Oltron: Software-Hardware Co-design for Outlier-Aware Quantization of LLMs with Inter-/Intra-Layer Adaptation

Chenhao Xue[1,4], Chen Zhang[2*], Xun Jiang[1], ZhuTianYa Gao[2], Yibo Lin[1,3,4], Guangyu Sun[1,3,4*]
[1]School of Integrated Circuits, Peking University. [2]Shanghai Jiao Tong University. [3]Institute of Electronic Design Automation, Peking University, Wuxi, China. [4]Beijing Advanced Innovation Center for Integrated Circuits.

Abstract

In Large Language Models (LLMs), outliers are identified by a small number of values with exceptionally high magnitudes, critically affecting model accuracy. Researchers have proposed several mixed-precision quantization techniques to manage these activation outliers. These approaches, employing value-wise outlier granularity, face challenges in balancing model accuracy with hardware efficiency. To address this issue, we capitalize on the observation that activation outliers of LLMs typically cluster within specific channels. Consequently, we introduce `Oltron`, a comprehensive software/hardware co-design strategy for outlier-aware quantization of LLMs with inter-/intra-layer adaptation. Our method includes three key innovations: firstly, a novel quantization algorithm that identifies the optimal ratio of outliers across different layers and channel groups within a layer; secondly, a reconfigurable architecture that adapts to inter- and intra-layer distributions; and thirdly, a tile-based dataflow optimizer that intricately arranges complex computations and memory access for mixed-precision tensors. `Oltron` outperforms the state-of-the-art outlier-aware accelerator, OliVe, achieving a 1.9x performance boost and 1.6x greater energy efficiency, while also enhancing model accuracy.

Keywords: Large Language Models, Quantization, Accelerators

ACM Reference Format:
Chenhao Xue[1,4], Chen Zhang[2*], Xun Jiang[1], ZhuTianYa Gao[2], Yibo Lin[1,3,4], Guangyu Sun[1,3,4*]. 2024. Oltron: Software-Hardware Co-design for Outlier-Aware Quantization of LLMs with Inter-/Intra-Layer Adaptation. In *61st ACM/IEEE Design Automation Conference (DAC '24), June 23–27, 2024, San Francisco, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3649329.3656221

1 Introduction

Transformer-based large language models (LLMs) have demonstrated great success [17]. However, their remarkable performance comes with exceptional model size and computational requirements. For instance, deploying the GPT-175B model for inference in FP16 consumes at least 350GB of storage, which requires at least 5 × 80GB A100 GPUs [1]. Quantization [2, 4, 5, 13, 18, 20–22] is one of the most effective ways to reduce the inference cost of LLMs. By quantizing the weights and activations into low bit-width, the demands for both memory size and bandwidth can be greatly reduced, and compute-intensive operations can be accelerated.

*The corresponding authors: Chen Zhang, chenzhang.sjtu@sjtu.edu.cn; Guangyu Sun, gsun@pku.edu.cn.

DAC '24, June 23–27, 2024, San Francisco, CA, USA

ACM ISBN 979-8-4007-0601-1/24/06…$15.00
https://doi.org/10.1145/3649329.3656221

Table 1. Comparison between outlier-aware accelerators and our proposed method `Oltron`.

Architecture		OLAccel [10]	OliVe [5]	`Oltron` **(Ours)**
Outlier Granularity		Value	Value	Channel
Accuracy	Outlier Bit-Width	16	4	8/12/16
Efficiency	Avg. Index Bit-Width	16	4	≈ 0
	Aligned Memory Access	✗	✓	✓
	Outlier PE Ratio	1/16	1	1/8
	Overhead Area Ratio	71%	0.2%	0.1%

Currently, the weights of LLMs can be quantized to 4-bit with minimal impact on model accuracy [3]. However, activation quantization of LLMs remains challenging, primarily due to outlier issues [2, 21]. The small fraction of activation values feature extremely large magnitudes and predominant impacts on model accuracy, which makes it difficult to apply low bit-width (4-bit) activation quantization without sacrificing model accuracy [13, 18, 20–22].

To maintain model accuracy, previous works have proposed outlier-aware quantization with dedicated hardware support, such as OLAccel [10] and GOBO [23]. These methods employ mixed-precision quantization, where low-bit (4-bit) quantization is applied to the majority of normal values, while high precision (16-bit) is reserved for the outliers. This approach achieves a nearly average 4-bit quantization with negligible accuracy loss. However, they treat outliers as sparse matrices, requiring additional codebooks for sparse format and complex control logic to handle irregular computations and memory accesses. OliVe [5], on the other hand, introduces outlier-victim pair (OVP) encoding with localized outlier storage, ensuring aligned memory access with minimal control overhead. Nevertheless, the localized outlier encoding scheme has limitations in extending the bit-width for outliers, resulting in insufficient precision in LLMs. Moreover, to locally process outlier values with random occurrence, OliVe augments all computing units with the capability to handle outlier values, which compromises hardware efficiency.

