rescaling, leading to either performance bottlenecks or limited quantization granularity. Additionally, DRAM alignment is often overlooked or sacrificed, as seen in all prior methods except Olive [2], despite its known impact on off-chip bandwidth efficiency. None of the existing approaches simultaneously support KV cache quantization, group-wise quantization, DRAM-aligned formats, and zero-overhead scaling—a gap that this work aims to close.

In this paper, we propose OA-LAMA, a hardware-software co-designed accelerator that introduces a novel Outlier-Adaptive Memory-Aligned Group (OAMAG) format. OA-LAMA is designed to preserve outlier precision, ensure memory alignment, and support efficient hardware execution for fine-grained mixed-precision group quantization in LLM inference. Our key contributions are:

- Outlier-Adaptive Memory-Aligned Group (OAMAG) Format: A mixed-precision group format that leverages outlier scattering and exponent-based scaling to preserve outliers while maintaining memory-aligned layout for DRAM efficiency.
- Distribution-Aware Group Allocation Strategy: A lightweight MSE-based optimization that dynamically determines the propor-

*These authors have contributed equally to this work.
§Corresponding author.

TABLE I: Quantization Method Comparison

Method		OPAL [1]	Olive [2]	Atom [3]	Tender [4]	Oltron [5]	**OAMAG (Ours)**
KV Cache Support		no	no	yes	yes	no	**yes**
Group Quantization		yes	no	yes	yes	no	**yes**
Accuracy	Outlier Bit-Width	BF16	Abfloat8	INT8	INT4	FP8/FP12	**INT8**
Efficiency	Scale Type	Exponential	FP16	FP16	1-bit Rescale	FP16	**Exponential**
	Dequantize Pattern	Shifting	Multiplication	Multiplication	Multiplication	Multiplication	**Shifting**
	Outlier/Scale Bit-Width Overhead	10.25%	0%	6.25%	∼1.0%	2.5%	**0%**
Aligned Memory Access	Computational Level	no	yes	yes	yes	yes	**yes**
	DRAM Level	no	yes	no	no	no	**yes**

tion of outlier and normal groups per layer, adapting to inter-layer outlier variation without manual tuning.

- Efficient Accelerator Architecture: A weight-stationary systolic array architecture with a three-level accumulation architecture, timing-balanced processing elements (PEs) and OAMAG encoders/decoders, which efficiently supports OAMAG quantization and enables synchronized high-throughput execution across mixed-precision PEs.

Through extensive evaluation on OPT and LLaMA models, OA-LAMA achieves 1.21–3.09× speedup and 1.35–2.47× energy efficiency improvements over state-of-the-art 4-bit quantization accelerators while consistently outperforming them in accuracy. These results demonstrate OA-LAMA's strength in pushing the frontier of outlier-aware LLM accelerator.

II. Background and Motivation

This section examines research trends in LLM quantization, covering group-wise quantization and mixed-precision quantization. We then identify two major challenges facing current outlier-handling techniques and outlier-aware accelerator designs and propose two solutions to address these challenges.

A. Quantization for LLMs

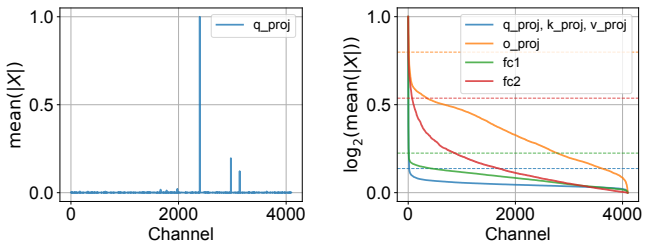
Basic quantization. The quantization process for neural networks can be formally expressed as follows:

$$W' = \lfloor \frac{W}{s} \rceil, s = \frac{\max(|W|)}{2^{n-1}-1}, \tag{1}$$

where W and W' denote the original and quantized data respectively, s is the scaling factor, and n specifies the quantization bit-width. The scaling factor s is primarily determined by the maximum of W. Traditional neural network quantization typically operates at either tensor-level or channel-level granularity [11]. However, when applied to LLMs, the presence of extreme outliers significantly impacts the determination of s at these coarse granularities, leading to substantial quantization errors [14], [18].

Group quantization. To address this limitation, state-of-the-art (SOTA) approaches adopt group-wise quantization schemes, where a group of contiguous elements share a common scaling factor s [3], [13], [17]. This finer-grained quantization approach effectively localizes the impact of outliers to smaller regions, thereby improving post-quantization model accuracy. However, the FP16 per-group scale introduces additional dequantization overhead. Recently, the Microscaling (MX) data format has emerged as an alternative fine-grained quantization scheme [29], [30]. When employing the MX format, the quantization formula can be reformulated as:

$$W'_g = \lfloor W_g \gg s_g \rceil, s_g = \lfloor \log_2(\frac{\max(|W_g|)}{2^{n-1}-1}) \rfloor, \tag{2}$$



(a) Per-channel absolute mean values in the q_proj layer activations. (b) Log_2-scaled absolute mean values after reordering across layers.

Fig. 1: Outlier distribution within and across layers of OPT-6.7B.

where W_g, W'_g and s_g represent the group-wise original data, quantized data, and scaling factor, respectively. This formula replaces the conventional multiply-divide operations with more hardware-efficient bit-shift operations, offering significant improvements in performance and resource utilization. However, despite the inherent precision limitation of exponent-based scaling, there remains a notable research gap in developing efficient architectural support for such novel formats in modern computing systems.

Mixed-precision quantization. Another intuitive approach involves mixed-precision quantization, where outliers are stored with higher precision to reduce quantization errors. The early works primarily focus on improving the precision of outliers at the element-wise level [2], [16], [21], [28]. For instance, a coordinate list [16], [28] can be employed to separate outliers from normal values. However, such method leads to a large hardware overhead and low performance benefits. Alternatively, recent researches have revealed that outliers in LLMs tend to concentrate in specific channels [14], [18]. State-of-the-art works tend to perform higher-precision quantization for these specific channels at the channel-wise level, thus reducing hardware overhead with better performance gain [3]–[5], [13], [14], [18].

B. Challenges of Outlier-aware Quantization

However, current outlier-aware research still faces the following two challenges, which make the implementation of mixed-precision quantization and group quantization difficult.

