Modeling a VSQ Accelerator

Samuel Bruce
sgbruce@mit.edu

Nicholas Dow
nldow@mit.edu

I. Introduction

The wide-scale adoption of machine learning algorithms in the form of Dense Neural Networks (DNNs) across various domains has led to demand for computational resources skyrocketing. Recently, transformer based architectures have become increasingly common for everyday tasks from natural language processing to CV. Transformer architectures have deterministic structure and require large amounts of computation, making them a target for specialized hardware which can improve performance for training and inference. One approach used to improve the efficiency of DNNs, in both space and computational complexity, is to reduce the representation size of the data in the model, called quantization. Quantization, however, can introduce rounding errors that impact the performance of models, and have been shown to impact transformer models more severely than CNNs [2].

In this paper we analyze a quantization based DNN accelerator and assess it's performance across several parameters. The accelerator, presented in [1], impressively achieves 38.7 TOPS/W at 0.67V with only a 0.7% loss of accuracy on the BERT-Base dataset. The accelerator achieves such low loss of accuracy and high throughput by using per-vector scaled quantization (VSQ), which employs an independent scale factor for each 64 element vector. These scale factors allow quantization down to 4-bit data representation with high accuracy. We begin by modeling this accelerator, replicating energy and area measurements documented in the paper sing a simulated CiMLoop environment. Then, using our software model, we investigate using the accelerator as a PE in a larger system and find an optimal system layout. Finally, we examine the effects of changing the technology used in the accelerator from the given 5nm technology to 40nm and 65nm components.

We made several insights in our analysis of the accelerator. Firstly, the workload shape greatly affects the buffer utilization in CiMLoop, especially for the A-Buffer. Smaller values for matrix A's row-size compared to the row-size of matrix B can limit the usage of the A-Buffer space given the availability of the accumulator. This becomes a focus of the system architecture, utilitzing the accelerator as a PE. We found that, for this system, and 8x2 array of PE's along with a global PE with a 65,565-bit SRAM memory optimizes the workload for large, square matrices. Finally, we found that under these optimal conditions the amortized cost of the system architecture is relatively small, with the majority of the energy consumption and area usage of the system being taken by the accelerator PEs.

II. Related Works

A. Foundational Paper

The primary work used for the basis of our model was accelerator architecture described in Keller et. al. [1]. The paper describes a unique hardware approach to achieving high energy efficiency and lower chip area footprint via smart quantization strategies. Quantization has the ability to lower the data-width of memory and computation, allowing chip designs to do the same computations with less area at the cost of some accuracy. Keller et. al. argues that while some accuracy can be traded for efficiency, there are quantifiable thresholds of accuracy loss where simply downsizing the model incurs less of an accuracy loss than quantizing the larger model. They argue that their main contribution of building an PE architecture around the concept of vector scaling quantization(VSQ) allows for models to be quantized to 4-bit while doing a simpler approach of matrix scaled quantization would cross the catastrophic accuracy threshold. In employing

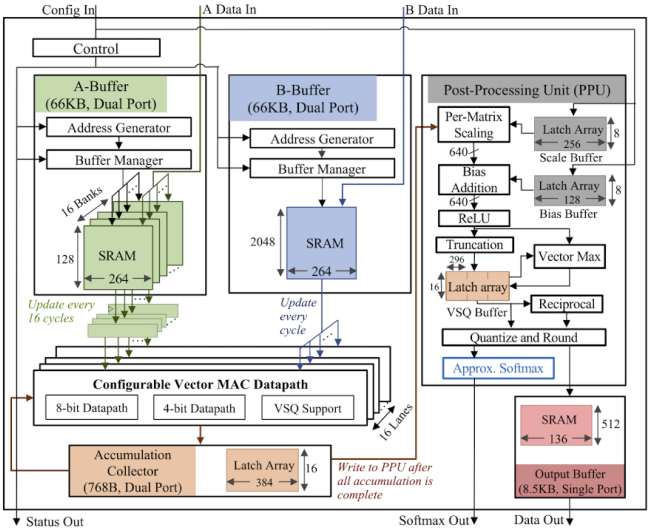

Fig. 1. Datapath Diagram of Accelerator [1]

their quantization strategy, their evaluations achieve efficiency of 95.6-TOP/W at 0.67 V using their 4-bit VSQ data path. Their VSQ strategy involved having a scaling factor per vector of 64 4-bit values, allowing for the quantization of the original input and weight values that introduces less noise as the range of values that the quantization had to compress was much

smaller than that scaling the entire matrix, as demonstrated by Figure 2. By having the factor be per-vector, additional hardware is needed to compute and apply the scaling during MACs as seen in Figure 1, but this overhead is amortized over the entire vector and needed only an extra 8-bits of data per vector. The accelerator achieves it's efficiency through use of the lower energy cost of 4-bit operations enabled by this quantization strategy.