The aforementioned outlier-aware accelerators handle outliers at value-wise granularity, and struggle to balance model accuracy and hardware efficiency. In this work, we aim to strike such balance by handling activation outliers of LLMs at channel granularity. We leverage the observation that activation outliers are not distributed in complete randomness, but instead tend to cluster in specific channels [2, 20]. By handling outliers at coarser granularity, we can still maintain model accuracy with low average bit-width and efficient hardware design.

We introduce `Oltron`, a holistic quantization framework that encompasses algorithm-software-hardware co-optimization to effectively handle outlier activation channels while simultaneously

ensuring hardware efficiency. We commence by conducting a thorough analysis of outlier distribution patterns, shedding light on the channel-wise non-uniform distribution with intra-/inter-layer heterogeneity. To tackle intra-layer heterogeneity, we introduce the *Tile-wise Outlier-Balanced Encoding* (TOBE) format. TOBE consistently guarantees dataflow regularity and incurs minimal encoding costs through compilation-time optimizations. To address inter-layer heterogeneity, we co-design TOBE with an adaptive quantization algorithm and a reconfigurable architecture, which enables flexible adjustment of TOBE settings to meet the precision requirements of each layer, while allowing for efficient TOBE acceleration. Our evaluation results show that `Oltron` achieves superior excels in both quantized model accuracy and hardware efficiency. Compared to state-of-the-art outlier-aware accelerators, `Oltron`'s accelerator design surpasses existing outlier-aware accelerators, OLAccel [10] and OliVe [5], by 3.6× and 1.9× performance improvement, and 1.9× and 1.6× energy reduction, respectively.

The main contributions of this paper are as follows:

- We introduce Tile-wise Outlier-Balanced Encoding (TOBE) format for LLM activation quantization, which encodes outlier channels with high precision while consistently maintaining dataflow regularity.
- We propose `Oltron`, a TOBE-based quantization framework integrated with software-hardware co-optimization, to fully harness the adaptability of TOBE to the inter-/intra-layer heterogeneity of outlier channel distribution.
- We propose efficient implementation of `Oltron`'s reconfigurable architecture complemented by compilation & algorithm support, and demonstrate that the accuracy and efficiency of `Oltron` outperform existing outlier-aware quantization algorithms and hardware accelerators.

2 Background & Motivation

In this section, we first review previous works dealing with the activation outlier challenge of LLMs. Our proposed channel-wise outlier-aware quantization aims to balance accuracy with efficiency. However, given the non-uniform distribution of outlier channels detailed in Sec. 2.2, it remains challenging to maintain regular computation and memory access, and thereby, to provide hardware efficiency. To tackle this challenge, we introduce a software-hardware co-design solution termed as `Oltron`, and present its framework overview in Sec. 2.3.

2.1 Related Works

Previous works [5, 10, 13, 18, 20, 23] propose various quantization/architecture co-design methods to tackle the activation outlier challenge of LLMs. Some quantization algorithms [13, 18, 20] smooth out activation distribution by scaling down outlier magnitudes, so that the tensor can be uniformly quantized. However, encoding outlier values with the same low bit-width as normal values cannot provide enough precision, consequently leading to degraded model accuracy. In addition, some outlier-aware architectures [5, 10, 23] are proposed to encode outliers with high precision and maintain model accuracy. For instance, OLAccel [10] and GOBO [23] store high-precision outliers separately from normal values, and employ dedicated processing units for the outliers. However, the outlier storage has variable length due to random occurrence of outliers, which results in unaligned memory access, Moreover,

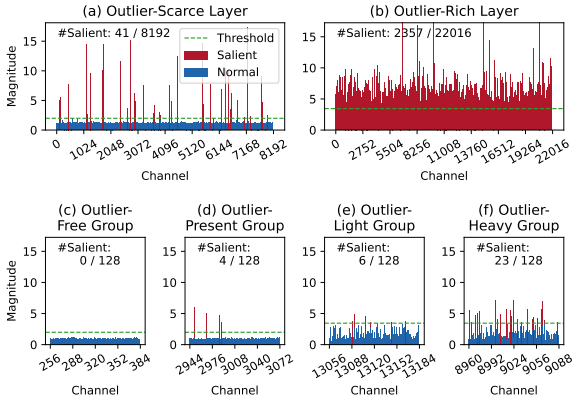


Figure 1. Input activation distribution of linear layers from LLaMA-65B. For illustration, channels of $> 2\times$ median magnitude are deemed *salient*. (a) and (b) are high-level views resized with max-pooling. (c-d) / (e-f) are zoom-in views of (a) / (b), respectively.

the separated outlier storage requires complex control logic to index outliers and orchestrate the computation between normal & outlier values. In contrast, OliVe [5] locally stores outlier values, and allocates co-located bit-width for indexing. OliVe manages to achieve aligned memory access and negligible control overhead. However, it can only extend limited bit-width to represent outlier values, which also suffers from insufficient precision and model accuracy loss. Moreover, in accommodation to outlier values with local storage and random occurrence, OliVe extends all computing units to enable outlier processing, which adversely impacts its hardware efficiency.