Challenge 1: Non-uniform outlier distribution across and within layers. Outlier distribution in LLMs is highly skewed both across and within layers. As illustrated in Fig. 1(a), certain channels in the q_proj layer of OPT-6.7B [31] exhibit significantly higher activation magnitudes, even after normalization. This intra-layer imbalance implies that a small subset of channels contains most of the quantization-critical outliers. Fig. 1(b) shows that outlier-rich channels vary significantly across layers. Assuming the 3σ criterion is used to identify outlier channels, the portions above the dashed lines

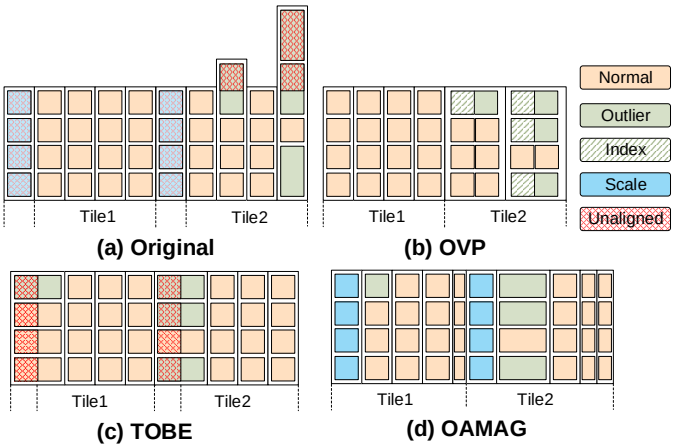


Fig. 2: Comparison between different methods to achieve aligned memory.

represent the outlier channels detected in each layer. For example, in `q_proj`, `k_proj`, and `fc1`, the number and magnitude of outlier channels differ from those in `o_proj`.

Existing static allocation schemes [1], [3], [5] fail to adapt to this variability. Allocating a fixed number of outlier channels across all layers leads to under-provisioning in layers with higher outlier densities, resulting in quantization error and accuracy degradation. Conversely, over-provisioning in layers with fewer outliers wastes precision budgets and reduces overall hardware efficiency. Channel reordering techniques [3]–[5] have been proposed to group outlier channels for improved precision allocation, yet these methods typically assume layer-invariant reordering and are not designed to integrate effectively with fine-grained group quantization. As a result, the combined challenge of intra-layer imbalance and inter-layer variance remains unresolved in existing designs.

Challenge 2: Efficient hardware support for fine-grained group quantization with memory alignment. While reducing group size improves quantization granularity and mitigates outlier interference, it introduces significant overhead in dequantization operations. Increasing the number of groups leads to more scaling factors, complicating control logic and hardware scheduling. Current hardware implementations [1], [3], [4] typically support coarse-grained group sizes (e.g., 128/256/1024) instead of fine-grained ones (e.g., 8/16/32). However, this creates a significant barrier for efficient hardware support of finer-grained group quantization.

Another major issue lies in memory alignment. Storing outliers at higher bit-widths typically leads to unaligned memory layouts, disrupting DRAM bandwidth and introducing irregular memory access patterns. To illustrate this, Fig. 2 compares four existing methods: Conventional implementation preserves full outlier values but results in misaligned memory tiles (Fig. 2(a)). OVP (Outlier-Victim Pair) [2] sacrifices adjacent normal values to encode outliers, thus reducing accuracy (Fig. 2(b)). TOBE [5] achieves tile-level outlier balancing but loses compatibility with off-chip memory access pattern (Fig. 2(c)), and introduces latency bottlenecks due to timing divergence between high-precision and low-precision heterogeneous processing elements (PEs). OA-LAMA, proposed in this paper, maintains both DRAM-level memory alignment and outlier precision while enabling efficient PE scheduling (Fig. 2(d)). Overall, the core difficulty lies in balancing precision allocation, memory alignment, and hardware compatibility, particularly when integrating outlier handling into low-bit quantization with high throughput demands.

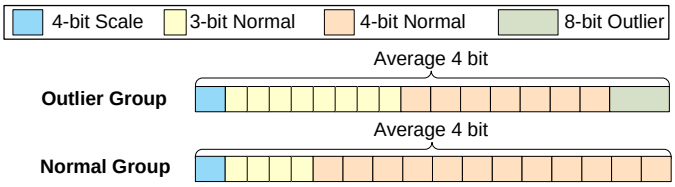


Fig. 3: Data structure in OAMAG format

C. Motivation for OA-LAMA

The analysis above reveals a fundamental trade-off space between accuracy, hardware efficiency, and scalability in outlier-aware quantization. Effective handling of non-uniform outlier distributions requires dynamic precision allocation, while hardware efficiency demands strict memory alignment and synchronized mixed-precision compute.

Solution 1: OAMAG format with outlier channel reordering. To address the imbalanced outlier/normal ratio across different layers, we introduce an outlier-adaptive memory-aligned mixed-precision group (OAMAG) format, which includes two group formats. To organically combine the outlier channel reordering technique with these formats, we propose a group-wise outlier channel reordering approach in this work, which scatters one outlier channel per group to minimize quantization error. Our reordering method utilizes a recurring reordering technique that compensates for the additional bitwidth overhead from supporting outlier channels by reducing the bitwidth of less significant trailing group values. We also employ a distribution-aware group allocation method to dynamically assign appropriate proportions of these two group formats to each layer based on its specific distribution.

Solution 2: Efficient systolic array unit with three-level accumulate architecture and timing-balanced PEs. To achieve memory alignment while maintaining computational efficiency and model accuracy, OA-LAMA introduces a three-level accumulation architecture with timing-balanced PEs to support our proposed memory-aligned format. The three-level accumulation architecture efficiently implements fine-grained group dequantization by incorporating tile accumulators within the systolic array. For the timing-balanced PEs, we fuse low-precision PEs to fully utilize their timing slack, thereby improving overall resource utilization. Through these techniques with an OAMAG encoder/decoder design, OA-LAMA effectively enhances hardware computation efficiency under fine-grained quantization while maintaining memory-aligned access patterns and optimizing computational resource usage.

III. Outlier-Adaptive Mixed-Precision Quantization

This section presents our outlier-adaptive mixed-precision group quantization method, which combines a novel reordering method and OAMAG format with an adaptive group allocation strategy.

A. Outlier-Adaptive Memory-Aligned Group (OAMAG) Format

To address the inter-layer imbalance in outlier distribution, we propose a hybrid group quantization scheme that combines two distinct formats - outlier groups and normal groups - to effectively handle layer-wise heterogeneity while maintaining hardware efficiency. The design of these formats carefully considers both computational flexibility and memory alignment requirements, achieved through strategic bit-width allocation to accommodate both outlier values and exponential scaling factors.

As illustrated in Fig. 3, the OAMAG format employs optimized data structures for each group type. Building upon prior work demonstrating the effectiveness of 4-bit exponential scaling [1], our

TABLE II: Comparison of Different Group Sizes in OAMAG Format

Wiki2 PPL(↓)	Group Size			
	8	16	32	64
Llama-7B	6.53	6.47	6.67	6.96
OPT-6.7B	11.95	11.69	12.79	14.90

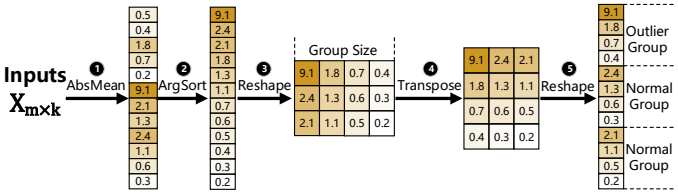


Fig. 4: Group-wise Outlier Scattering Method

implementation carefully balances precision and hardware regularity. For outlier groups, we allocate 8-bit representation for outlier values while quantizing 8 non-significant normal values to 3-bit to maintain an average 4-bit bandwidth. Normal groups employ a similar strategy, using 3-bit quantization for 4 non-significant normal values to compensate for scaling factor overhead while keeping the majority of data in 4-bit format.

