

AO-BFP: An Adaptive Mixed-Precision and Outlier-Aware Block Floating-Point Accelerator for Large Language Model Inference

Yifan Wang, Zetao Guo, Wendi Sun, Wenhao Sun, Qiyan Fang, Song Chen[✉], Yi Kang

University of Science and Technology of China

{wangyif513, ztguo, wdsun, wh1997, qiyanfang}@mail.ustc.edu.cn

{songch, ykang}@ustc.edu.cn

Abstract—Large Language Models (LLMs) have achieved remarkable success in Natural Language Processing (NLP) tasks, but their deployment is severely constrained by intensive computation and memory costs. Block Floating-Point (BFP) extends the dynamic range beyond INT with shared exponents, while reducing memory and alignment overhead compared to floating-point formats. However, when bit-widths are further reduced, BFP becomes sensitive to outliers; existing mixed-precision BFP methods largely rely on heuristic settings of mantissa width and block size, and the induced bit-level sparsity has yet to be systematically leveraged in hardware. In this paper, we propose AO-BFP, an adaptive BFP framework for LLM inference. At the algorithm level, we propose an adaptive outlier exponent mapping mechanism combined with mixed-precision exploration driven by layer-wise sensitivity analysis. At the hardware level, we design a reconfigurable bit-serial accelerator with a unified datapath that efficiently leverages BFP-induced bit sparsity. Compared with prior LLM accelerators such as ANT, OliVe, and BitMoD, AO-BFP achieves superior performance while preserving model accuracy, delivering speedups of 1.61 \times , 1.39 \times , and 1.11 \times , respectively.

Index Terms—LLM inference, block floating-point, mixed precision quantization, outlier, bit-serial accelerator

I. INTRODUCTION

LLMs have demonstrated remarkable effectiveness across a wide range of NLP tasks [1], [2], and their language understanding capability has also been successfully extended to multi-modal applications [3]. Nevertheless, the intensive computational and memory requirements of LLMs remain major obstacles to their sustainable use and further development.

Benefiting from native GPU support, INT8 and other fixed-point formats have been widely adopted in prior quantization studies [4], [5]. However, the limited dynamic range of integer types makes them highly sensitive to extreme values and wide distributions, leading to large errors under low-bit settings. To mitigate this issue, some researchers have explored finer-grained scaling factors such as per-token or per-group quantization [6], [7], but such methods incur significant overhead by storing additional scaling factors and introducing costly

This work was supported in part by the Graduate School Special Funding Program of the University of Science and Technology of China, in part by the National Natural Science Foundation of China under Grant 92473114, in part by the Chinese Academy of Sciences (CAS) Project for Young Scientists in Basic Research under Grant YSBR-029, and in part by the Strategic Priority Research Program of CAS under Grant XDB0660000.

dequantization operations. To strike a better balance between memory cost and representational capacity, BFP quantization has been proposed as an intermediate format between fixed-point and floating-point representations [8]. Instead of assigning each value its own exponent, BFP groups multiple mantissas under a shared exponent, which reduces storage and hardware overhead. At the same time, it eliminates intra-block alignment and avoids costly dequantization multiplications, while retaining the dynamic range comparable to floating-point.

Mixed-precision quantization has been incorporated into the BFP framework to flexibly allocate precision for improved accuracy-efficiency tradeoffs. However, existing mixed-precision BFP quantization [9] typically relies on heuristic or empirical choices of mantissa width and block size, lacking systematic joint optimization. In particular, outliers dominate shared-exponent selection and force normal values to be right-shifted, sharply increasing quantization error. This effect is further amplified under low bit-widths and large block sizes, which misguides the precision configuration search toward conservative settings, thereby limiting the potential of mixed-precision strategies.

On the other hand, the shared-exponent nature of BFP introduces a unique advantage: when mantissas are right-shifted for exponent alignment, high-order bits are consistently filled with zeros, leading to bit-level sparsity. It can be exploited in hardware by detecting and skipping redundant operations, offering additional opportunities for efficiency gains. However, existing accelerators offer little native support, leaving the benefit underutilized. To address these challenges, this paper proposes AO-BFP, a novel adaptive BFP framework that enables holistic co-optimization from quantization algorithms to hardware execution.

- We propose Adaptive Outlier Exponent Mapping with Local Outlier Clustering (AOEM-LOC), which effectively alleviates the interference of outliers on shared exponent selection.
- We propose a mixed-precision BFP framework that jointly optimizes mantissa width and block size, guided by Hessian-based layer-wise sensitivity analysis and solved via Pareto-constrained knapsack search for globally optimal precision allocation.

- We design and implement a unified and reconfigurable BFP hardware architecture, which employs a bit-serial datapath to flexibly support mixed precision, outlier compensation, and bit-sparsity skipping. The architecture further maintains compatibility with both general matrix–matrix multiplication (GEMM) and general matrix–vector multiplication (GEMV) computation modes.