2.2 Outlier Distribution Characteristics

Fig. 1 visualizes the input activation distribution of two representative linear layers in the LLaMA-65B model. We randomly sampled 128 sequences from Wikitext2 dataset [8]. For every input channel, we recorded the maximum absolute value across all tokens. From these figures, we observe some intriguing characteristics of outlier distribution, namely: the presence of salient channels, and the inter-/intra-layer heterogeneity of outlier distribution.

Salient Channels refers to the ones with significantly larger magnitudes than most of other channels. The saliency of these channels is reflected in two aspects. Firstly, they tend to have larger quantization errors than normal channels under the same bit-width. Secondly, quantizing salient channels alongside normal channels results in either clamping of the former or substantial rounding errors in the latter, both resulting in model accuracy losses.

Inter-Layer Heterogeneity refers to the observation that different layers exhibit significantly different composition ratios of salient channels. For instance, as shown in Fig. 1(b), the MLP contraction layer contains a considerably higher number of outlier channels compared to the attention input projection layer depicted in Fig. 1(a). Therefore, we categorize the former as an *outlier-rich* layer and the latter as an *outlier-scarce* layer.

Intra-Layer Heterogeneity refers to the observation that salient channels are randomly distributed within a layer, resulting in various salient channel proportions across channel groups. In Fig. 1(c-f), significant distribution variances are evident among four distinct channel groups from the two selected layers, each comprising 128 channels. Some groups, such as those in Fig. 1(c), are entirely devoid of outliers, while others, like those in Fig. 1(f), exhibit a larger

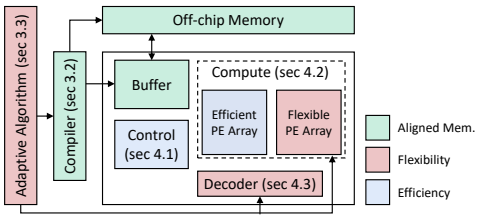


Figure 2. Oltron's framework overview.

proportion of salient channels, with the remaining groups falling somewhere in between.

2.3 Framework Overview

To address the efficiency challenges arised from inter-/intra-layer heterogeneity, we introduce a novel representation format called *Tile-wise Outlier-Balanced Encoding* (TOBE). TOBE supports encoding salient channels with arbitrary proportion and unrestricted precision, while maintaining storage regularity. To efficiently harness the flexible expressiveness of TOBE, we introduce Oltron, a versatile and reconfigurable quantization framework empowered with software-hardware co-optimization. As for software, Oltron utilizes an adaptive quantization algorithm to reduce mixed-precision overhead without compromising encoding precision, and leverages dedicated compilation support to achieve negligible encoding overhead. As for hardware, Oltron adopts an efficient accelerator design to support TOBE-based dataflow, with runtime reconfigurability to accommodate TOBE with various salient channel proportion & precision settings. The specific contents of the framework and their corresponding chapters are summarized in the Fig. 2

3 Software Design

In this section, we first present details of TOBE that balances the distribution of outlier channels. TOBE features the capability to keep computation and memory access regular without sacrificing outlier's high precision encoding. Since operators are connected through tensors, i.e. one operator's output is another operator's input, we propose a TOBE-aware dataflow optimization that optimizes the memory-related layout conversion operations across the whole computation graph. Finally, based on TOBE, we propose an adaptive quantization algorithm that automatically selects salient channel configurations to determine the best trade-off between mixed-precision overhead and model accuracy.

3.1 Tile-wise Outlier-Balanced Encoding

To execute large tensor operators in LLMs, tiling is a must to split computations and associated input tensors into sub-blocks, allowing efficient caching in on-chip memory. However, mixed-precision quantization may not integrate well with the tiling approach. To encode the mixed-precision tensor, a straightforward approach is to store outliers in their original positions (Fig. 3(a)). As outliers require additional encoding bit-width, this method results in irregular tensor storage and inconsistent data tile sizes. Such imbalance leads to unaligned memory access and under-utilization of on-chip resources, and ultimately reduced achievable performance. The split scheme [10, 23] addresses the memory alignment challenge by separating sparse outlier encoding to ensure uniformity of normal tiles (Fig. 3(b)), yet the problem persists for the variable-length outlier encoding. It also introduces complicated control overhead to manage the separately stored and computed normal/outlier values. To achieve memory alignment with reduced control overhead, the outlier-victim pair encoding [5] improves upon the original scheme.

