Integer Unit-based Outlier-Aware LLM Accelerator Preserving Numerical Accuracy of FP-FP GEMM

Jehun Lee
Seoul National University
Seoul, Korea
jehun.lee@snu.ac.kr

Jae-Joon Kim
Seoul National University
Seoul, Korea
kimjaejoon@snu.ac.kr

Abstract—**The proliferation of large language models (LLMs) has significantly heightened the importance of quantization to alleviate the computational burden given the surge in the number of parameters. However, quantization often targets a subset of a LLM and relies on the floating-point (FP) arithmetic for matrix multiplication of specific subsets, leading to performance and energy overhead. Additionally, to compensate for the quality degradation incurred by quantization, retraining methods are frequently employed, demanding significant efforts and resources. This paper proposes OwL-P, an outlier-aware LLM inference accelerator which preserves the *numerical* accuracy of FP arithmetic while enhancing hardware efficiency with an integer (INT)-based arithmetic unit for general matrix multiplication (GEMM), through the use of a shared exponent and efficient management of outlier data. It also mitigates off-chip bandwidth requirements by employing a compressed number format. The proposed number format leverages outliers and shared exponents to facilitate the compression of both model weights and activations. We evaluate this work across 10 different transformer-based benchmarks, and the results demonstrate that the proposed integer-based LLM accelerator achieves an average 2.70× performance gain and 3.57× energy savings while maintaining the numerical accuracy of the FP arithmetic.**

Index Terms—**floating-point computation, transformer, accelerator, outlier**

I. Introduction

Transformer-based models are showing remarkable results in various Natural Language Processing (NLP) tasks [1]. The success of Large Language Models (LLMs) [2], [3] has accelerated the trend towards scaled-up models, demanding more computational power and massive memory footprint for inference. In response, quantization emerged as a promising compression method, leading to the proposal of hardware-friendly quantization approaches [4], [5].

However, conventional quantization methods have limitations. These methods often require retraining to ensure inference quality, or they are selectively applied only to specific layer subsets, such as attention mechanism, feed-forward network (FFN), query, key and value (QKV) generation [6]. In case of transformer decoder-based LLMs, the auto-regressive nature of these models amplifies the cumulative computation errors, which complicates the quantization of activation values [7]–[9]. Consequently, there is no guarantee that conventional quantization methods can always be applied to rapidly evolving models. These limitations leave many accelerator products, such as Google TPUv4i, AMD XDNA and Intel Meteor Lake neural processing unit (NPU), employing floating-point (FP) arithmetic units [10]–[12].

TABLE I
Comparison between OwL-P and other methods

Data Format	Arithmetic	Numerical Accuracy (compared to FP)
FP	FP	FP
INT quantization	INT	Heavy approximation
INT quantization + FP outliers	INT+FP$^{(1)}$	Heavy approx for normal values + FP outliers
Block FP	INT+$\alpha^{(2)}$	Light approximation
OwL-P (ours)	**INT+$\alpha^{(2)}$**	**Same as FP**

FP indicates floating-point.
(1) - Dedicated unit for FP outlier, (2) - small extra logic.

Table I shows the data format, arithmetic types and numerical accuracy of computation results for various methods aimed at accelerating LLMs. The FP number format has the advantage of representing a wide range of numbers, but it typically requires complex arithmetic units and incurs computation overhead. The quantization using the integer (INT) format and arithmetic allows users to use simple integer arithmetic units, but it suffers from accuracy degradation due to heavy approximation of FP computations [13]–[18]. Recently, the INT + FP outlier data format has been gaining interests for LLM quantization. By quantizing most of the values using integer format and handling small number of outlier values with FP format, it can increase the quality of LLM outcomes [19], [20]. However, dedicated FP computation units for outliers impose a hardware overhead. Additionally, computation results still suffer from numerical errors which are mostly incurred by the normal values that are quantized. Meanwhile, the block FP format utilizes shared exponents so that it can use the integer arithmetic with small extra logic, thereby reducing hardware overhead [4], [21], [22]. However, the block FP results are still an approximation of FP results. We propose a method to compute and compress conventional FP data by utilizing the shared exponents and outlier values to use simpler INT-based arithmetic units while preserving the numerical accuracy of FP-FP general matrix multiplication (GEMM).

Our proposed method, OwL-P (INT unit-based Outlier-aware LLM accelerator preserving numerical accuracy of FP-FP GEMM), offers a bullet-proof design that ensures the same numerical accuracy and network accuracy as the FP computations while using INT units only. The rationale behind the proposed method is that the exponent values of the weight and activation in LLMs typically have a very narrow distribution, except for a small portion of outliers [19], [23].

Based on the characteristics, our proposed scheme expresses most of the exponents in FP values with a shared exponent and small bias bits, thereby achieving data compression. Unlike previous outlier-aware schemes and adaptive FP quantization schemes [4], [5], [23]–[25], our approach does not involve any bit truncation of mantissa, thus preserving the numerical accuracy of the conventional FP computation. At the same time, it utilizes the INT arithmetic-based hardware without using dedicated hardware for FP, minimizing hardware overhead. Both normal and outlier multiplications are performed within the same processing element (PE), thereby increasing the system performance and efficiency. To the best of our knowledge, the proposed OwL-P is the first accelerator that uses INT unit to compute inference of LLM models while preserving the numerical accuracy of FP-FP GEMM.