II. BACKGROUND AND MOTIVATION

A. Block Floating-Point Representation

An IEEE 754 half-precision floating-point number consists of three components: a 1-bit sign s , a 5-bit exponent e , and a 10-bit mantissa m , as shown in Fig. 1(a). The actual value is expressed as

$$x = (-1)^s \cdot (1.m) \cdot 2^{e - e_{bias}} \quad (1)$$

BFP assigns a group of B numbers to a shared exponent. Formally, a block of B values can be expressed as

$$x_i = (-1)^{s_i} \cdot m_i \cdot 2^{E_{\text{shared}} - E_{\text{bias}}}, \quad i = 1, 2, \dots, B \quad (2)$$

where s_i is the sign bit, m_i is the mantissa of the i -th value, and E_{shared} denotes the block-wise shared exponent.

To convert floating-point numbers into the BFP format, the shared exponent is selected as $E_{\text{shared}} = \max(e_i) + 1$, which shifts the implicit leading one into the fractional part of the mantissa. The exponent difference is given by $d_i = E_{\text{shared}} - e_i$. Each mantissa is then right-shifted by d_i bits to align with the shared exponent: $\hat{m}_i = m_i \gg d_i$. The quantized BFP value follows the same form as Eq. (2), with m_i replaced by \hat{m}_i .

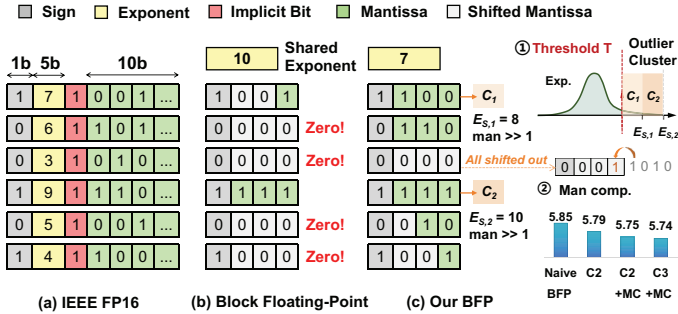


Fig. 1. An example of a tensor at different formats. (a) IEEE FP16, (b) Block Floating-Point, (c) AO-BFP used AOEM-LOC, evaluated on LLaMA-7B [10] under the W6A6 setting.

B. Motivation

Outlier misleads precision allocation. Eqs. (3) and (4) show the impact of outliers on BFP quantization error, with unit least precision (ULP) and mean squared error (MSE):

$$ULP = 2^{E_{\text{max}} - w + 1}, \quad MSE = ULP^2 / 12 \quad (3)$$

When a block contains an outlier with an exponent exceeding normal values by Δ , it dominates the shared exponent, and the ULP and MSE of normal values are:

$$\frac{ULP_{\text{outlier}}}{ULP_{\text{base}}} = 2^{E_{\text{outlier}}^{\text{max}} - E_{\text{normal}}^{\text{max}}} = 2^{\Delta}, \quad \frac{MSE_{\text{outlier}}}{MSE_{\text{base}}} = 2^{2\Delta} \quad (4)$$

MSE increases exponentially with $2^{2\Delta}$, so even a small 2-3 bit gap can invalidate a 3-bit mantissa, as shown in Fig. 1(b). This effect is especially pronounced under low bit-width and large block size. As a result, outliers mislead precision allocation in mixed-precision settings, driving the search toward conservative configurations and limiting overall efficiency.

Existing outlier-aware approaches either lose accuracy at low bit-widths [11], or rely on hardware-unfriendly solutions [12], [13]. A robust quantization scheme should decouple shared exponent selection from outliers while preserving a unified datapath with minimal hardware overhead. In Section III-A, we present AOEM-LOC to mitigate outlier dominance, and in Section IV-B a unified-path alignment mechanism is introduced to handle outliers within the main datapath.

Mixed-Precision BFP: Joint Mantissa–Block Optimization. Existing mixed-precision BFP quantization methods often select W_{man} and Blk_{size} heuristically rather than through systematic co-optimization. The two are inherently complementary: smaller blocks capture local ranges at the cost of storage, while wider mantissas improve accuracy at higher overhead. Prior works address this tradeoff only coarsely: DBPS [14] applies uniform, independent adjustments across the model, whereas BitQ [15] jointly searches W_{man} and Blk_{size} but still relies on heuristic rules without theoretical guidance.

In Section III-B, we propose a sensitivity-driven joint optimization framework: Hessian eigenvalues guide per-layer sensitivity analysis, candidate pairs in the discrete $(W_{\text{man}}, Blk_{\text{size}})$ space form Pareto frontiers, and the resulting sets are solved as a multi-dimensional knapsack problem via dynamic programming for globally optimal allocation.

BFP-Induced Bit Sparsity Remains Underutilized. BFP quantization is also accompanied by a favorable property—bit-level sparsity. During the right-shift alignment of mantissas, many low-order bits are truncated, significantly increasing the overall sparsity of neural networks. As shown in Fig. 2, after applying BFP quantization across different OPT [16] model sizes and configurations, the mantissa bit-level sparsity rises from 38%-49% to 62%-68%. Such sparsity lowers the computation density, but its potential for energy-efficiency optimization remains unexploited. While a natural way to exploit this property is through bit-serial multipliers, we further extend it into a unified and reconfigurable architecture that efficiently supports different block sizes and mantissa widths, while enabling bit-sparsity exploitation and outlier handling.

Building on the above observations, we propose AO-BFP, an accelerator with algorithm-hardware co-design. Fig. 3 shows its top-level architecture, consisting of an adaptive quantization framework and a bit-level execution engine (Sections III, IV).