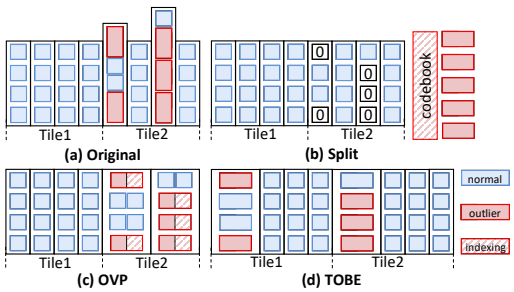


Figure 3. Comparisons between different encoding format.

It prunes the victim values adjacent to the outliers, and allocates the saved bit-width for extended outlier representation (Fig. 3(c)). However, this approach only offers limited bit-width expansion for outliers, resulting in constrained precision of outlier representation, and ultimately impacting the model's accuracy.

TOBE achieves regular memory access pattern by adjusting precision at channel granularity, and reordering high precision channels to ensure balanced sizes of tiled sub-blocks, which greatly reduces the complicated control logic required for the sparse outlier values (Fig. 3(d)). As opposed to handling outlier values individually in previous encoding schemes, TOBE encodes salient channels with severe outlier issues at high precision. Regarding the observed intra-layer random distribution of salient channels (Fig. 1), to achieve balanced tile sizes, TOBE evenly distributes salient channels into tiled sub-blocks by reordering the channel permutation. Since on-chip buffer accesses off-chip memory in units of data tiles, TOBE's uniform data tile sizes ensure regular off-chip memory access. In addition, within each tiled sub-block, the assigned salient channels are placed at the forefront. The determined and consistent data layout of tiled sub-blocks ensures that on-chip memory access and computation can proceed in a regular manner.

Oltron mainly applies TOBE on input activation tensors of linear layers that exhibit significant outlier problems. To guarantee the correctness of the matrix multiplication results, given the reordered channel permutations of activation tensors, we also need to reorder the input channels of weight tensors correspondingly, as shown in Fig. 4(a). As the salient channels are selected based on their extreme magnitudes on calibration data, the channel permutations are predetermined. Therefore the rows of weight tensors can be permuted statically before deployment.

3.2 Dataflow Optimization

TOBE reorders the tensor layout tile-wise in memory to enhance intra-layer computation efficiency. However, this encoding necessitates an explicit reordering operation to prepare the data because the output of the previous operator is not inherently encoded in TOBE, as shown in Fig. 4. To mitigate the overhead introduced by the reordering operation, we propose dataflow optimization with two strategies applied to the computation graph, indicated by circled numbers ❶-❹ in Fig. 4.

First, we leverage the inherent commutative characteristics in matrix multiplications. In a $\langle M \times N \times K \rangle$ matrix multiplication, the reordering operation can be directly applied to the previous layer's output. For example, reordering rows in the M dimension or columns in the N dimension does not influence the final results. Moreover, if the layers between the two matrix multiplication layers also exhibit commutative characteristics, such as element-wise operators (activation layers), this reordering strategy can propagate

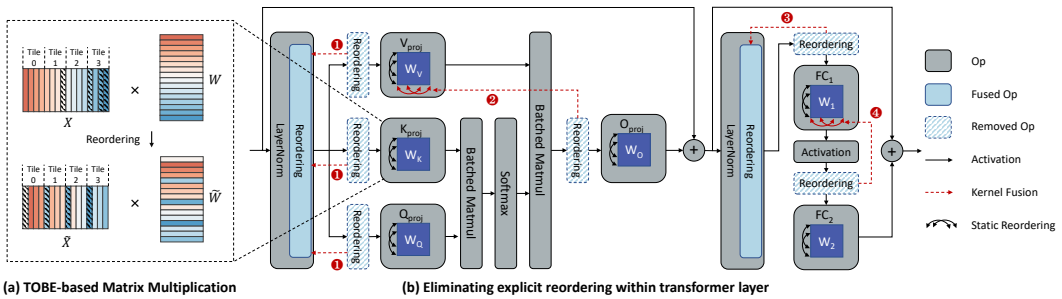


Figure 4. TOBE-based computation workflow. (a) Linear layers adopt TOBE-based matrix multiplication. (b) The encoding overhead of explicit run-time reordering is mitigated with graph-level compilation-time optimization.