The selection of group size presents a fundamental design trade-off distinct from conventional uniform-bitwidth group quantization. Larger group sizes reduce the number of outlier groups, potentially leaving significant outliers unprotected, while smaller sizes increase the proportion of 3-bit values, both of which can adversely affect model accuracy. Through comprehensive evaluation across different models, as shown in Table II, we identify 16 as the optimal group size. Notably, we deliberately avoid 2-bit quantization for compensation due to its demonstrated negative impact on model accuracy, a finding supported by both prior research [32] and our experiments.

B. Group-wise Outlier Scattering Method

Leveraging the channel-wise clustering property of outliers established in Section II-A, we propose a novel group-wise outlier scattering method for the OAMAG format, illustrated in Fig. 4. This technique strategically redistributes outliers through channel reordering, concentrating them at the head of each group while relegating the non-significant channels to the tail positions, which is consistent with our OAMAG format to compress non-significant normal values bit-width.

The proposed channel reordering methodology involves five key computational stages: ❶ Absolute mean values are computed along the channel dimension of the input activations. ❷ Channel indices are sorted based on these computed mean magnitudes. ❸ The sorted indices are then reshaped into a two-dimensional matrix according to the predetermined group size specification. ❹ Through matrix transposition, this operation systematically redistributes outlier channels to leading positions within each group while simultaneously allocating the least significant channels to terminal positions. ❺ The restructured matrix is flattened to generate the final reordering indices that optimize both outlier protection and computational efficiency.

To minimize the hardware overhead of channel reordering, we employ a static operator fusion approach that eliminates explicit reordering operations during runtime. For weight tensors, the reordering can be efficiently performed offline during model compilation. For activations, we strategically fuse the reordering operation with preceding computational layers. Specifically, following the standard transformer block architecture [33], we integrate the reordering of query, key and value projection and first fully connection layers' activations into the preceding layer normalization operations. Similarly, the reordering for output projection and second fully connection layers' activation is incorporated into the weight output channels of value projection and first fully connection layers, respectively. This optimization approach has been demonstrated in prior work [3], [5] to introduce negligible computational overhead while maintaining full functional equivalence with explicit reordering implementations. This is achieved by simply replacing explicit data movement with lightweight address remapping logic that incurs less than 0.1% area overhead. The fusion strategy effectively eliminates the need for dedicated reordering hardware while preserving the benefits of our group-wise outlier scattering method.

C. Distribution-Aware Group Allocation Strategy

To address the inter-layer variation in outlier proportions, we propose a distribution-aware group allocation strategy to dynamically assign appropriate proportions of two OAMAG formats to each layer based on its specific distribution. As illustrated by the OAMAG format data structure in Fig. 3, the optimal allocation should balance between protecting sufficient outliers and minimizing the noise introduced by excessive 3-bit quantization. Our approach systematically evaluates potential threshold selections through mean squared error (MSE) minimization, formulated as:

$$arg \min_{threshold} \sum (Q(W)Q(X) - WX)^2, \tag{3}$$

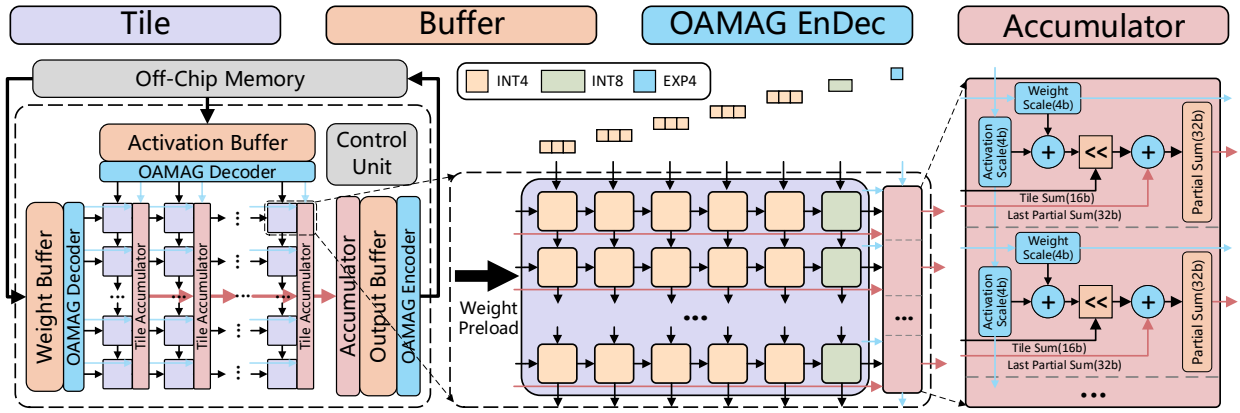
where Q represents the OAMAG quantization function, and W denotes the layer's weights, and X corresponds to the activations from the calibration set. This formulation simultaneously captures the statistical distribution of outliers and their impact on quantization error while ensuring optimization occurs on representative input patterns. When integrated with our channel reordering technique (Fig. 4), the approach achieves effective outlier preservation with controlled noise introduction through several key mechanisms: automatic adaptation to layer-wise distributions without manual tuning, optimal precision allocation balancing, and seamless OAMAG format compatibility. The static threshold determination occurs during model compilation using calibration data, eliminating runtime overhead while ensuring layer-specific optimization based on actual activation patterns and outlier distributions. This represents a significant advance over fixed allocation schemes by providing both theoretical optimality guarantees and practical implementation efficiency.

IV. Hardware Architecture of OA-LAMA

This section elaborates on the architectural details of the OA-LAMA. Our design efficiently supports matrix operations with OAMAG formats and can be integrated with modern hardware accelerators such as Google's TPU [34].

A. Overview of OA-LAMA

Fig. 5(a) illustrates the OA-LAMA architecture that efficiently supports the OAMAG format, including a weight-stationary systolic array with tile accumulators, three on-chip buffers for activation/weight/output, OAMAG EnDec (Encoders/Decoders) for data conversion between the original format and the OAMAG format, and an accumulator supporting the weight-stationary systolic array design. The dataflow follows a four-stage pipeline similar to Google's TPU [34]: **(1) Preloading On-Chip Buffers:** OAMAG-formatted data is loaded into activation/weight buffers. **(2) Weight Loading:** Weights are decoded through OAMAG decoders to obtain INT3/4/8 values with a shared scale, then distributed across systolic array tiles and tile accumulators. **(3) Matrix Computation:** Decoded activations flow through the systolic array and compute with weights.