III. QUANTIZATION FRAMEWORK

A. Outlier-Aware Exponent Mapping

To mitigate the loss caused by outliers in BFP quantization, this paper proposes a method named Adaptive Outlier Exponent Mapping with Local Outlier Clustering (AOEM-LOC). This method strategically derives the shared exponent solely from normal values, while outliers effectively map to a discretized

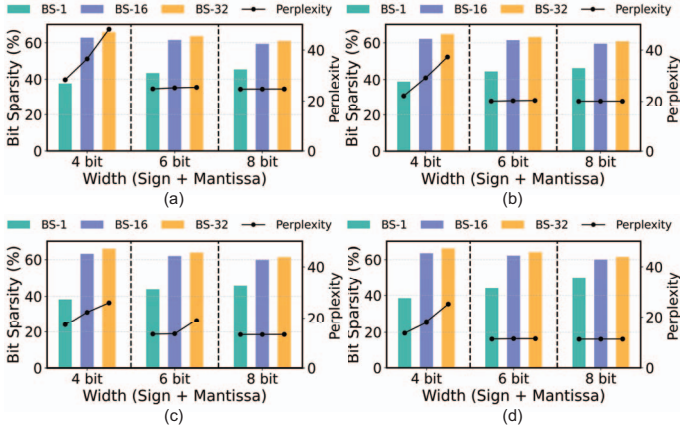


Fig. 2. Bit sparsity and perplexity of OPT models before and after BFP quantization under different W_{man} and Blk_{size} . Bars indicate sparsity and lines indicate perplexity. Quantization increases sparsity by about 20%, and the resulting accuracy variation motivates our use of mixed precision. (a)-(d) correspond to OPT-125M, OPT-350M, OPT-1.3B, and OPT-2.7B, respectively.

exponent set $E_S = \{E_{s,1}, \dots, E_{s,K}\}$. The difference in exponent is compensated by a right-shift in mantissa. Notably, this discretization does not compress outliers into a single value but instead employs a clustered set of discrete exponents and corresponding mantissas to cover a range, ensuring that outliers retain reasonable numerical representation without incurring additional floating-point storage overhead. As depicted in Fig. 1(c), assuming a threshold $T = 6$, with $E_{s,1} = 8$ and $E_{s,2} = 10$, the exponent of the first number is mapped to $E_{s,1}$ with its mantissa right-shifted by 1-bit, and the second number is mapped to $E_{s,2}$ with a corresponding 1-bit right shift.

To jointly optimize the threshold T and the exponent set E_S , a two-step one-dimensional k-means strategy is employed. First, all exponents are clustered with $k = 2$ to separate normal values and outliers, where the maximum exponent of the lower cluster defines the threshold T . Subsequently, a $k = K$ clustering is applied to the outlier cluster to generate the compact exponent set E_S . Each outlier exponent is mapped to one of these special points, and an index is recorded to preserve positional information $\langle pos_i, E_s^k \rangle$. Since the proportion of outliers is controlled below 1%, the index table is minimal and can be stored with low overhead through compact encoding. Furthermore, AOEM-LOC introduces a mantissa compensation mechanism to address cases where a large shared exponent shifts the mantissa out of range, as shown in Fig. 1(c). If the highest discarded bit is 1, the least significant bit (LSB) of the mantissa is forcibly set to 1 to improve numerical fidelity.

Fig. 1(c) validates the effectiveness of above approaches: compared to naïve BFP, incorporating outlier mapping ($k = 2$ clusters) reduces perplexity, and combining it with mantissa compensation (Clus2 + MC) further improves numerical precision, demonstrating their complementary roles in enhancing performance. The experiments also investigate the impact of the second-stage clustering number K . Increasing K from 2 to 3 yields a negligible improvement in perplexity but a higher implementation cost. Hence, we adopt $K = 2$ as a balanced

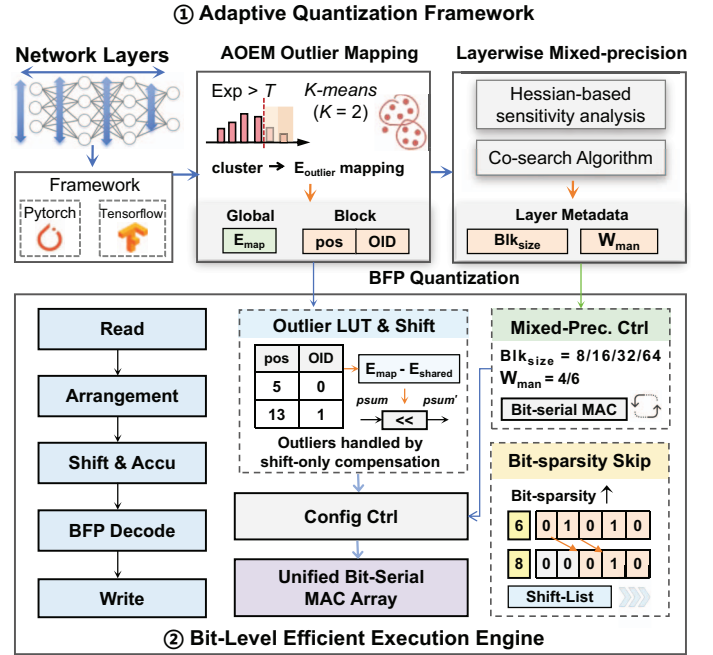


Fig. 3. Top-Level architecture of the proposed AO-BFP accelerator.

choice for accuracy, storage, and hardware feasibility.