Algorithm 1 Adaptive Quantization Algorithm

Input: activation statistics $\mathcal{M} = \{M \in \mathbb{R}^{\#\text{batch} \times d_l} | 1 \le l \le L\}$, storage budget $B^* \in \mathbb{R}$;
Output: salient channel sets $\mathcal{S} = \{S_l | S_l \subseteq \{1, 2, \cdots d_l\}, 1 \le l \le L\}$, layer-wise salient channel data types $\vec{t} \in \mathcal{T}^L$;

for layer $l \in \{1, 2, \cdots, L\}$ **do**
  $\tau_l \leftarrow 3 \times$ Standard-Deviation(M_l)
  $t_l \leftarrow$ FP8
while target budget B^* not reached **do**
  $\langle e, B \rangle \leftarrow$ Estimate-MSE-And-Budget$(\mathcal{M}, \vec{\tau}, \vec{t})$
  $i \leftarrow -\infty$
  $T \leftarrow$ Considered-Modification$(\vec{\tau}, \vec{t})$
  for $\langle \vec{\tau}', \vec{t}' \rangle \in T$ **do**
    $\langle e', B' \rangle \leftarrow$ Estimate-MSE-And-Budget$(\mathcal{M}, \vec{\tau}, \vec{t})$
    $i' \leftarrow$ Estimate-Improvement$(e - e', B - B')$
    if $i' > i$ **then**
      $\langle i, \vec{\tau}, \vec{t} \rangle \leftarrow \langle i', \vec{\tau}', \vec{t}' \rangle$
$\mathcal{S} \leftarrow$ Select-Salient-Channels$(\mathcal{M}, \vec{\tau})$
Return $\mathcal{S}, \vec{t}$

through them. Two cases, layers Q_{proj} (❷) and FC_2 (❹), illustrate situations where we can eliminate explicit reordering operators by statically permuting the columns of previous layers' weight tensors.

Second, we use a kernel fusion method to merge the reordering operation with the previous layer. For $Q_{proj}/K_{proj}/V_{proj}$ (❶) and FC_1 (❸), we fuse them into previous LayerNorm operators, where each output channel is redirected to a different address in the off-chip memory to match the TOBE format of the next layer's output.

We implement this dataflow optimization in the framework's compilation pass. We traverse the entire computation graph, examining every pair of linear layers (e.g., FC1 and FC2). We also analyze the operators directly connected between them (e.g., activation layers) and apply the two aforementioned strategies to eliminate any additional memory access caused by explicit reordering operations.

3.3 Adaptive Quantization Algorithm

`Oltron` adopts mixed-precision quantization at the granularity of channels. In this approach, data within each channel is encoded in either low-precision or high-precision format. Normal channels use a 4-bit uniform encoding. For salient channels, we allow the algorithm to automatically select from multiple high-precision data type candidates, including `fp8`, `fp12`, and `fp16`.

Our algorithm automatically determines two key factors: (1) the composition ratio of salient channels relative to normal channels and (2) the most suitable bit-width for salient channels. It begins with an initial budget parameter, denoted as $\langle B^* \in \mathbb{R} \rangle$, and a set of parameters $\langle \vec{\tau} \in \mathbb{R}^L, \vec{t} \in \mathcal{T}^L \rangle$. The budget B represents the target averaged bit-width of the mixed-precision models. We use $\vec{\tau}$ to classify values as outliers or normal values, which is initialized based on an empirical 3σ rule [19], and $\vec{t}$ to specify the data types to be used for high-precision channels. With these inputs, our algorithm tunes the $\langle \vec{\tau}, \vec{t} \rangle$ parameters to control the ratios of outliers to normal values in each layer. In each iteration, we determine the remaining budget by comparing the current average bit-width B with the target budget B^*. Based on this comparison, we either assign more channels as outliers or recall outliers back. We determine the changes to the outlier ratio in each layer by calculating the mean squared error (MSE) e between the original and post-change values. The algorithm continues iterating until it reaches the target bit-width B^*.

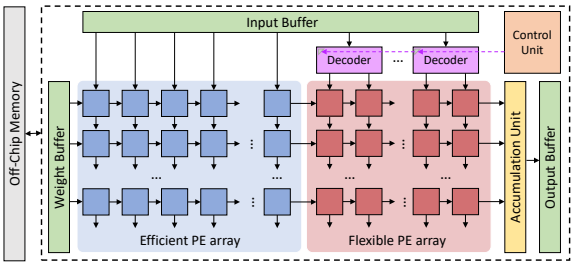


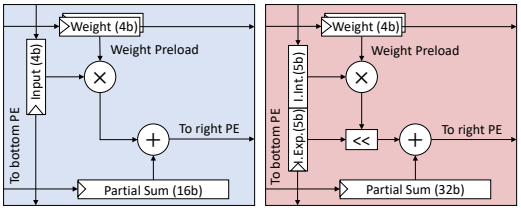
Figure 5. Overview of `Oltron` architecture.

4 Architecture Design

In this section, we present `Oltron`'s efficient and reconfigurable architecture. Our hardware design efficiently supports TOBE-based dataflow by leveraging the regularity merit of TOBE, and incorporates reconfigurable components to flexibly accommodate various TOBE settings.