The main contributions of this paper are summarized as follows:

- We propose an INT unit-based PE array to accelerate LLM applications. Unlike previous designs based on INT units, our proposed design preserves numerical accuracy of FP multiplication and accumulation (MAC) computations. This feature allows users to run inference on network models trained in FP format using efficient INT-based hardware, while maintaining LLM quality score without any degradation.
- We propose a number format that utilizes a shared exponent along with a small number of outliers based on the statistical distribution of exponents. This format reduces memory footprint and enhances performance in memory-bound LLM computation scenarios.
- We propose an outlier scheduling scheme that enables the computation of both outlier and normal values in the same INT-based unit. With this scheduling scheme, the number of outlier paths in our PE can be minimized, resulting in negligible hardware overhead.

II. Challenges and Motivation

A. Limitations in Prior Quantization Methods

Many existing quantization methods aim to represent the wide data range in LLMs with reduced bit precision. It has been observed that INT quantization of LLMs often struggles to maintain network accuracy, primarily due to difficulties in quantizing activations [20], [26]. As an alternative, weight-only quantization, which quantizes the weight to INT values while keeping FP values for activations, has gained interest for LLM quantization. However, this method often requires dequantization of INT weights and fallback to FP-FP computations. Another well-known method is the microscaling scheme based on block FP format with shared exponent [4], [5]. While this scheme alleviates the challenges of INT quantization for covering wide range of values, it still cannot guarantee the same level of numerical accuracy as FP computations, since it is still an approximation of FP computations. Particularly, a fraction of weight and activation values with much larger magnitudes than others significantly affect inference quality in LLMs [19]. These outliers pose challenges for the block FP approach, which struggles to accommodate the entire range of exponents due to outliers.

TABLE II
Ratio of normal values in transformer-based networks.

	BERT -Base	BERT -Large	GPT2 -Base	GPT2 -Large	Llama2 -7B	Llama2 -70B
Weight	98.5	98.6	98.2	98.4	98.4	98.6
Activation	96.6	97.9	96.8	97.3	97.6	97.8

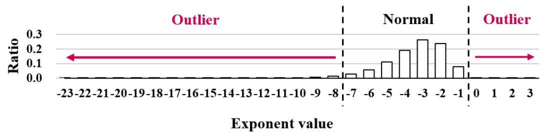


Fig. 1. Exponent distribution for FFN weights in the first layer of GPT2-Base. The red arrow indicates outlier values.

In this study, we aim to address the challenges by representing most exponent values using a shared exponent with small number of bias bits, and by covering the rest of values using outlier with original FP format. However, implementing separate INT and FP arithmetic units would incur substantial hardware overhead. Therefore, we developed an INT unit with a built-in outlier path that can handle both block FP normal values and FP outliers using the same unit. Implementation details will be explained in Section IV.

B. Exponent Distribution in Neural Networks

In our analysis of exponent occurrences in weight from the FFN in the first layer of GPT2-Base [2], we observed that the seven most common exponents that are consecutively distributed account for the majority of occurrences as shown in Fig. 1. We refer to these values as normal values and the values outside the range as outliers in the rest of the paper. The normal values account for 98.4% of the total weight exponent values and are continuously distributed in GPT2-Base. As shown in Table II, the characteristics are also observed in other transformer-base networks [1]–[3]. This property enables the use for a compressed number format to represent normal values with small exponent differences, which is important for both energy efficiency and performance.

III. Lossless Compression of Model

A. Proposed OwL-P Number Format

Fig. 2 provides a comparison between general FP and the OwL-P number formats used to represent values in tensor. The compressed data based on OwL-P number format alleviates the pressure on data bandwidth due to the smaller bit precision. Fig. 2(a) shows the conventional BF16 data format which is described as the following equation.

$$\text{BF16} : (-1)^{sign} \times 2^{(exponent-127)} \times 1.frac \quad (1)$$

Fig. 2(b) shows our proposed number format that includes a exponent shared by each subset tensor within each layer of a DNN. This format includes a sign bit, 3-bit exponent bias, and 7-bit fraction to maintain the information of the original FP format. Since the differences across the exponent values of the normal values within a tensor are less than 7, the exponents can be represented using only 3-bit bias with a shared base

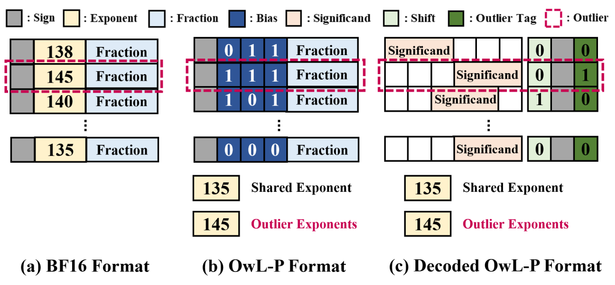


Fig. 2. (a) conventional BF16 format, (b) proposed OwL-P format and (c) decoded OwL-P format. The bias decoder aligns the position of significands based on two LSB bits in bias.

exponent. The proposed OwL-P number formats for normal and outlier values can be described as follows.

$$\text{OwL-P}\begin{cases} Normal : (-1)^{sign} \times 2^{(shared\_exp-127)+bias} \times 1.frac \\ Outlier : (-1)^{sign} \times 2^{(outlier\_exp-127)} \times 1.frac \end{cases} \quad (2)$$

We leave bias=3'b111 as an indicator for the outlier. The 8-bit outlier exponent values are stored separately from normal values as in Fig. 2(b). As a result, it is possible to achieve compression of exponent bits while preserving the information in the BF16 format without any loss of information.