B. Layer-wise mixed-precision exploration

In our mixed-precision design, we further consider the joint design space of mantissa width and block size. Specifically, we expand the search space to:

$$C = \{(W_{\text{man}}, Blk_{\text{size}}) \mid W_{\text{man}} \in \{4, 6\}, Blk_{\text{size}} \in \{8, 16, 32, 64\}\}$$

Direct enumeration of all 8^L combinations is infeasible. Inspired by Hessian-based mixed-precision quantization [17], we adopt the largest eigenvalue of the Hessian matrix to characterize the sensitivity of layer l to quantization noise:

$$\Delta \ell \approx \sum_{i=1}^L \sum_{j=1}^L \frac{1}{2} \varepsilon_i H_{ij} \varepsilon_j, \quad (5)$$

H_{ij} denotes the Hessian matrix elements and ε_i represents perturbations to the parameters. However, in LLaMA-2-7B, each decoder layer contains about 153M parameters, yielding a $7M \times 7M$ Hessian matrix, making second-order statistics intractable. We therefore approximate the eigenvalue via the power iteration method [18]. By sorting layers according to their maximum Hessian eigenvalues, we obtain a relative sensitivity ordering. As shown in Fig. 4, with the largest λ_{max} , layer 20 is the most sensitive to perturbations, and perturbing it along the corresponding dominant eigenvector yields the steepest loss increase. Layers with smaller λ_{max} (e.g., layer 1) exhibit only mild loss changes under the same perturbation.

Although layer sensitivities are known, allocating W_{man} and Blk_{size} remains nontrivial. To address this, we initially adopt a classical greedy strategy by sorting decoder layers in ascending order of sensitivity. For each layer l , we take the most aggressive setting (4, 64) first, which yields the highest

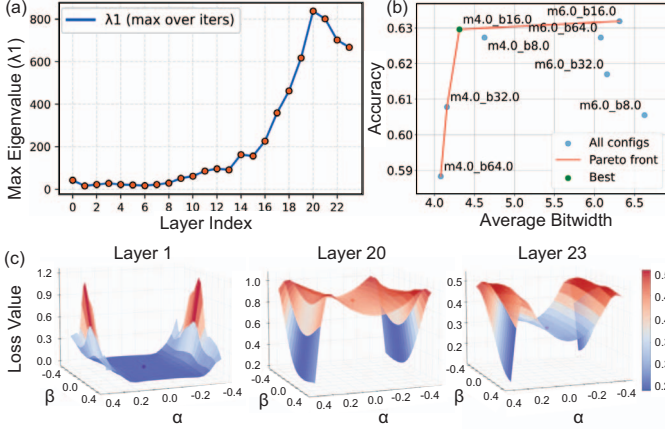


Fig. 4. (a) Maximum eigenvalue (λ_{\max}) of the Hessian matrix for each layer of OPT-1.3B, used to measure relative sensitivity. (b) Average bit-width and accuracy of layer 2 under different $(W_{\text{man}}, \text{Blk}_{\text{size}})$ configurations, with the Pareto frontier highlighted. (c) 3-D loss landscape of layers 1, 20, and 23 along the first dominant eigenvector of the Hessian.

compression ratio, each layer configuration is fixed if the resulting global loss increase ΔL is acceptable; otherwise, W_{man} is increased or Blk_{size} is reduced until the error falls within budget.

However, empirical evaluation reveals two major limitations: (i) the layer-wise error response to $(W_{\text{man}}, \text{Blk}_{\text{size}})$ is not monotonic—particularly in high-dimensional activation spaces, extreme settings such as *maximal mantissa with minimal block size* are often suboptimal; (ii) the greedy iteration lacks a global perspective under inter-layer error coupling—if early layers consume excessive error budget, subsequent highly sensitive layers may be left infeasible, leading to local optima.

To address these issues, we employ a layer-wise Pareto framework. For a given layer l , we enumerate all candidate combinations in C , evaluate their average bit-width r and accuracy a , and construct the Pareto set, as shown in Fig. 4(b):

$$\mathcal{PF}_l = \{k \mid \nexists k' (r_{l,k'} < r_{l,k} \wedge a_{l,k'} \geq a_{l,k})\} \quad (6)$$

which retains only non-dominated solutions in (r, a) space. Based on the evaluation, $|\mathcal{PF}_l|$ is typically no larger than 4. The union of all per-layer Pareto sets then serves as the candidate pool of a multi-dimensional knapsack problem, which we solve via dynamic programming under a global average bit-width constraint to maximize accuracy, offering a global view that avoids the suboptimality of layer-wise greedy strategies.

IV. AO-BFP HARDWARE ARCHITECTURE

A. Architecture Overview

The hardware architecture of AO-BFP is illustrated in Fig. 5. It mainly consists of a top-level controller, activation buffer, weight buffer, output buffer, BFP encoder, partial accumulation unit, outlier LUT, and a compute array composed of configurable bit-serial multipliers. During computation, activations are streamed from the input buffer, while weights are offline-encoded into shift lists to exploit bit-level sparsity. The partial accumulation unit adapts to both GEMM and GEMV dataflows.