4.1 Architecture Overview

Fig. 5 presents the overview of `Oltron`'s architecture. It consists of a weight-stationary systolic array with hybrid PE designs, on-chip buffers for input/weight/output, an accumulation unit, a control unit, and decoders. `Oltron` architecture adopts a 4-stage pipeline similar to Google's TPU [6]: **(1) Reading Off-chip Memory:** The weight/input buffer loads outlier-balanced data tiles from off-chip memory. **(2) Preloading:** Given the salient channel configuration of the input tile, weight tiles are preloaded accordingly, and the control unit propagates control signals to decoders. **(3) Matrix Multiplication:** Data of input/output channels flows in the corresponding PE array columns/rows, respectively. The Multiply-And-Accumulate (MAC) operations for normal and outlier values can be concurrently and seamlessly executed on-chip, since both normal and outlier input values utilize the same quantization scale, and their multiplication results are stored in a unified integer format. In this sense, `Oltron` does not require complex control logic



(a) Efficient PE **(b)** Flexible PE

Figure 6. Oltron PE design.

to manage normal & outlier value computation. **(4) Writing Off-chip Memory:** Finally, the output buffer writes the accumulated output tile back to off-chip memory.

4.2 PE design

Oltron's architecture utilizes a hybrid-PE design, which features the majority of PEs with a simplistic design to enhance efficiency, and the minority of PEs with a flexible design to provide adaptivity. The **Efficient PEs** (Fig. 6a) can only support signed int4×int4 multiplication with limited accumulation bit-width (16-bits), resulting in better energy efficiency and reduced area. The **Flexible PEs** (Fig. 6b) are modified atop the efficient PE design for better flexibility. First, we augment the 4-bit multiplier to support both signed and unsigned input operands. The original 4-bit input integer register is extended with an extra sign bit to accommodate the modification. Second, to support int×fp multiplication, we add a shifter controlled by a 5-bit input exponent. Finally, we extend the accumulation bit-width to 32 to capture the highly dynamic range of fp inputs. With the above modification, each flexible PE can support int4×fp8 (E5M2), and multiple flexible PEs can be composed to support fp inputs with more mantissa bits [15].

4.3 Decoder Design

Oltron employs a reconfigurable decoder design to support the processing of a mixture of data types, including int4, fp8, and fp12. Our current decoder design supports up to 12-bit fp precision, as we find that fp12 can achieve comparable accuracy results to those of fp16 on the tested LLMs. However, it is feasible to extend current design to support higher fp precision if necessary.

With a specified data type, the decoder accesses on-chip buffer accordingly. It loads 1 byte (two int4) or 2 bytes (two fp8 or one fp12 padded with 4 bits) when accessing on-chip buffer (Fig. 7(a)). The decoder also converts the fetched data into a unified exponent-integer pair format to simplify subsequent computation. For int4, the decoder simply fills the exponent fields with zeros, and the integer fields with the input values (Fig. 7(b)). For fp, it decodes each input value into exponent-integer pair with equation Eq. (1):

$$\texttt{sign} \times (1 << \texttt{mb} + \texttt{mantissa}) << (\texttt{exponent} - \texttt{bias}) \quad (1)$$

where mb is the mantissa bit-width. To calculate the effective exponent, the constant bias is properly selected to avoid any fractions in integer fields, and overflow-proof subtraction module is adopted to avoid unexpected extremely large values. To calculate the integer, it requires 4 bits for fp8 with $mb = 2$, or 8 bits for fp12 with $mb = 6$ to represent the result. The converted exponent-integer pair of fp8 can fit into the activation buffer of a single flexible unit (Fig. 7(c)); while that of fp12 cannot. Therefore, we decompose fp12's exponent-integer pair into two sub-pairs, and use two flexible PEs to collaboratively execute MAC operations (Fig. 7(d)).

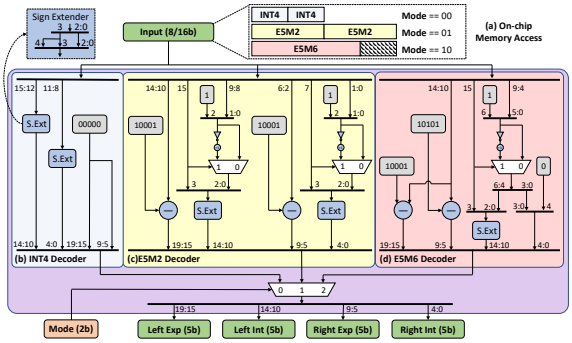


Figure 7. Oltron Decoder Design.

5 Evaluation

In this section, we evaluate Oltron in terms of model accuracy, performance, and energy efficiency.

5.1 Methodology

Quantization Setup Our method is evaluated on LLaMA(7B-65B) [16] and OPT(6.7B-66B) [24] families. We evaluate the perplexity of language generation on WikiText2 [8] and C4 [11] datasets.

Quantization Baseline We compare Oltron with existing weight-activation quantization works, including OmniQuant [13] and OliVe [5]. For Oltron, we quantize weight tensors with GPTQ [3]. All the PTQ quantization methods use 128 randomly sampled calibration sequences from WikiText2 [8].