(a) OA-LAMA (b) Tile (c) Tile Accumulators

Fig. 5: Hardware Architecture of (a) OA-LAMA, (b) Tile, and (c) Tile Accumulator.

A three-level accumulation architecture enables hardware-friendly dequantization, as detailed in Section IV-B. **(4) Output Writeback:** Final outputs are encoded into OAMAG format through OAMAG encoders and stored to off-chip memory.

B. Three-Level Accumulation Architecture

Traditional tensor/channel-wise quantization architectures [5], [34] face inefficiencies when handling fine-grained group quantization due to per-group scaling factors. To address this challenge, we propose a three-level hierarchical accumulation architecture formalized as:

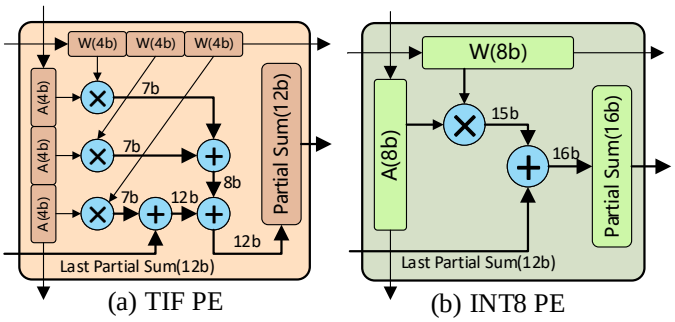
$$\begin{aligned} y &= \sum^{s}(\sum^{l}(\sum^{g}(2^{e_t^w}2^{e_g^w}w \times 2^{e_t^x}2^{e_g^x}x))) \\ &= 2^{e_t^w}2^{e_t^x}\sum^{s}(\sum^{l}(2^{e_g^w}2^{e_g^x}\sum^{g}(w \times x))) \\ &= \sum^{s}(\sum^{l}(\sum^{g}(w \times x) \ll (e_g^w + e_g^x))) \ll (e_t^w + e_t^x), \end{aligned} \quad (4)$$

where $e_t^{w/x}$ denote tensor-level exponential scale for weight and activation respectively, both of which are obtained offline through calibration process, and $e_g^{w/x}$ represent group-wise exponential scale. The hardware dimensions comprise input channel slice number s determined by systolic array capacity, systolic array spatial length l, and group size g equivalent to tile spatial length. This dimensional decomposition enforces the critical constraint $C_{in} = s \times l \times g$ for input channels, ensuring aligned mapping of group-quantized tensors to the systolic array's processing elements through three-level tiling.

First-Level Accumulation. As shown in Fig. 5(b), this stage performs intra-group summation through the systolic array within each tile. Following the OAMAG format defined in Section III-B with group size $g = 16$, quantized data undergoes multiply-accumulate (MAC) operations before being transferred to the tile accumulator. As shown in Fig. 5(c), within the tile accumulator, the pregroup partial sums are dequantized by applying bit-shifting operations based on shared scale $(e_g^w + e_g^x)$.

Second-Level Accumulation. Illustrated in Fig. 5(a), this phase propagates dequantized partial sums along the systolic array's spatial dimension. Each tile's results flow through adjacent tile accumulators column-wise, progressively combining outputs across the array's length l, ultimately feeding the unified sum to the accumulator.

Third-Level Accumulation. To address the limited physical dimensions of the systolic array, intermediate partial sums are temporarily stored in the output buffer after processing each token batch. The architecture subsequently reloads subsequent weight slices into the array for new partial sum computation. During slice-wise iterative processing, the output buffer simultaneously retrieves historical partial sums, performs accumulation with current results through the accumulator, and writes back updated partial sums. This cyclic fetch-merge-update mechanism enables full-channel computation flow.



(a) TIF PE (b) INT8 PE

Fig. 6: Microarchitecture of (a) TIF PE and (b) INT8 PE.

C. Timing Balanced PE Design

Conventional mixed-precision architectures such as Oltron [5] suffer from frequency limitations due to divergent timing delays between high/low-precision processing elements (PEs), where the maximum operating frequency is constrained by the critical path of high-precision PEs. To fully exploit the timing slack of low-precision PEs, we propose a timing-balanced PE design. As illustrated in Fig. 5(b) and aligned with the OAMAG format ($g = 16$), each tile row contains one INT8 PE and 15 INT4 PEs. We fuse three INT4 PEs into a unified computational unit, termed the Triple-INT4 Fused (TIF) PE, as shown in Fig. 6(a). Each TIF PE processes three parallel INT4 dot-product pairs through a two-stage adder tree to generate partial sums. INT8 PE keeps the original design, as shown in Fig. 6(b).

For the TIF PE, the maximum partial sum value is bounded by $15 \times 2^{3+3} = 960 < 2^{11} = 2048$, making signed 12-bit partial sum registers sufficient to prevent overflow. For the INT8 PE, as discussed in the previous, the 4-bit left-shifting leads to a maximum partial sum value of $960 \times 2^4 + 2^{14} = 31744 < 2^{15} = 32768$. Consequently, the signed 16-bit partial sum register shown in Fig. 6(b) ensures overflow-free computation while preserving numerical resolution. This design reduces the tile output latency from 16 cycles to 6 cycles. Leveraging the single-cycle delay between tile accumulators in our three-level accumulation architecture, a 256×256 systolic array $(l = 256, g = 16)$ achieves a total output latency of $(6 + \frac{l}{g}) = 22$ cycles, significantly lower than the original $l = 256$ cycle baseline.

This architecture achieves hardware efficiency through fusing low-precision PEs, enabling synchronous operation of heterogeneous PEs at near-maximal frequency.

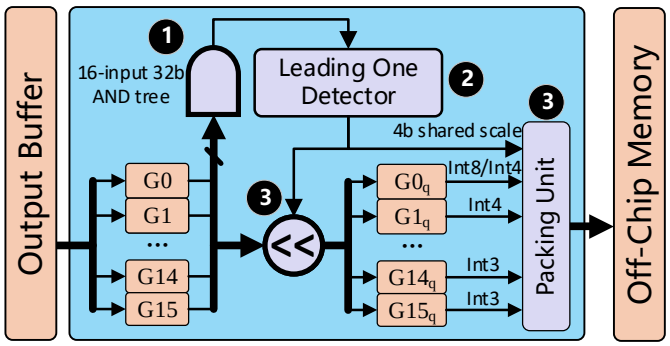


Fig. 7: OAMAG Encoder Design

D. Hardware-friendly OAMAG Decoding/Encoding Scheme

The memory-aligned nature of the OAMAG format enables an exceptionally streamlined decoder implementation. The decoding logic reduces to simple bit-field extraction and concatenation operations, requiring only basic shift-and-mask circuitry that enables efficient decoding that directly matches tile inputs (15 × 4-bit + 1 × 8-bit) and adds negligible area overhead (0.1% of total PE array area, as shown in Table V).