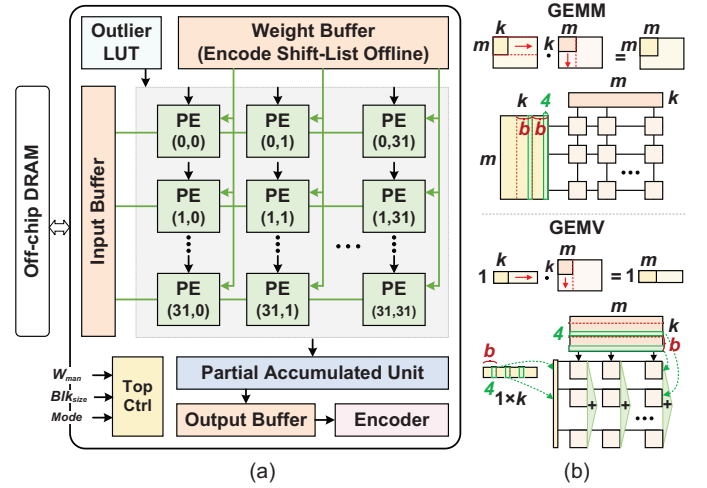


Fig. 5. (a) Overview of AO-BFP hardware architecture. (b) GEMM dataflow used in the prefilling phase, and GEMV used in the decoding phase.

B. AO-BFP Processing Element

As illustrated in Fig. 6, AO-BFP adopts a bit-serial processing element (PE) with three key enhancements: (i) a shift-list mechanism that skips redundant zero-bit operations to exploit bit-level sparsity, (ii) abstraction of block size as sequential computation rounds rather than inter-PE accumulation, eliminating costly FP16 additions and extra exponent handling, (iii) direct outlier compensation within the main datapath through a lightweight shift operation, avoiding separate bypass logic.

Step 1 performs exponent addition and sign-bit generation: the activation exponent a_e and weight exponent w_e are summed as $(a_e + w_e)$ to provide the target for normalization. It also generates the product sign y_s , which is forwarded to the partial-product accumulation path. **Step 2** performs bit-serial mantissa multiplication on four parallel inputs. The mantissa width sets the number of serial cycles. W_{man} includes the sign bit, giving an effective precision of 3-5 bits, with bit sparsity around 60%, leaving only two effective bits on average. After bit-serial multiplication, outlier compensation is uniformly applied in a dedicated barrel shifter (Shifter-EX). The compensation results are merged with the normal path output in the same adder tree, avoiding additional accumulation paths or bypass logic. During offline quantization, global exponents are fixed. At runtime, a local LUT records each outlier's position and load to guide Shifter-EX in computing the shift offset, as defined in Eq. (7).

$$\Delta E = \begin{cases} \Delta E_a & a \in O, w \notin O \\ \Delta E_w & a \notin O, w \in O \\ \Delta E_a + \Delta E_w & a \in O, w \in O \end{cases} \quad (7)$$

$$\Delta E_a = E_a^{\text{map}} - E_a^{\text{shared}}, \quad \Delta E_w = E_w^{\text{map}} - E_w^{\text{shared}}$$

After bit-serial multiplication, Shifter-EX compensates the partial product pp in a single cycle: $pp' = pp \ll \Delta E$, where ΔE denotes the exponent offset of the outlier. pp' is then fused with the normal path output in the same adder tree and stored in registers. This design ensures: regardless of whether outliers

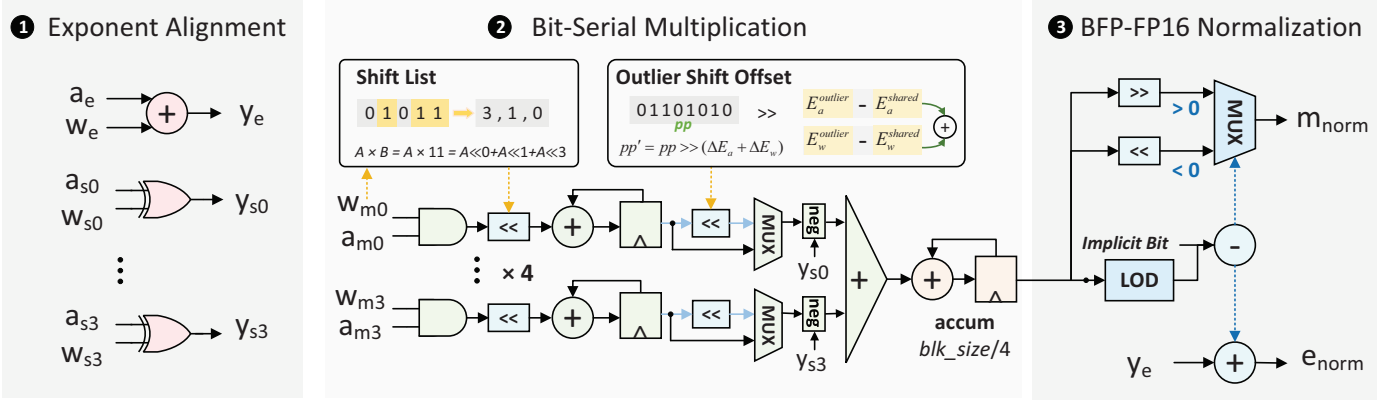


Fig. 6. The microarchitecture of AO-BFP PE.

appear in weights, activations, or both, the datapath remains constant—outliers are corrected only once at the pp level, with only minor impact on delay and control complexity.