Accelerator Baseline We compare the performance and energy of Oltron against two outlier-aware DNN quantization accelerators, including OLAccel [10] and OliVe [5]. We use quantized model with perplexity close to original as benchmark workload.

Architecture Implementation We implement the Oltron PE and decoder described in Sec. 4 with the Verilog RTL. We use Synopsys DC [7] and TSMC-28nm PDK to synthesize the designs and estimate the area, latency, and power. We use CACTI [9] to estimate the area and power of memories. We develop a cycle-level simulator based on DnnWeaver [14] to estimate the overall performance. We use DeepScaleTool [12] to scale all designs to the same process for iso-area comparison.

5.2 Accuracy Result

We first evaluate the quantized model and report the perplexity. We focus on 4-bit quantization, since near-lossless accuracy is achieved with higher bit-width in previous works [13, 20]. As shown in Tbl. 2, Oltron outperforms existing methods on various models. OliVe can only extend limited bit-width to encode outlier values, which leads to significant accuracy losses. OmniQuant smoothes activation distribution to encode all values with int4, which achieves comparable accuracy with Oltron on small models, but falls short in maintaining accuracy on larger models. The comparative results indicate that Oltron's capability to maintain high-precision outlier quantization can effectively preserve LLMs' model accuracy. We also report the perplexity results of Oltron with uniform TOBE configuration across all layers. Under the same storage budget, the adaptive quantization algorithm described in Sec. 3.3 consistently outperforms the uniform counterpart.

5.3 Accelerator Performance and Power

Area We compare the accelerator area breakdown in Tbl. 3. All the accelerators use the same on-chip buffer configuration and similar core area. Oltron integrates more PEs within similar core area than

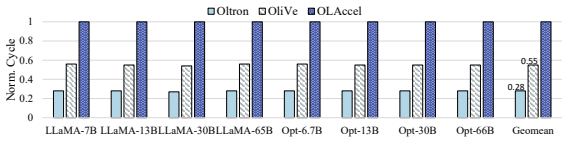
Table 2. Perplexity results of quantized model on Wikitext2 and C4 datasets (lower is better).

Model/PPL↓		LLaMA-7B		LLaMA-13B		LLaMA-30B		LLaMA-65B		OPT-6.7B		OPT-13B		OPT-30B		OPT-66B	
Method	A bits	WIKI	C4	WIKI	C4	WIKI	C4	WIKI	C4	WIKI	C4	WIKI	C4	WIKI	C4	WIKI	C4
FP16	16	5.68	7.08	5.09	6.61	4.10	5.98	3.53	5.62	10.86	11.74	10.12	11.19	9.56	10.69	9.34	10.28
Olive	4	144.78	117.49	42.24	43.13	36.55	33.78	1.4e7	1.8e7	107.15	61.24	416.57	994.64	334.7	572.02	4058.83	2926.87
Omniquant	4	11.26	14.51	10.87	13.78	10.33	12.49	9.17	11.28	12.24	13.56	11.65	13.46	10.60	11.89	10.29	11.35
`Oltron`*	4.01	126.81	92.50	185.50	170.75	164.97	357.00	30.61	45.73	16.63	16.26	13.41	14.66	11.20	12.68	151.48	190.71
`Oltron`	4.01	36.47	44.62	144.08	100.18	439.25	131.15	15.85	20.85	12.69	13.58	**11.49**	**12.53**	10.72	**11.87**	11.61	11.71
`Oltron`*	4.1	14.47	16.80	**9.48**	**12.42**	**7.51**	**9.42**	**6.69**	**9.41**	**11.99**	**13.04**	**11.61**	**12.42**	10.64	**11.67**	10.50	**11.09**
`Oltron`	4.1	11.67	15.21	**8.20**	**10.84**	**6.68**	**8.65**	**5.82**	**8.19**	**12.00**	**13.02**	**11.35**	**12.27**	**10.51**	**11.63**	10.49	**11.05**

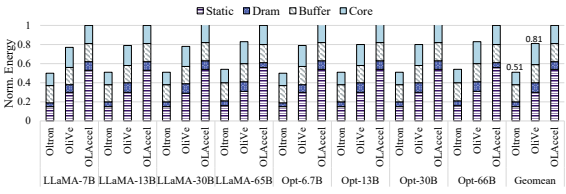
* Use the same salient channel configuration across all layers.

Table 3. The configuration and area breakdown of `Oltron` and other baselines.

Architecture	PE Number	Core Area	Buffer
`Oltron`	4096	0.322 mm^2	512 KB 4.2 mm^2
OliVe	2048	0.318 mm^2	
OLAccel	1152	0.320 mm^2	



(a) Normalized Latency



(b) Normalized Energy

Figure 8. Comparison of different accelerator designs.