B. Bias Decoding Scheme

To process the proposed format within the INT unit-based PE array, the decoder converts the data format into aligned integer format before feeding the data into PEs, as illustrated in Fig. 2(c). The decoder first determines whether the data is a normal or outlier value based on the bias bits. The outlier tag bit indicates whether the decoded value is an outlier. Subsequently, the significand of normal values is aligned based on the bias values according to Algorithm 1. To reduce the overhead of the 3-bit variable shifting, the decoder shifts the significand values using only 2 LSBs out of the 3-bit bias value and generates a shifting bit to conduct the MSB-dependent shifting in PE array. A detailed explanation of the shift bit is provided in Sec. IV-B. To preserve numerical accuracy, our design streams in the pre-aligned integer data, passing the entire significand values from the decoder to the PE.

Algorithm 1 Bias decoding algorithm.

```
Require: Sign, $s$; 3-bit Bias, $b_2b_1b_{0(2)}$; Significand, $sig$;
Ensure: Pre-aligned INT, $p$; Shift Bit, $sh$; Sign, $sign$; Outlier Tag, $tag$;
  $Def\ decode(sign, b_2b_1b_{0(2)}, sig)$ :
  if $b_2 \& b_1 \& b_0$ then
      $p = \{(3'b000, sig)\}$;   // outlier.
      $sh = 1'b0$;
      $tag = 1'b1$;
      return $\{p, sh, sign, tag\}$;
  else
      $p = sig \ll b_1b_{0(2)}$;   // normal.
      $sh = b_2$;
      $tag = 1'b0$;
      return $\{p, sh, sign, tag\}$;
  end if
```

IV. The Proposed OwL-P Architecture

A. Overall Architecture

We introduce a unified INT-based PE that manages both normal and outlier data within the same PE array to address the scheduling difficulties and increased area/energy overhead encountered when using dedicated FP-based PEs for outliers. Fig. 3(a) illustrates the overall architecture of the OwL-P processor, which includes INT-based systolic arrays operates in a weight-stationary fashion. At the beginning of computation, the bias decoder and outlier scheduling unit convert the data chunk from the buffer into aligned integer values for computation.

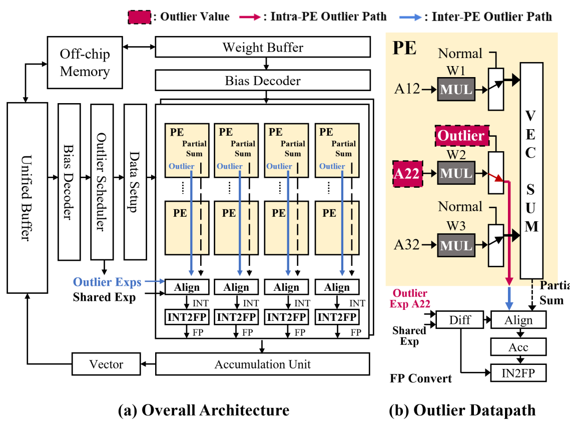


Fig. 3. (a) Overall hardware architecture of OwL-P. Outliers and normal partial sum values are propagated downward via separate paths. (b) Multiplication results of normal values are directed to the vector sum block, while results associated with outliers are bypassed through the intra-PE outlier path.

Fig. 3(b) shows an example of the operating principle of the proposed INT-based PE. The PE performs the dot-product of the decoded OwL-P format, regardless of whether the values are outliers or normal. Therefore, there is no need for dedicated MAC units to handle the outliers as in previous works [17], [18], [27]. When one of multiplication operand is outlier, then result is identified as outlier result. The outlier multiplication result is not fed to the vector sum block. Instead, the outlier multiplication result is bypassed via the intra-PE outlier path while the normal multiplication results are accumulated into partial sum in the vector sum block.

The outlier computations results of each PEs are propagated downward through vertical inter-PE outlier paths. The bypassed outlier multiplication result is aligned with the accumulated partial sum at the bottom of a PE column with proper shift information based on the exponent, and converted into FP at the INT-to-FP (INT2FP) unit. This ensures that the proposed PE produces the same result as the conventional FP MAC operation without any approximation. The vector unit encodes the FP output data from the systolic array into the OwL-P data format as shown in Fig. 3(a).