AO-BFP supports variable block sizes by abstracting it as computation rounds instead of hardware parallelism. Each PE processes four mantissas per cycle, thus, a block of size B is executed as $B/4$ rounds (e.g., $B = 16$ requires four rounds). After each round of bit-serial multiplication, pp is accumulated into a register while the round counter is updated. Compared with prior designs [14], [19] that rely on reconfigurable adder tree, AO-BFP avoids redundant FP16 inter-PE additions and exponent operations, thereby reducing control complexity and computation overhead. **Step 3** performs normalization. After accumulation, the results are normalized into the BFP-FP format and converted to FP16 for inter-PE accumulation.

C. GEMM/GEMV Reconfigurable Dataflow

To accommodate the diverse matrix operations in the inference phase of LLMs, AO-BFP provides a configurable dataflow between GEMM and GEMV, as shown in Fig. 5(b). m denotes the PE-array dimension, and k denotes the input-channel size of a tile. The main differences lie in the input broadcast scheme and the partial-sum accumulation method. In GEMM mode, the accelerator follows an outer-product dataflow: for each tile, an $m \times 4$ activation panel is broadcast across PE rows, while $4 \times m$ weight panel is broadcast along PE columns. Each PE performs bit-serial MACs with 4-way parallelism and accumulates partial sums locally. After each BFP block MAC completes, each PE’s results are transferred to partial accumulation unit and accumulated individually until the final output is produced.

In GEMV mode, to avoid the low utilization of conventional output-stationary dataflow, AO-BFP partitions the input vector into sub-blocks computed independently within each column. Weight matrices of size $4m \times m$ are preloaded along columns, while the activation vector of size $1 \times 4m$ is broadcast simultaneously to all rows via a global bus. Partial sums are reduced vertically by the partial accumulation unit, with PE columns operating in parallel on vector slices, improving utilization under small-batch scenarios.

V. EVALUATION

A. Experiment Setup

We evaluated the effectiveness of the proposed quantization scheme in OPT and LLaMA on WikiText-2. We compare our method against four open-source baselines: SmoothQuant [20], Ominiquant [21], OliVe [22], and BFP quantization [9]. For all BFP methods, the shared exponent is set to 5-bit, and the block size is set to 32. We implement the AO-BFP hardware architecture in Verilog HDL and synthesize it using Synopsys Design Compiler with the 28nm technology. Furthermore, we implement a cycle-accurate simulator based on BitMoD [23] to evaluate end-to-end performance. We evaluate LLMs with a batch size of 1 and an input sequence length of 256, following prior edge-oriented work [24]. DRAM power (DDR4 model) is estimated with DRAMSim3 [25], and 512 KB activation buffer and 512 KB weight buffer are modeled using CACTI [26].

B. Evaluation on AO-BFP Quantization Scheme

Table I reports the perplexity on the WikiText2 dataset for representative OPT and LLaMA models under different quantization schemes. Our AO-BFP achieves competitive performance with SmoothQuant W8A8 while using only 4.75-bit precision. Compared with BFP, AO-BFP consistently reduces perplexity across all models, demonstrating the effectiveness of adaptive exponent mapping and mixed precision. When compared with 4-bit SmoothQuant or OliVe, AO-BFP exhibits substantially better performance, especially on larger models such as LLaMA-2-13B. For instance, OliVe W4A4 suffers from severe degradation with perplexity above 42, while AO-BFP remains below 6. This highlights the vulnerability of uniform low-bit quantization to outliers, which become increasingly significant as model size grows. AO-BFP alleviates this issue by adaptive bit-width assignment and effective outlier handling.

C. Evaluation on AO-BFP Hardware

Performance. Fig. 7 presents the normalized performance of the FP16 baseline, ANT [27], OliVe [22], BitMoD [23], and our proposed AO-BFP accelerator, in two variants: AO-BFP-D (dense, bit-serial without zero-skipping) and AO-BFP-S (sparse, bit-serial with zero-skipping). ANT, OliVe, and

TABLE I
PERPLEXITY RESULTS OF QUANTIZED MODEL ON WIKITEXT2 (LOWER IS BETTER).

Method	Config	OPT-1.3B	OPT-2.7B	OPT-6.7B	LLaMA-2-7B	LLaMA-2-13B	LLaMA-3-8B
FP32	/	13.35	11.73	10.01	5.81	5.09	6.14
SmoothQuant	W8A8	14.62	12.50	10.85	5.85	4.92	6.34
SmoothQuant	W6A6	15.42	12.68	11.34	7.47	6.97	7.70
SmoothQuant	W4A4	126.05	252.04	491.34	83.12	35.88	430.40
OmniQuant	W6A6	14.99	13.18	10.96	11.26	10.87	/
OmniQuant	W4A4	19.19	15.19	12.48	14.51	13.78	/
OliVe	W4A4	69.07	46.00	107.15	144.78	42.24	/
BFP	W6A6	16.83	11.90	10.04	5.87	5.20	6.18
BFP	W4A4	26.47	26.36	13.64	7.38	6.55	7.46
AO-BFP (Our work)	W4.75A4.75	14.73	12.58	10.94	5.88	5.20	6.21
AO-BFP (Our work)	W4.58A4.58	15.54	13.79	11.39	5.98	5.26	6.38
AO-BFP (Our work)	W4.25A4.25	16.79	15.13	12.24	6.00	5.28	6.47