OliVe, since the synthesized area of OliVe's PE is ≈ 2× larger than `Oltron`'s efficient PE design. Similar to OliVe, `Oltron`'s decoding logic placed alongside the array incurs negligible area overhead. While the complex control logic of OLAccel takes up a large portion of the total area, which constrains computational power.

Performance Fig. 8a shows the normalized latency of different designs on various models. Leveraging the higher computational power and reconfigurability, `Oltron` exhibits superior performance than the baseline designs. On average, `Oltron` achieves 1.9× and 3.6× speedup values over OliVe and OLAccel, respectively.

Energy Fig. 8b shows the normalized energy of different designs. We collect the static energy and dynamic energy (DRAM, on-chip buffer, and core). Compared with the baseline designs, `Oltron` consumes the least static energy due to better performance. `Oltron` also has lower dynamic energy of cores, owing to the simple design of the majority of PEs. Overall, `Oltron` achieves 1.6× and 1.9× energy reduction values over OliVe and OLAccel, respectively.

6 Conclusion

In this paper, We propose `Oltron`, a holistic outlier-aware quantization framework for accelerating LLMs. The key insight is to encode outlier channels at high precision while maintaining representation regularity unaffected by non-uniform outlier channel distribution. For intra-layer heterogeneity, we propose Tile-wise Outlier-Balanced Encoding (TOBE) with consistent regular memory access and computation pattern. For inter-layer heterogeneity, we propose `Oltron`, which integrates adaptive quantization algorithm and reconfigurable hardware to accommodate each layer's precision requirement with minimum overhead. `Oltron` pushes the limit of LLM post-training quantization to a new state-of-the-art. Moreover, `Oltron`'s design surpasses existing outlier-aware accelerator, OliVe, by 1.9 × performance improvement and 1.6× energy efficiency improvement.

Acknowledgments

This work is supported by National Natural Science Foundation of China (Grant No. 62032001) and 111 Project (B18001).

References

[1] Tom Brown et al. 2020. Language models are few-shot learners. *NIPS* 33 (2020), 1877–1901.
[2] Tim Dettmers et al. 2022. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339* (2022).
[3] Elias Frantar et al. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323* (2022).
[4] Cong Guo et al. 2022. Ant: Exploiting adaptive numerical data type for low-bit deep neural network quantization. In *MICRO*. IEEE, 1414–1433.
[5] Cong Guo et al. 2023. OliVe: Accelerating Large Language Models via Hardware-friendly Outlier-Victim Pair Quantization. In *ISCA*. 1–15.
[6] Norman P Jouppi et al. 2017. In-datacenter performance analysis of a tensor processing unit. In *ISCA*. 1–12.
[7] Pran Kurup et al. 2012. *Logic synthesis using Synopsys®*. Springer Science & Business Media.
[8] Stephen Merity et al. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843* (2016).
[9] Naveen Muralimanohar et al. 2009. CACTI 6.0: A tool to model large caches. *HP laboratories* 27 (2009), 28.
[10] Eunhyeok Park et al. 2018. Energy-efficient neural network accelerator based on outlier-aware low-precision computation. In *ISCA*. IEEE, 688–698.
[11] Colin Raffel et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J Mach Learn Res* 21, 1 (2020), 5485–5551.
[12] Satyabrata Sarangi et al. 2021. DeepScaleTool: A tool for the accurate estimation of technology scaling in the deep-submicron era. In *ISCAS*. IEEE, 1–5.
[13] Wenqi Shao et al. 2023. Omniquant: Omnidirectionally calibrated quantization for large language models. *arXiv preprint arXiv:2308.13137* (2023).
[14] Hardik Sharma et al. 2016. Dnnweaver: From high-level deep network models to fpga acceleration. In *COGARCH*.
[15] Hardik Sharma et al. 2018. Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural network. In *ISCA*. IEEE, 764–775.
[16] Hugo Touvron et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).
[17] Ashish Vaswani et al. 2017. Attention is all you need. *NIPS* 30 (2017).
[18] Xiuying Wei et al. 2023. Outlier Suppression+: Accurate quantization of large language models by equivalent and optimal shifting and scaling. *arXiv preprint arXiv:2304.09145* (2023).
[19] Wikipedia contributors. 2022. 68–95–99.7 rule — Wikipedia, The Free Encyclopedia. [Online].
[20] Guangxuan Xiao et al. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *ICML*. PMLR, 38087–38099.
[21] Zhewei Yao et al. 2022. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *NIPS* 35 (2022), 27168–27183.
[22] Zhihang Yuan et al. 2023. RPTQ: Reorder-based Post-training Quantization for Large Language Models. *arXiv preprint arXiv:2304.01089* (2023).
[23] Ali Hadi Zadeh et al. 2020. Gobo: Quantizing attention-based nlp models for low latency and energy efficient inference. In *MICRO*. IEEE, 811–824.
[24] Susan Zhang et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068* (2022).