For output processing, the accelerator employs a more sophisticated encoder to compress systolic array results into OAMAG format. As detailed in Fig. 7, this hardware-optimized implementation refines the quantization principles defined in Eq. 2. While the original formulation requires group-wise maximum magnitude detection, our implementation employs hardware-optimized adaptations for practical deployment, as illustrated in Fig. 7. The encoding process comprises three tightly coupled stages: ❶ **Bitwise AND-Reduction:** A parallel AND-reduction tree computes shared most-significant-bit (MSB) positions across all group elements, effectively approximating the maximum magnitude identification required by Eq. 2 without explicit magnitude comparisons. ❷ **Exponential Scale Derivation:** The reduced bit pattern is processed by a single-cycle Leading One Detector (LOD) to generate the group scaling exponent e_g, aligning with the exponential quantization scheme while eliminating complex operations. ❸ **Requantization and Packing:** Raw int32 outputs undergo right-shifting by e_g bits followed by precision truncation to the OAMAG specifications. The packing unit packs these values with their shared scale into memory-aligned 64-bit blocks.

The fully pipelined OAMAG encoder design achieves sustained throughput through bitwise AND reduction, LOD, and parallelized shifters, significantly reducing hardware overhead and maintaining single-cycle group processing latency.

V. Experiment

A. Methodology

Quantization Setup. To evaluate our proposed quantization framework, we implement experiments on PyTorch. Our experiments comprehensively cover model families, including OPT (6.7B-13B) [31], LLaMA (7B-65B) [35], and LLaMA-2 (7B-13B) [36]. We focus on 4-bit weight-activation quantization, which currently represents the accuracy loss limit for practical weight-activation quantization scenarios. For accuracy evaluation, we adopt two standard metrics: perplexity (lower is better) and zero-shot accuracy (higher is better). Perplexity is measured on WikiText2 [37] datasets, while zero-shot evaluation is conducted using lm-eval-harness [38] across multiple tasks, including PIQA, ARC, BoolQ, HellaSwag, and WinoGrande.

Quantization Baselines. We compare OA-LAMA against the leading 4-bit weight-activation quantization methods, including OliVe [2], Tender [4], Oltron [5], OPAL [1], and Atom [3]. In terms of weight quantization, OAMAG format employs the same quantization techniques as activation. To demonstrate the superiority of OAMAG format and outlier handling approach, OA-LAMA does not adopt any other advanced weight quantization error reduction techniques such as GPTQ [17], AWQ [13], or OWQ [23]. Oltron and Atom utilize GPTQ for weight quantization, while OPAL employs OWQ [23] for weight quantization. OWQ achieves lossless accuracy performance that is even better than GPTQ on weight-only quantization. For outlier channel selection, we employ 128 randomly sampled sentences from WikiText2 [37] as calibration data, consistent with Oltron, OPAL, and Atom. Regarding the reordering approach, we adopt the method similar to Atom and Oltron by fusing the reordering operation into the previous layer through kernel fusion. In contrast, Tender stores channel indices in an index buffer due to its row-chunking technique.

Quantization Configuration. Regarding the quantization bit-width calculation, since different works adopt varying computation baselines (particularly when incorporating outlier-preserving schemes and group-wise quantization), we propose a unified bit-width measuring metric defined as:

$$bit_{avg} = \frac{bit_{normal} + bit_{outlier} + bit_{scale}}{N_{\text{group}}}, \tag{5}$$

where bit_{normal} ,$bit_{outlier}$, and bit_{scale} respectively denote the total bitwidth allocated for normal values, outliers, and scale factors per group, with N_{group} representing the number of elements per group. Notably, Atom employs FP16 scales with 128-element grouping while maintaining INT8 quantization for outlier channels (e.g., 128 out of 4096), yielding $bit_{avg} = 4.25$. OPAL utilizes 4-bit scales with 128-element grouping, preserving BF16 precision for 0.25% of weight channels and every 4 out of 128 activation grouping elements, resulting in $bit_{avg} = W4.06A4.41/7.31$. OPAL also employs an inter-layer mixed-precision quantization scheme (A4/7), where layers following layer normalization (LN) in both Attention and Feed-Forward Network (FFN) components are quantized to 4 bits, while other layers in them are quantized to 7 bits. Oltron does not employ group-wise quantization. In the original implementation, two quantization results with bit_{avg} of 4.01 and 4.1 are reported based on different outlier-normal value ratios. Additionally, OAMAG format maintains bit_{avg} of 4 through its innovative scale and outlier extra bit-width fusing technique. For a fair comparison, OAMAG format provides two distinct results demonstrating without fusing: ❶ without fusing of scale ($bit_{avg} = 4.25$) and ❷ all groups utilizing outlier preservation without fusing of both ($bit_{avg} = 4.5$), as will be presented in Table III. For Oltron and OPAL, we report only the results documented in their publications as their code is not available.

OA-LAMA Implementation. The described OA-LAMA architecture, comprising the systolic array and EnDec modules in Section IV, is implemented in Verilog RTL. The design is synthesized using Synopsys Design Compiler [39] under TSMC 28nm technology node at 1 GHz operating frequency to evaluate performance, power, and area (PPA) metrics. On-chip buffer characteristics are analyzed with CACTI [40] under identical process conditions. A cycle-accurate simulator derived from DNNWeaver [41] is employed for architectural performance and energy efficiency evaluation. To ensure equitable comparison, competing baseline designs are scaled to the same technology node using DeepScaleTool [42].

Accelerator Baselines. The OA-LAMA is benchmarked against four mixed-precision accelerators: OPAL [1], Oltorn [5], OliVe [2], and Tender [4]. Each baseline's architecture, including decoders, computation units, and encoders, is synthesized under equivalent die area constraints to ensure iso-area configurations. All designs adopt

TABLE III: Perplexity Results of Quantized Models on WikiText2 Dataset

Method	Bit-Width W	A	KV	OPT-6.7B	OPT-13B	Llama-7B	Llama-13B	Llama-30B	Llama-65B	Llama-2-7B	Llama-2-13B
FP16	16	16	16	10.86	10.12	5.68	5.09	4.01	3.52	5.47	4.88
OliVe	4	4	16	39.18	65.42	32.15	15.64	13.59	12.85	44.07	50.28
Tender	4	4	4	13.56	16.43	23.85	13.68	12.07	8.85	36.47	55.08
Oltron	4	4.01	16	12.69	11.49	36.47	144.08	439.25	15.85	-	-
Oltron	4	4.1	16	12.00	11.35	11.67	8.02	6.68	5.82	-	-
OAMAG	4	4	4	**11.69**	**10.99**	**6.47**	**5.60**	**4.74**	**4.12**	**6.27**	**5.40**
Atom	4.25	4.25	4.25	**11.23**	**10.43**	**6.16**	5.46	4.54	3.89	6.03	5.27
OAMAG*	4.25	4.25	4.25	11.28	10.51	6.18	**5.37**	**4.46**	**3.87**	**5.93**	**5.18**
OPAL	4.06	4.41/7.31	16	**10.98**	10.31	-	-	-	-	6.49	5.30
OAMAG†	4.5	4.5	16/4.5	11.06/11.16	**10.30**/10.43	**6.02/6.03**	**5.31/5.32**	**4.38/4.38**	**3.79**	**5.81/5.82**	**5.11/5.11**

* In OAMAG, we achieve $bit_{avg} = 4.25$ when the bit-width of the group-wise scale is not fused into every group's bit-width.
† In OAMAG, we achieve $bit_{avg} = 4.5$ when all groups utilizing outlier preservation, and both the bit-width of the group-wise scale and outlier's extra bit-width are not fused into every group's bit-width.