BitMoD enlarge the numerical range to handle outliers; e.g., BitMoD keeps activations in FP16, so memory traffic and computation remain substantial. In contrast, AO-BFP fully leverages both the dynamic range and bit-level sparsity of BFP. Activations and weights are represented at low precision (4.58-bit on average). Moreover, ANT, OliVe, and BitMoD all employ systolic arrays, which are efficient for GEMM but suffer from low PE utilization for GEMV. AO-BFP adopts a configurable dataflow to support efficient LLM inference across both phases. Overall, AO-BFP-S delivers average speedups of $4.25\times$, $1.61\times$, $1.39\times$, and $1.11\times$ over FP16, ANT, OliVe, and BitMoD, respectively, and achieves an additional $1.28\times$ speedup over AO-BFP-D by exploiting bit-level sparsity.

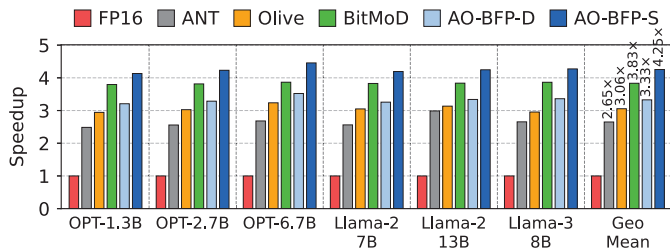


Fig. 7. Speedup of different accelerators (higher is better).

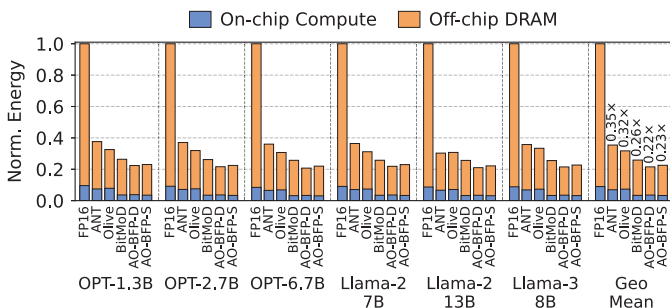


Fig. 8. Energy consumption breakdown of different accelerators.

Energy Consumption. Fig. 8 shows the normalized energy breakdown across accelerators, with on-chip compute energy including both buffer and core energy. AO-BFP-S reduces

TABLE II
AREA BREAKDOWN OF AO-BFP AND OTHER ACCELERATORS.

Architecture	Component	Number	Area (mm^2)
Baseline	FP16 PE ($2630.50 \mu\text{m}^2$)	1024	2.694
BFP	Decoder ($2534.11 \mu\text{m}^2$)	4	1.391
	BFP6 PE ($1348.07 \mu\text{m}^2$)	1024	
AO-BFP	Decoder ($2534.11 \mu\text{m}^2$)	4	1.269
	AO-BFP PE ($1229.63 \mu\text{m}^2$)	1024	

energy mainly by shrinking activation and weight footprints and skipping zero bit-slices induced by BFP. In contrast, ANT, OliVe, and BitMoD require higher activation precision to handle outliers, leading to substantially higher DRAM energy. On average across models, AO-BFP-S achieves $4.44\times$, $1.58\times$, $1.41\times$, and $1.15\times$ higher energy efficiency than FP16, ANT, OliVe, and BitMoD, respectively. Compared to AO-BFP-D, AO-BFP-S increases total energy by only 4.8%, indicating that the memory-access overhead of sparse compression is modest. **Area.** Table II reports the area breakdown of the FP16 baseline, a conventional BFP, and AO-BFP at 28 nm and 400 MHz. Benefiting from bit-width reduction, the two BFP PE designs exhibit substantially lower hardware cost—48% and 53% smaller, respectively, than the FP16 PE. AO-BFP further reduces per-PE area by 8.8% relative to BFP. The decoder overhead is negligible, accounting for only 0.8% of the PE array area.

VI. CONCLUSION

In this paper, we present AO-BFP, an algorithm-hardware co-design framework for efficient LLM acceleration. We propose adaptive exponent mapping and layer-wise mixed-precision BFP to mitigate outliers and reduce bit widths with negligible accuracy loss. A unified bit-serial PE is designed to support mixed-precision BFP and integrate outlier handling into the datapath, while exploiting BFP-induced bit-level sparsity for speedup. Experimental results show that AO-BFP achieves $1.61\times/1.39\times/1.11\times$ speedups and $1.58\times/1.41\times/1.15\times$ energy efficiency gains over ANT/OliVe/BitMoD, respectively.