B. OwL-P Processing Element Implementation

The conventional fused FP MAC unit introduces significant hardware overhead due to the process of alignment, normalization and rounding [28], [29]. However, the small variation in exponents in the OwL-P format presents an opportunity for more efficient computation using an INT unit. The detailed structure of OwL-P PE based on the proposed idea is depicted in Fig. 4(a). This PE executes an 8-way dot-product with built-in outlier paths. The path selection unit, positioned after the multiplication units, dictates the intra-PE path where the

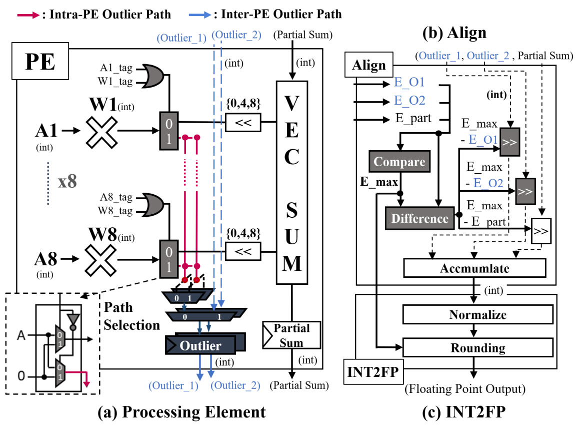


Fig. 4. (a) Detailed implementation of a PE. Components related to outliers are highlighted. (b) Align unit to align the outlier MAC results and normal MAC results. (c) INT2FP unit to convert accumulated results back into the FP format.

multiplication result is transferred. Since there are two outlier registers at the end of outlier path as shown in Fig. 4(a), each PE can deliver up to two outlier results to the lower PE in a single cycle with inter-PE outlier paths. The conflict between inter and intra-PE outliers path are avoided through the scheduling method, which will be introduced in Section V-A.

Meanwhile, the majority of the activation and weight data in LLMs consist of the normal multiplication results. Before the accumulation, each normal multiplication result mul_{r} undergoes a shift based on the combination of the shifting bits transferred from the decoder, for both activation sh_{a} and weight sh_{w}, as follows:

$$\text{Shifted result} = mul_{\mathrm{r}} \ll 4 * (sh_{\mathrm{a}} + sh_{\mathrm{w}})$$

Since the shifting bit is the MSB of the bias bits in the OwL-P data format, a 4-bit shifting of the significand is applied after multiplication. Note that shifting of the significand based on 2 LSBs on the bias bits already occurred in the bias decoder, as explained in Section III-B. A 3-way {0,4,8}-bit variable shifter leads to much smaller hardware overhead than the large barrel shifters used in conventional FP units.

C. Floating Point Conversion of PE Output

At the end of the column in the systolic array, outlier results and partial sum are converted to a single FP number based on its exponent value, as depicted in Fig. 4(b) and (c). Each outlier has its own exponent for conversion. The exponent for the partial sum $E_{\_\mathrm{part}}$ is the sum of the shared exponent for activation and the shared exponent for weight. The exponent value of the outlier result $E_{_}$o is either (shared exp + outlier exp) or (outlier exp + outlier exp) depending on the conditions of input and weight values. The align unit identifies the maximum exponent $E_{\_\mathrm{max}}$ among shared exponent and outlier exponents, and then align the partial sum and outlier results based on the respective difference from $E_{\_\mathrm{max}}$ as shown in Fig. 4(b). The accumulated result from the align unit is converted into a FP result in the INT2FP unit as shown in Fig. 4(c). This way, OwL-P can deal with both normal and outlier values within the same PE achieving the same numerical accuracy as the conventional FP

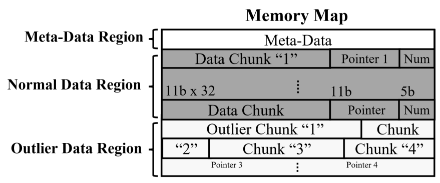


Fig. 5. Off-chip memory map for the OwL-P data format.

computations. Consequently, the inference of the LLM yields the same results as using conventional FP arithmetic unit.

D. Data Memory Map

The key to data compression in OwL-P lies in its utilization of the shared exponent and efficient outlier handling mechanism. Fig. 5 illustrates the off-chip memory map for the OwL-P data format, which consists of three regions. First, the metadata region includes the chunk start address, shared exponent, and layer information. The shared exponent for each subset of layers is stored in the meta-data region once the training is completed. Second, the normal data region represents a set of 32 values, each comprising sign, bias, and fraction for each 11-bit value. The subsequent 11-bit field serves as a pointer to indicate the address where outlier data are stored. Additionally, a 5-bit data field denotes the number of outliers that are present in the 32 data points. Lastly, outlier chunks are stored in outlier data region to provide exponents for outliers that exist in the corresponding normal data chunk. The location of the outlier chunk can be determined by an address counter based on the number of outliers for each normal data region. In case the number of outliers is too large for all the outlier exponents to be stored in the on-chip buffer, the outliers can be fetched from the external memory using a combination of the 11-bit address pointer values and meta-data.