TABLE IV: Zero-Shot Results on Multiple Benchmarks (PQ: PIQA, Ae: ARC-e, Ac: ARC-c, BQ: BoolQ, HS: HellaSwag, WG: Winogrande)

Method	Bit-Width W/A/KV	OPT-6.7B PQ/Ae/Ac/BQ/HS/WG	Average	Llama-7B PQ/Ae/Ac/BQ/HS/WG	Average	Llama-2-7B PQ/Ae/Ac/BQ/HS/WG	Average
FP16	16/16/16	76.55/60.06/34.64/66.06/67.21/65.27	61.63	77.37/52.53/41.38/73.12/72.99/66.85	64.04	76.93/53.53/40.61/71.16/72.94/67.17	63.72
Tender	4/4/4	74.32/56.82/33.79/58.93/64.54/61.80	58.37	69.70/58.50/36.26/69.30/57.30/59.04	58.35	58.98/49.03/30.46/64.28/46.26/53.67	50.45
Atom	4.25/4.25/4.25	48.80/26.73/24.91/37.83/25.26/49.01	35.42	76.28/52.10/38.99/69.79/69.81/63.69	**61.78**	74.76/50.51/38.91/69.36/69.37/64.48	61.23
OPAL	4.06/(4.41/7.31)/16	-	-	-	-	75.24/-/37.80/-/-/-	-
OAMAG	4/4/4	74.37/56.73/33.28/64.13/63.84/62.51	**59.14**	76.01/50.00/39.51/73.00/69.89/62.12	61.76	**76.06**/51.47/**38.23**/68.35/70.16/63.14	**61.24**

weight-stationary systolic arrays with the same global buffer size and memory bandwidth to maximize computational utilization. Under iso-area constraints, output dimension is fixed at 64 to standardize the number of accumulators and encoders across each design. For Oltorn and OliVe, vector processing units (VPUs) functionally equivalent to Tender's implementationn are integrated as encoders to perform requantization operations on systolic array outputs. OPAL's native layer-wise 4/7-bit mixed-precision strategy is constrained to uniform 4-bit operations for equitable comparison, with non-essential functional units ($\log_2$-based softmax units) removed to isolate linear layer computation capabilities.

B. Quantization Performance Results

Perplexity. Table III shows perplexity results under $bit_{avg} = 4/4.25/4.5$, three bit-width configuration settings. At $bit_{avg} = 4$ - our targeted bit-width, OAMAG achieves state-of-the-art quantization results across all models as the only work supporting KV4 quantization. It demonstrates particularly outstanding performance on Llama-series models, outperforming the SOTA work Oltron (W4.125A4.1KV16) by an average of 62.07% in perplexity. For a fair comparison, at $bit_{avg} = 4.25$, compared with Atom, OAMAG shows slightly inferior results only on the OPT suite and Llama-7B but achieves better performance on all other Llama models, with an average improvement of 10.16%. At $bit_{avg} = 4.5$, we present two variants of OAMAG: with KV cache unquantized and quantized to 4.5 bits to compare with OPAL. OAMAG (W4.5A4.5KV4.5) presents the best quantization results with an average of 5.56% perplexity loss. OPAL surpasses other works including ours on OPT models due to its particularly higher bit-width but shows limited effectiveness on Llama models - it even underperforms OAMAG (W4A4KV4) on Llama-2-7B. Notably, OAMAG shows superior generality across different models, even outperforms OPAL under KV16 on OPT-13B.

Zero Shot Accuracy. Table IV compares the zero-shot accuracy of OAMAG against Tender, Atom, and OPAL across six benchmark tasks. Olive and Oltron are omitted as they lack zero-shot evaluation capability. As the only framework supporting full $bit_{avg} = 4$ quantization (W4A4KV4), OAMAG achieves the best accuracy on both OPT and Llama-2 models at the lowest bitwidth, while maintaining near-competitive performance on Llama model (just 0.02 loss behind Atom). Additionally, OAMAG demonstrates remarkable robustness, showing a maximum of 4% accuracy degradation across all models. In contrast, we observe significant limitations in other methods: Atom suffers nearly 43% accuracy drop on OPT model while Tender shows 21% degradation on Llama-2 model.

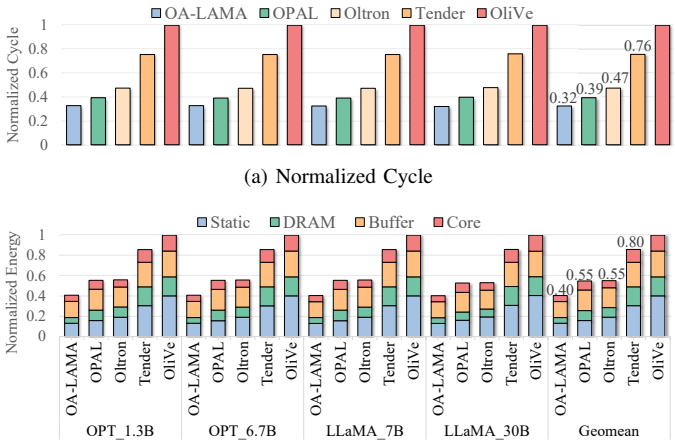
TABLE V: Area of OA-LAMA Accelerator and Baseline Accelerator

Arch.	**Core** Component	Number	Area	**Other**
OA-LAMA	Decoder(38.98μm^2)	8		Accumlation Units*: #64, 0.008mm^2
	TIF PE(303.24μm^2)	1536		
	INT8 PE(296.18μm^2)	256	0.482mm^2	
	Tile Accumulator(320.71μm^2)	256		
	Encoder†(464.69μm^2)	4		
OPAL	Data Distributor(2996.09μm^2)	27	0.481mm^2	VPUs†: #64, 0.069mm^2
	Compute Lane*(14370.53μm^2)	27		
	FP Adder Tree(1453.22μm^2)	4		
	Quantizer†(5948.00μm^2)	1		
Tender	4bit PE(297.02μm^2)	1792	0.480mm^2	Buffer: 128KB, 0.55mm^2
Oltron	Decoder(44.69μm^2)	2		
	Efficient PE(163.30μm^2)	2624	0.483mm^2	
	Flexible PE(296.18μm^2)	256		
OliVe	Decoder(173.05μm^2)	85	0.479mm^2	
	4bit PE(385.56μm^2)	1344		

* All accelerators are equipped with 64 accumulation units except OPAL, which implements accumulation internally within compute lanes.
† All designs incorporated 64 VPUs except OA-LAMA and OPAL, which feature functionally equivalent components respectively.