REFERENCES

- [1] J. Achiam, S. Adler, S. Amni, T. Brown, B. Chan, B. Chess, R. Conerly, C. Dennison, D. Farhi, L. Fedus, *et al.*, “Gpt-4 technical report,” *arXiv preprint arXiv:2303.08774*, 2023.
- [2] DeepSeek-AI, D. Guo, D. Yang, A. Liu, Q. Zhu, Y. Ma, and *et al.*, “Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning,” <https://arxiv.org/abs/2501.12948>, 2025.
- [3] Z. Peng, W. Wang, L. Dong, Y. Hao, S. Huang, S. Ma, and F. Wei, “Kosmos-2: Grounding multimodal large language models to the world,” *arXiv preprint arXiv:2306.14824*, 2023.
- [4] W. Tao, B. Zhang, X. Qu, J. Wan, and J. Wang, “Cocktail: Chunk-adaptive mixed-precision quantization for long-context llm inference,” in *2025 Design, Automation Test in Europe Conference (DATE)*, 2025, pp. 1–7.
- [5] X. Xie, L. Wang, L. Xiao, M. Han, L. Sun, S. Zheng, and X. Xu, “Fineq: Software-hardware co-design for low-bit fine-grained mixed-precision quantization of llms,” in *2025 Design, Automation Test in Europe Conference (DATE)*, 2025, pp. 1–7.
- [6] Z. Liu, B. Oguz, C. Zhao, E. Chang, P. Stock, Y. Mehdad, Y. Shi, R. Krishnamoorthi, and V. Chandra, “LLM-QAT: Data-free quantization aware training for large language models,” in *Findings of the Association for Computational Linguistics: ACL 2024*, 2024, pp. 467–484.
- [7] 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,” in *MLSys*, 2024.
- [8] K. Kalliojarvi and J. Astola, “Roundoff errors in block-floating-point systems,” *IEEE Transactions on Signal Processing*, vol. 44, no. 4, pp. 783–790, 1996.
- [9] 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 *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 9988–10006.
- [10] Meta, “Meta llama,” <https://github.com/meta-llama/llama>.
- [11] S. Ashkboos, I. Markov, E. Frantar, T. Zhong, X. Wang, J. Ren, T. Hoeffler, and D. Alistarh, “QUICK: Towards end-to-end 4-bit inference on generative large language models,” in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Nov. 2024, pp. 3355–3371.
- [12] G. Wang, S. Cai, W. Li, D. Lyu, and G. He, “Ofq-llm: Outlier-flexing quantization for efficient low-bit large language model acceleration,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 72, no. 8, pp. 4077–4090, 2025.
- [13] 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 *Forty-first International Conference on Machine Learning*, 2024.
- [14] S. Lee, J. Choi, S. Noh, J. Koo, and J. Kung, “Dbps: Dynamic block size and precision scaling for efficient dnn training supported by risc-v isa extensions,” in *2023 60th ACM/IEEE Design Automation Conference (DAC)*, 2023, pp. 1–6.
- [15] Y. Xu, Y. Lee, G. Yi, B. Liu, Y. Chen, P. Liu, J. Wu, X. Chen, and Y. Han, “Bitq: Tailoring block floating point precision for improved dnn efficiency on resource-constrained devices,” *CoRR*, vol. abs/2409.17093, 2024.
- [16] 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.
- [17] Z. Dong, Z. Yao, A. Gholami, M. Mahoney, and K. Keutzer, “Hawq: Hessian aware quantization of neural networks with mixed-precision,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 293–302.
- [18] Z. Yao, A. Gholami, K. Keutzer, and M. W. Mahoney, “Hessian-based analysis of large batch training and robustness to adversaries,” in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS’18, 2018, p. 4954–4964.
- [19] S.-H. Noh, J. Koo, S. Lee, J. Park, and J. Kung, “Flexblock: A flexible dnn training accelerator with multi-mode block floating point support,” *IEEE Transactions on Computers*, vol. 72, no. 9, pp. 2522–2535, 2023.
- [20] 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 *Proceedings of the 40th International Conference on Machine Learning*, ser. ICML’23, 2023.
- [21] W. Shao, M. Chen, Z. Zhang, P. Xu, L. Zhao, Z. Li, K. Zhang, P. Gao, Y. Qiao, and P. Luo, “Omniquant: Omnidirectionally calibrated quantization for large language models,” in *The Twelfth International Conference on Learning Representations*, 2024.
- [22] 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*, ser. ISCA ’23. Association for Computing Machinery, 2023.
- [23] Y. Chen, A. F. AbouElhamayed, X. Dai, Y. Wang, M. Andronic, G. A. Constantinides, and M. S. Abdelfattah, “Bitmod: Bit-serial mixture-of-datatype llm acceleration,” in *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2025, pp. 1082–1097.
- [24] 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 llm compression and acceleration,” in *MLSys*, 2024.
- [25] S. Li, Z. Yang, D. Reddy, A. Srivastava, and B. Jacob, “Dramsim3: A cycle-accurate, thermal-capable dram simulator,” *IEEE Computer Architecture Letters*, vol. 19, no. 2, pp. 106–109, 2020.
- [26] R. Balasubramonian, A. B. Kahng, N. Muralimanohar, A. Shafiee, and V. Srinivas, “Cacti 7: New tools for interconnect exploration in innovative off-chip memories,” *ACM Trans. Archit. Code Optim.*, vol. 14, no. 2, 2017.
- [27] 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)*, 2022, pp. 1414–1433.