V. Outlier-aware Scheduling

A. Regulating Number of Outliers

In the proposed OwL-P processor, the data chunks packed in the unified buffer are decoded by the bias decoder unit, and then forwarded into the systolic array. There can be an issue when the number of outliers in a given input column exceeds the number of the column-wise inter-PE outlier bypass paths. To address the challenge, we propose an outlier-aware scheduling that regulates the number of outliers. Fig. 6 shows

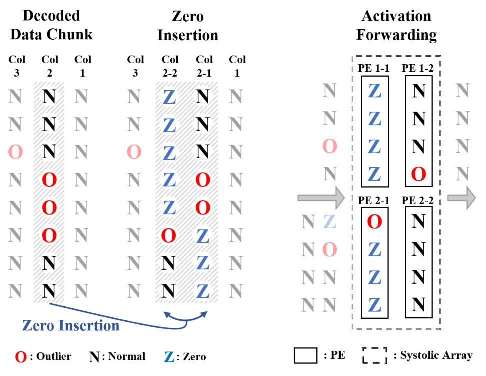


Fig. 6. Regulating number of outliers in a single column with zero insertion for outlier scheduling.

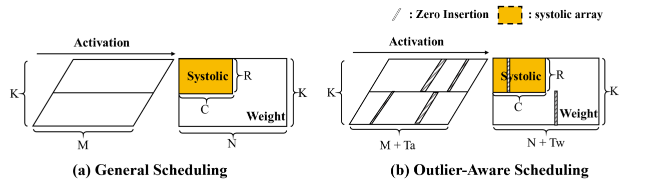


Fig. 7. (a) Data scheduling for general 2D systolic array. (b) OwL-P scheduling with additional cycle Ta and Tw to support outlier handling.

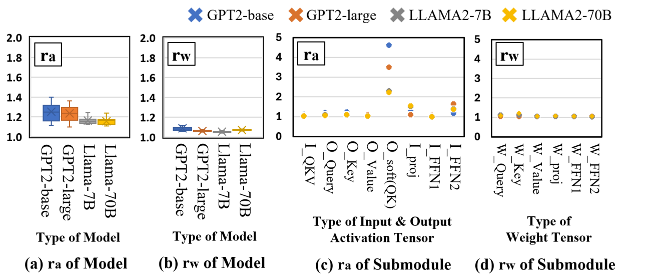


Fig. 8. (a) r_a across various models. (b) r_w across various models. (c) r_a for various activation tensor of submodule. I/O indicates input/output activation respectively. (d) r_w for various weights tensor of submodule.

an example for regulating the number of outliers up to the number of outlier paths. In this simplified setup, each PE can handle up to two outlier results in each cycle. In case of the column 1, the number of outlier is not greater than 2, and hence the outlier paths seamlessly handle both normal values and outliers. However, when the number of outliers is greater than the number of the outlier paths, we insert zeros to limit the number of outliers in a single PE column. For example, the input column 2 has 3 outliers. In this case, outlier scheduler insert zeros to column 2 and reconstruct it into columns 2-1 and 2-2, each containing 2 and 1 outliers respectively. Following the zero insertion, the activation values are forwarded into the PE through the data setup unit, avoiding the outlier data collision during computation. In principle, this approach allows for handling an arbitrary number of outliers when necessary. Although adding extra cycles due to zero insertions degrades the computational speed, the impact on overall performance is negligible in LLM applications due to the low probability of outlier occurrence.

B. *Scheduling Overhead*

We analyze the implication of our outlier-aware scheduling on computing speed. Fig. 7(a) illustrates operation of a systolic array, assuming operand matrices of size (M, K) for the activation and (K, N) for the weight, with K representing the reduction dimension in GEMM for an (R, C) systolic array. In the weight stationary dataflow, total computation cycles can be expressed as following equation [30].

$$T_{\text{cycle}} = (2R + C + M - 2) \times \left\lceil \frac{N}{C} \right\rceil \times \left\lceil \frac{K}{R} \right\rceil \quad (3)$$

To assess the impact of the zero insertion cycles depicted in Fig. 7(b) on computational performance, we define $r_a = (M + T_a)/M$ as the relative increase in the inputting cycles related to M due to the zero insertion cycles caused by the activation outliers T_a. Similarly, we define $r_w = (N + T_w)/N$ as the relative increase in the effective number of weight columns

TABLE III
r_A VALUES FOR VARIOUS DATASETS WITH LLAMA2 FAMILY.

	HellaSwag	**WinoGrande**	**PIQA**	**WikiText-2**	**MMLU**
Llama-7B	1.216	1.297	1.359	1.168	1.179
Llama-70B	1.263	1.282	1.345	1.158	1.126

The r_w value of the Llama2-7B and 70B model remains constant at 1.052 and 1.071 respectively, regardless of the dataset.