C. Accelerator Implementation Results

Area. Table V presents the area breakdown and component configurations of OA-LAMA and baseline accelerators, all normalized to 0.48mm^2 under TSMC 28nm technology. OPAL employs data distributors and compute lanes for outlier-aware quantization, with each lane supporting 128 4-bit and 4 BF16 MAC operations. To



(a) Normalized Cycle

(b) Normalized Energy

Fig. 8: Comparison on performance (a) and energy efficiency (b) between OA-LAMA and other baselines

ensure iso-area configuration, OPAL is configured with 27 compute lanes. Given the original design where 8 compute lanes share one FP Adder Tree, we provisionally allocate 4 FP Adder Trees for this implementation. Tender's architecture implements 28×64 PEs, while the original 7:1 ratio between Efficient PEs and Flexible PEs in Oltron is adapted to a (41:4)×64 configuration under iso-area constraints to maintain hardware resource parity. OliVe's 4-bit PEs demonstrate 21×64 PE allocation due to the large circuit footprint of E4M4 PEs in the original paper. The VPU is implemented as 12-stage pipelined dividers for requantization. Experimental results demonstrate that the TIF PE achieves comparable PPA metrics to INT8 PE while outperforming other baseline accelerators, with OA-LAMA's encoding logic introducing negligible area overhead (<0.3% of total area) compared to baseline implementations.

Performance. Fig. 8(a) illustrates the normalized cycle of OA-LAMA and baseline accelerators across varying models. OliVe exhibits the worst latency due to its E4M4 PEs designed to accommodate outlier magnitudes, while Tender incurs latency penalties from shifted accumulation results, both requiring extended accumulator bit-widths, which incurs severe hardware overhead. Oltron benefits from area-efficient PEs but suffers from significant requantization overhead due to FP16 scaling factors. OPAL's exponential scaling method minimizes requantization overhead, but its BF16 outliers introduce memory bandwidth and computational unit overhead. In contrast, OA-LAMA achieves superior throughput through memory-aligned group quantization and timing-balanced PE design, demonstrating 3.09×, 2.34×, 1.46×, and 1.21× speedup over OliVe, Tender, Oltron, and OPAL, respectively, while maintaining higher accuracy.

Energy. Fig. 8(b) compares normalized energy consumption across cores, global buffers, and DRAM. The weight-stationary architecture induces substantial buffer access energy across all designs. OliVe and Tender exhibit the highest energy costs due to wide bit-width accumulator overhead, while OPAL incurs non-negligible memory access overhead from additional outlier and scaling factor handling, resulting in energy consumption comparable to Oltron. By contrast, OA-LAMA's memory-aligned group format minimizes energy costs, achieving 2.47×, 2.12×, 1.36×, and 1.35× energy savings versus OliVe, Tender, Oltron, and OPAL respectively. This efficiency stems from coordinated bit-width reduction in memory transactions (41% fewer DRAM access energy compared to OPAL) and elimination of floating-point conversion logic.

TABLE VI: Ablation Study on Different Quantization Techniques used in OA-LAMA on WikiText2 PPL↓

Quantization Method	**OPT-6.7B**	**Llama-7B**
FP16 (baseline)	10.86	5.68
W4A4 g128 FP16 scale	13.12	6.76
W4A4 g16 exponent scale	13.37	6.38
+ Keeping each group's outlier in INT8	11.06 (2.31↓)	6.02 (0.36↓)
W4A4 g16 fused expoenent scale/ outlier extra bit-width		
Normal group only (int4/3)	13.87	6.74
Outlier group only (int8, int4/3)	11.68	6.57
Mixed-group	11.37	6.47
+ KV Cache only	11.69 (0.32↑)	6.47 (0↑)

D. Ablation Study

We measure the impact of the techniques involved in OA-LAMA on model accuracy, using perplexity as the metric, with results shown in Table VI. First, we compare the conventional g128 FP16 scaling and the g16 exponent scaling. The results show that g16 exponent scaling outperforms g128 FP16 scaling on Llama-7B but still lags behind on OPT-6.7B. However, when outliers in each group are quantized to INT8, the method significantly outperforms FP16 method on both models. Next, we incorporate the package-level memory alignment considerations to achieve true W4A4 quantization. Accounting for outlier groups yields substantially lower perplexity than ignoring them, and further improvements are achieved by introducing mixed grouping. When applying KV cache quantization, OPT-6.7B exhibits a 0.32 increase in perplexity, while Llama-7B shows no measurable increases. This demonstrates OA-LAMA's advantage in KV cache quantization.

VI. Conclusions

This work addresses the critical challenge of achieving both memory alignment and model accuracy in quantized LLM deployment, particularly when handling non-uniform outlier distributions that disrupt conventional quantization approaches. We present OA-LAMA, a co-designed framework that resolves this dual challenge through three key innovations: (1) the memory-aligned OAMAG format that maintains DRAM-compatible access patterns while preserving outlier precision, (2) adaptive group allocation for inter-layer variance, and (3) a three-level accumulation architecture with timing-balanced PEs that supports efficient mixed-precision computation. Experimental results demonstrate OA-LAMA's superior accuracy over 4-bit quantization methods while delivering 1.21-3.09× speedup and 1.35-2.47× energy efficiency gains compared to state-of-the-art accelerators. These advancements establish new Pareto frontiers in accuracy-efficiency optimization while fully maintaining memory alignment across all hierarchy levels, enabling more practical deployment of large language models.

Acknowledgement

This work is supported by National Key Research and Development Program of China (No. 2024YFB4504200), the Guangzhou-HKUST(GZ) Joint Funding Program (No. 2025A03J3568), and partly by the National Science Foundation of China Fund (No. U24B20151) and Shenzhen Bureau of Science and Technology Research Fund (No. KJZD20240903102708012). We also thank the AMD Heterogeneous Accelerated Compute Cluster (HACC) Program at NUS for providing access to hardware resources.

REFERENCES

[1] J. Koo, D. Park, S. Jung, and J. Kung, "Opal: Outlier-preserved microscaling quantization accelerator for generative large language models," in *Proceedings of the 61st ACM/IEEE Design Automation Conference*, 2024, pp. 1–6.

[2] C. Guo, J. Tang, W. Hu, J. Leng, C. Zhang, F. Yang, Y. Liu, M. Guo, and Y. Zhu, "Olive: Accelerating large language models via hardware-friendly outlier-victim pair quantization," in *Proceedings of the 50th Annual International Symposium on Computer Architecture*. IEEE, 2023, pp. 1–15.