TABLE IV
r_A AND r_W VALUES FOR VARIOUS DATASETS WITH BERT FAMILY.

	r_a		r_w	
Dataset	**SQuAD2**	**GLUE**	**SQuAD2**	**GLUE**
BERT-Base	1.293	1.306	1.048	1.052
BERT-Large	1.301	1.308	1.049	1.052

[*] GLUE datasets includes cola, mnli, mlpc, qnli, qqp, rte, sst2, stsb

N due to the weight outliers T_w. Then, the total computation cycles can be modified considering the effect of r_a and r_w as follows.

$$T_{\text{cycle}} = (2R + C + M \times r_a - 2) \times \left\lceil \frac{N \times r_w}{C} \right\rceil \times \left\lceil \frac{K}{R} \right\rceil \quad (4)$$

The impact on overall compute cycle is expected to be limited since M tends to be relatively small as limited batching typically occurs in LLM workloads. Experimental results r_a and r_w in Fig. 8 and Table III, IV are evaluated with two outlier paths for activation or weight respectively. In Fig. 8(a) show that r_a values range between 1.1 and 1.3 across various networks. In Fig. 8(c), the activation processed through non-linear functions such as softmax, tends to exhibit relatively high r_a, but their overall impact is negligible, as confirmed in Fig. 8(a). In Fig. 8(b) and (d), we can observe that r_w does not exceed 1.1 in all cases, demonstrating that the influence of r_w is limited across various neural networks.

Additionally, Table III demonstrates the value or r_a during inference using the Llama2-7B and 70B model across various datasets. It can be observed that there exists negligible variation in the values of depending on the dataset including WikiText-2, HellaSwag, WinoGrande, PIQA and MMLU [31]–[35]. Table IV shows the r_a and r_w extracted from the profiling of BERT family for various datasets including GLUE and SQuAD2 [36], [37]. Each model exhibits similar r_a and r_w values, regardless of the dataset.

VI. EVALUATION

A. *Evaluation Methodology*

We analyze the performance gain and energy savings of our engine across 10 different workloads. Evaluations are conducted on various transformer-based models BERT, GPT2, Llama2 families. We use Synopsys Integrated Circuit Compiler II for place and route the systolic array with commercial 28nm CMOS process. We compile SRAM memory array for buffers. The ScaleSIM was used for cycle accurate simulations [30]. We conduct simulations assuming a off-chip bandwidth of 256 GB/s provided by HBM2.

B. *Hardware Evaluation*

The baseline FP hardware performs BF16 multiplication and FP32 accumulation at 500MHz. The proposed design conducts 8-way INT dot-product at the same frequency target. Fig. 9 shows the area and power comparison between the baseline and the proposed design for various numbers of outlier paths

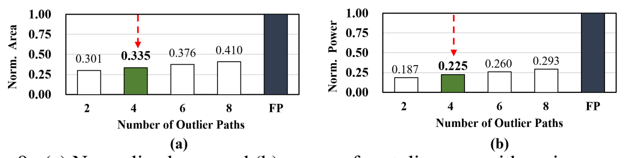


Fig. 9. (a) Normalized area and (b) power of systolic array with various number of outlier paths.

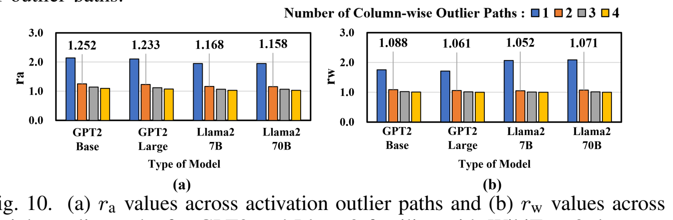


Fig. 10. (a) r_a values across activation outlier paths and (b) r_w values across weight outlier paths for GPT2 and Llama2 families with WikiText-2 datset.

per array. It can be observed that the proposed design consumes much smaller area and power than the baseline thanks to the use of INT unit. Considering the trade-off between the hardware overhead and the performance impact of the outlier paths shown in Fig. 9 and Fig. 10, we choose to use total of 4 outlier paths within a PE. Table V summarizes the details of design parameters and comparison of power and area between the baseline and the proposed OwL-P design. The baseline consists of 16 systolic arrays, each containing 32×32 MACs. The proposed hardware OwL-P employs INT-based units, thereby containing 3× larger number of computation units in the same compute area.

TABLE V
THE COMPARISON TABLE BETWEEN TPU-LIKE DESIGN AND THE OWL-P.

Parameter	TPU-like Systolic Engine	OwL-P
Data Type	BF16 Mult, FP32 Add	INT-based MAC (4-outliers/PE)
PE Pipeline	4-stage	2-stage
Memory	12MB	12MB
Power (W)	**13.04W**	**8.93W**
MACs	**16384**	**49152**
Area (mm^2)	**Total: 49.46mm^2** MAC array 73.1% Datasetup 2.7% Layout Overhead 23.3%	**Total: 49.52mm^2** MAC array 73.3% Datasetup 2.0% Others 4.7% Layout Overhead 20.0%

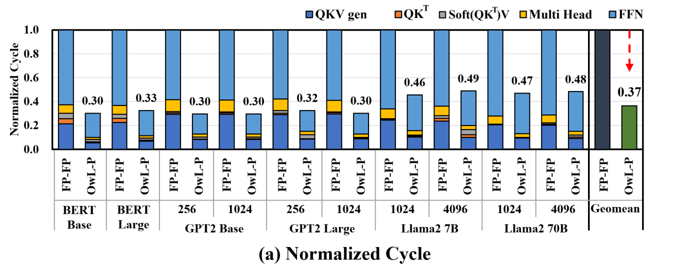
The area and power numbers are for the compute logic excluding memory buffers.

C. Performance Gain

Fig. 11(a) shows the total cycles normalized to that of the baseline for each network model. A detailed breakdown of operations such as QKV generation, attention score calculation, multi-head projection, and FFN are also provided. In the case of BERT family, matrix multiplication is performed with an input token length of 512. When using GPT2 and Llama2 families for generation tasks, we apply KV caching and continuous batching, utilizing a batch size of 32 [38]. As expected, the proposed INT-based PE design shows a much smaller total cycle time than the FP baseline across all the models because 3× more PEs are placed within the same compute area. In addition to the larger number of PEs, the off-chip access is also reduced because the proposed number format uses a smaller memory footprint. Overall, the proposed design achieves 2.70× performance gain on average compared with the FP baseline.

D. Energy Savings

Fig. 11(b) shows the energy savings of the proposed design over the FP baseline. The proposed design achieves significant


(a) Normalized Cycle

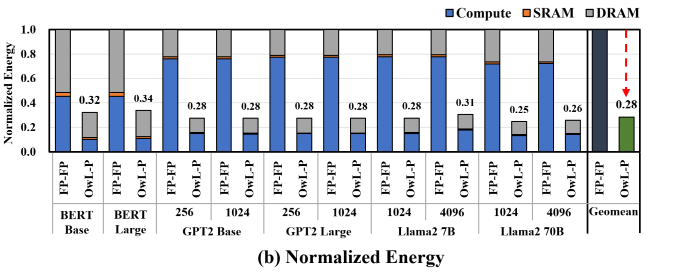

(b) Normalized Energy

Fig. 11. (a) Relative number of total cycles and (b) relative energy consumption. Each value is normalized to the counterpart value of FP baseline. The target token lengths for generation are 256 and 1024 for GPT2, and 1024 and 4096 for Llama2.

energy saving over the FP baseline across various network models. Due to the improved energy efficiency from the INT MAC over FP MAC, the energy consumed in a single PE tile decreases 4.89×. The reduced off-chip access due to the compressed data format also contributes to the substantial reduction in energy consumption for off-chip communications. Additionally, the use of compressed data format leads to an increase in effective bandwidth, which result in improved utilization of the array. These factors contribute to 4.19-5.54× compute energy savings. Overall, the proposed design achieves 2.94-4.04× energy reduction compared to the FP baseline.

VII. CONCLUSION

We propose an outlier-aware INT-based accelerator OwL-P, which produces equivalent numerical accuracy as FP MAC hardware. Our design philosophy is based on the observation that a narrow range of exponents covers the majority of weight and activation values in transformer-based LLM networks, facilitating compression through a shared exponent and bias bits in OwL-P data format. This compression, coupled with the processing of outliers alongside normal values within the same INT-based PE array with built-in outlier paths, distinguishes OwL-P from previous approaches that relied on independent FP MACs for outlier handling. By utilizing an efficient outlier bypass mechanism, OwL-P achieves significant area and energy savings. Simulation results demonstrate a remarkable 2.70× performance gain and 3.57× energy savings compared to conventional FP MAC-based accelerators.

ACKNOWLEDGEMENT

This work was supported in part by Institute of Information communications Technology Planning Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2024-00395134, DPU-Centric Datacenter Architecture for Next-Generation AI Devices, IITP-2023-RS-2023-00256081: artificial intelligence semiconductor support program to nurture the best talents), BK21 FOUR program at Seoul National University, and ISRC at Seoul National University. The EDA tool was supported by the IC Design Education Center(IDEC). (Corresponding Author: Jae-Joon Kim).

References

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.

[2] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, “Language models are unsupervised multitask learners,” *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

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

[4] B. Darvish Rouhani, D. Lo, R. Zhao, M. Liu, J. Fowers, K. Ovtcharov, A. Vinogradsky, S. Massengill, L. Yang, R. Bittner *et al.*, “Pushing the limits of narrow precision inferencing at cloud scale with microsoft floating point,” *Advances in neural information processing systems*, vol. 33, pp. 10 271–10 281, 2020.

[5] B. Darvish Rouhani, R. Zhao, V. Elango, R. Shafipour, M. Hall, M. Mesmakhosroshahi, A. More, L. Melnick, M. Golub, G. Varatkar *et al.*, “With shared microexponents, a little shifting goes a long way,” in *Proceedings of the 50th Annual International Symposium on Computer Architecture*, 2023, pp. 1–13.

[6] T. J. Ham, Y. Lee, S. H. Seo, S. Kim, H. Choi, S. J. Jung, and J. W. Lee, “Elsa: Hardware-software co-design for efficient, lightweight self-attention mechanism in neural networks,” in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2021, pp. 692–705.

[7] J. Lin, J. Tang, H. Tang, S. Yang, X. Dang, and S. Han, “Awq: Activation-aware weight quantization for llm compression and acceleration,” *arXiv preprint arXiv:2306.00978*, 2023.

[8] J. Jang, Y. Kim, J. Lee, and J.-J. Kim, “Figna: Integer unit-based accelerator design for fp-int gemm preserving numerical accuracy,” pp. 760–773, 2024.

[9] 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.