[3] Y. Zhao, C.-Y. Lin, K. Zhu, Z. Ye, L. Chen, S. Zheng, L. Ceze, A. Krishnamurthy, T. Chen, and B. Kasikci, "Atom: Low-bit quantization for efficient and accurate llm serving," *Proceedings of Machine Learning and Systems*, vol. 6, pp. 196–209, 2024.

[4] J. Lee, W. Lee, and J. Sim, "Tender: Accelerating large language models via tensor decomposition and runtime requantization," *arXiv preprint arXiv:2406.12930*, 2024.

[5] C. Xue, C. Zhang, X. Jiang, Z. Gao, Y. Lin, and G. Sun, "Oltron: Algorithm-hardware co-design for outlier-aware quantization of llms with inter-/intra-layer adaptation," in *Proceedings of the 61st ACM/IEEE Design Automation Conference*, 2024, pp. 1–6.

[6] B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal *et al.*, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, vol. 1, p. 3, 2020.

[7] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[8] Microsoft, "Github copilot," https://github.com/features/copilot, 2023.

[9] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi *et al.*, "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning," *arXiv preprint arXiv:2501.12948*, 2025.

[10] N. Corporation, "Nvidia blackwell b200 gpu," https://www.nvidia.com/en-us/data-center/dgx-b200/, 2024.

[11] 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.

[12] S. Li, X. Ning, L. Wang, T. Liu, X. Shi, S. Yan, G. Dai, H. Yang, and Y. Wang, "Evaluating quantized large language models," *arXiv preprint arXiv:2402.18158*, 2024.

[13] J. Lin, J. Tang, H. Tang, S. Yang, W.-M. Chen, W.-C. Wang, G. Xiao, X. Dang, C. Gan, and S. Han, "Awq: Activation-aware weight quantization for on-device llm compression and acceleration," *Proceedings of Machine Learning and Systems*, vol. 6, pp. 87–100, 2024.

[14] 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.

[15] Y. Lin, H. Tang, S. Yang, Z. Zhang, G. Xiao, C. Gan, and S. Han, "Qserve: W4a8kv4 quantization and system co-design for efficient llm serving," *arXiv preprint arXiv:2405.04532*, 2024.

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

[17] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, "Gptq: Accurate post-training quantization for generative pre-trained transformers," *arXiv preprint arXiv:2210.17323*, 2022.

[18] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, "Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale," *Advances in Neural Information Processing Systems*, vol. 35, pp. 30 318–30 332, 2022.

[19] Z. Yao, R. Yazdani Aminabadi, M. Zhang, X. Wu, C. Li, and Y. He, "Zeroquant: Efficient and affordable post-training quantization for large-scale transformers," *Advances in Neural Information Processing Systems*, vol. 35, pp. 27 168–27 183, 2022.

[20] S. Ashkboos, I. Markov, E. Frantar, T. Zhong, X. Wang, J. Ren, T. Hoefler, and D. Alistarh, "Quik: Towards end-to-end 4-bit inference on generative large language models," *arXiv preprint arXiv:2310.09259*, 2023.

[21] C. Guo, C. Zhang, J. Leng, Z. Liu, F. Yang, Y. Liu, M. Guo, and Y. Zhu, "Ant: Exploiting adaptive numerical data type for low-bit deep neural network quantization," in *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2022, pp. 1414–1433.

[22] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "Qlora: Efficient finetuning of quantized llms," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[23] C. Lee, J. Jin, T. Kim, H. Kim, and E. Park, "Owq: Outlier-aware weight quantization for efficient fine-tuning and inference of large language models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 12, 2024, pp. 13 355–13 364.

[24] Z. Yuan, Y. Shang, Y. Zhou, Z. Dong, Z. Zhou, C. Xue, B. Wu, Z. Li, Q. Gu, Y. J. Lee *et al.*, "Llm inference unveiled: Survey and roofline model insights," *arXiv preprint arXiv:2402.16363*, 2024.

[25] C. Hooper, S. Kim, H. Mohammadzadeh, M. W. Mahoney, S. Shao, K. Keutzer, and A. Gholami, "Kvquant: Towards 10 million context length llm inference with kv cache quantization," *Advances in Neural Information Processing Systems*, vol. 37, pp. 1270–1303, 2024.

[26] Z. Liu, J. Yuan, H. Jin, S. Zhong, Z. Xu, V. Braverman, B. Chen, and X. Hu, "Kivi: A tuning-free asymmetric 2bit quantization for kv cache," *arXiv preprint arXiv:2402.02750*, 2024.

[27] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, and I. Stoica, "Efficient memory management for large language model serving with pagedattention," in *Proceedings of the 29th Symposium on Operating Systems Principles*, 2023, pp. 611–626.

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

[29] N. G. Bita Darvish Rouhani, A. M. Tom Savell, M. Z. Kyung-Nam Han, J. K. Ritchie amd Hall, Y. Y. Eric Chung, R. W. Michael Schulte, N. S. Ian Bratt, J. B. Jelena Milanovic, M. C. Pradeep Dubey, A. R. Alexander Heinecke, S. D. Martin Langhammer, M. S. Maxim Naumov, Paulius Micikevicius, and C. Verrilli, "Ocp microscaling (mx) specification," *Open Compute Project*, 2023.

[30] B. D. Rouhani, R. Zhao, A. More, M. Hall, A. Khodamoradi, S. Deng, D. Choudhary, M. Cornea, E. Dellinger, K. Denolf *et al.*, "Microscaling data formats for deep learning," *arXiv preprint arXiv:2310.10537*, 2023.

[31] 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.

[32] Z. Chen, B. XIE, J. Li, and C. Shen, "Channel-wise mixed-precision quantization for large language models," *arXiv preprint arXiv:2410.13056*, 2024.

[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[34] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, 2017, pp. 1–12.

[35] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[36] 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.

[37] W. contributors, "68-95-99.7 rule - wikipedia," in *The Free Encyclopedia*. [Online], 2022.

[38] L. Gao, J. Tow, S. Biderman, S. Black, A. DiPofi, C. Foster, L. Golding, J. Hsu, K. McDonell, N. Muennighoff, J. Phang, L. Reynolds, E. Tang, A. Thite, B. Wang, K. Wang, and A. Zou, "A framework for few-shot language model evaluation," Sep. 2021, zenodo.

[39] P. Kurup and T. Abbasi, *Logic Synthesis Using Synopsys*, 2nd ed. Springer Publishing Company, Incorporated, 2011.

[40] N. Muralimanohar, R. Balasubramonian, and N. Jouppi, "Cacti 6.0: A tool to model large caches. hp laboratories," *Tech. Rep. HPL-2009-85*, 2009.

[41] H. Sharma, J. Park, D. Mahajan, E. Amaro, J. K. Kim, C. Shao, A. Mishra, and H. Esmaeilzadeh, "From high-level deep neural models to fpgas." in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. Ieee, 2016, pp. 1–12.

[42] S. Sarangi and B. Baas, "Deepscaletool: A tool for the accurate estimation of technology scaling in the deep-submicron era." in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. Ieee, 2021, pp. 1–5.