[10] N. P. Jouppi, D. H. Yoon, M. Ashcraft, M. Gottscho, T. B. Jablin, G. Kurian, J. Laudon, S. Li, P. Ma, X. Ma *et al.*, “Ten lessons from three generations shaped google’s tpuv4i: Industrial product,” in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2021, pp. 1–14.

[11] A. Rico, S. Pareek, J. Cabezas, D. Clarke, B. Ozgul, F. Barat, Y. Fu, S. Münz, D. Stuart, P. Schlangen *et al.*, “Amd xdna™ npu in ryzen™ ai processors,” *IEEE Micro*, 2024.

[12] Intel Corporation, “Intel® Core™ Ultra Processor Datasheet, Volume 1 of 2,” Tech. Rep., 2024, accessed: 2024-09-17. [Online]. Available: https://www.intel.com/content/www/us/en/content-details/792044/intel-core-ultra-processor-datasheet-volume-1-of-2.html

[13] X. Wei, Y. Zhang, X. Zhang, R. Gong, S. Zhang, Q. Zhang, F. Yu, and X. Liu, “Outlier suppression: Pushing the limit of low-bit transformer language models,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 17 402–17 414, 2022.

[14] X. Wei, Y. Zhang, Y. Li, X. Zhang, R. Gong, J. Guo, and X. Liu, “Outlier suppression+: Accurate quantization of large language models by equivalent and optimal shifting and scaling,” *arXiv preprint arXiv:2304.09145*, 2023.

[15] 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.

[16] T. Dettmers and L. Zettlemoyer, “The case for 4-bit precision: k-bit inference scaling laws,” in *International Conference on Machine Learning*. PMLR, 2023, pp. 7750–7774.

[17] 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.

[18] J. Koo, D. Park, S. Jung, and J. Kung, “Opal: Outlier-preserved microscaling quantization a ccelerator for generative large language models,” *arXiv preprint arXiv:2409.05902*, 2024.

[19] T. Dettmers, M. Lewis, Y. Belkada, and Z. Luke, “Llm. int8 (): 8-bit matrix multiplication for transformers at scale,” *arXiv preprint arXiv:2208.07339*, 2022.

[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 *International Conference on Machine Learning*. PMLR, 2023, pp. 38 087–38 099.

[21] M. Drumond, T. Lin, M. Jaggi, and B. Falsafi, “Training dnns with hybrid block floating point,” *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[22] 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.

[23] S. Jain, S. Venkataramani, V. Srinivasan, J. Choi, K. Gopalakrishnan, and L. Chang, “Biscaled-dnn: Quantizing long-tailed datastructures with two scale factors for deep neural networks,” in *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019, pp. 1–6.

[24] 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*, 2023, pp. 1–15.

[25] 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,” pp. 1414–1433, 2022.

[26] Y. Li, R. Gong, X. Tan, Y. Yang, P. Hu, Q. Zhang, F. Yu, W. Wang, and S. Gu, “Brecq: Pushing the limit of post-training quantization by block reconstruction,” *arXiv preprint arXiv:2102.05426*, 2021.

[27] 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.

[28] J. Y. F. Tong, D. Nagle, and R. A. Rutenbar, “Reducing power by optimizing the necessary precision/range of floating-point arithmetic,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, no. 3, pp. 273–286, 2000.

[29] Y. Tao, G. Deyuan, F. Xiaoya, and J. Nurmi, “Correctly rounded architectures for floating-point multi-operand addition and dot-product computation,” in *2013 IEEE 24th International Conference on Application-Specific Systems, Architectures and Processors*. IEEE, 2013, pp. 346–355.

[30] A. Samajdar, Y. Zhu, P. Whatmough, M. Mattina, and T. Krishna, “Scale-sim: Systolic cnn accelerator simulator,” *arXiv preprint arXiv:1811.02883*, 2018.

[31] S. Merity, C. Xiong, J. Bradbury, and R. Socher, “Pointer sentinel mixture models,” 2016.

[32] K. Sakaguchi, R. L. Bras, C. Bhagavatula, and Y. Choi, “Winogrande: An adversarial winograd schema challenge at scale,” *Communications of the ACM*, vol. 64, no. 9, pp. 99–106, 2021.

[33] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi, “Hellaswag: Can a machine really finish your sentence?” *arXiv preprint arXiv:1905.07830*, 2019.

[34] Y. Bisk, R. Zellers, J. Gao, Y. Choi *et al.*, “Piqa: Reasoning about physical commonsense in natural language,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 05, 2020, pp. 7432–7439.

[35] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, “Measuring massive multitask language understanding,” *arXiv preprint arXiv:2009.03300*, 2020.

[36] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, “Glue: A multi-task benchmark and analysis platform for natural language understanding,” 2018.

[37] P. Rajpurkar, R. Jia, and P. Liang, “Know what you don’t know: Unanswerable questions for squad,” *arXiv preprint arXiv:1806.03822*, 2018.

[38] G.-I. Yu, J. S. Jeong, G.-W. Kim, S. Kim, and B.-G. Chun, “Orca: A distributed serving system for {Transformer-Based} generative models,” in *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, 2022, pp. 521–538.