The PE datapath design also leverages data reuse strategies to achieve it's efficiency. The accelerator is innately parallel over the vector width and the 16 vector lanes. The lowest two loops of the loop nest are input stationary over the tile of A, and output stationary over the inner tiling loop. Each bank of A is equipped with a register that is written to every 16 MACs for high temporal reuse, and B is broadcasted across the vector lanes for high spatial reuse.

B. Other Papers

The authors of MINOTAUR also attempt tog improve efficiency while maintaining accuracy of more precise data types [3] through datatype design. They introduce the Posit, which is a variant of FP8 that can express a larger range of values more accurately than FP8 at a slightly increased cost to FP8, with the authors aiming to be cheaper to compute with than Brain FP8 but more accurate that FP8. With their setup, they achieve 0.5 TOPS/W using a 40nm node, which we can directly compare to during our evaulations.

Another work attempting to exploit quantization for efficiency gains is *Panacea* by Kam et. al. [4]. *Panacea* introduces a novel deep neural network (DNN) accelerator that combines accuracy-preserving asymmetric quantization with bit-slice sparsity to improve energy efficiency without sacrificing model accuracy. The asymmetric quantization technique employed in Panacea allows for finer control over the quantization range by separately quantizing positive and negative values, which is particularly effective for maintaining accuracy in low-precision settings. Furthermore, its use of bit-slice sparsity enables dynamic adjustment of computational effort based on bit significance, leading to notable energy savings.

A work that targets the outliers that widen the quantization scaling is that of OPAL by Koo et. al. [5]. *Opal* introduces a hardware-software co-design approach to enhance the efficiency of large language models (LLMs) by employing a novel quantization technique. This method utilizes a microscaling data format that preserves a select number of activation outliers within each sub-tensor block, ensuring that significant information is retained even under aggressive quantization. By combining this with mixed-precision strategies—allocating higher bit-widths to sensitive layers and lower bit-widths to others—OPAL achieves substantial reductions in memory usage and computational demands. Additionally, the architecture incorporates a $\log_2$-based approximation for softmax operations, further optimizing power efficiency.

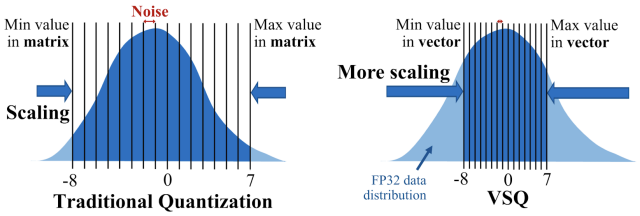

Fig. 2. Compression advantage of VSQ over Matrix Scaling Quantization [1]

III. Modeling the VSQ Accelerator

As our project is modeling an accelerator and following on with state-space exploration of architectural changes, we leave most of the implementation and experimental approach to section IV.

IV. Experiment Setup

To test the various architectural configurations, we started by implementing a software instantiation of the target accelerator PE in CiMLoop. We found this to be the most feasible way for us to quickly make the desired changes to the system architecture while making the accurate measurements of the energy and area taken up by each component.

A. Architecture Modeling

To attempt to model the parameters of original accelerator as faithfully as possible, we minimized the simplification of the components of accelerator as much as we could. Given the limitations of CiMLoop however, we had to make some compromises when modeling the most complex unit of the accelerator, the PPU. As the post-processing unit operates on the accumulated sums of the output, it's impact on the energy per compute of the accelerator is less than that of the buffers and MACs, so simplifying it had less of an impact on our model's accuracy. For the components of the Bias and Scaling buffers, we input energy-scaled register files to represent them only reading from DRAM per computed tile of the input map. The ReLU was modeled using a comparator, and the truncation and quantization elements were modeled using multipliers scaled to the width of the data. The entire PPU compound component was energy-scaled to represent it's frequency of execution on batches of output data.

Another component that was difficult to directly model behavior for in CiMLoop was the accumulator. The fan-out only for compute disallowed for the read/write expected after every vector MAC operation, so we had to result to directly energy scaling to the reported energy baseline stated in the original paper.

The rest of the accelerator hardware was much easier to model in CiMLoop. We modeled the architectural fan-out with 16 vector-lanes with a bank of the A Buffer under each. The A Buffer and B Buffer were modeled as smart SRAM components (a memory component along with an address generator), with the A Buffer having a register file directly underneath each of it's 16 banks. Given that the original accelerator had data widths of both 8-bit and 4-bit, we modeled

that expansion of data fanout at the MAC level similar to the paper, where the 4-bit vector had 64 multiples and the 8-bit had 32. Where it was appropriate to pass the per-vector scale factor, we expanded the vector width to 264 to accommodate the extra 8-bit scale factor. The scale factor computation and multiplication was done with fixed-width multipliers that estimate the $8 \times 8 \rightarrow 8$ and $8 \times 14 \rightarrow 22$ bit multiplications, respectively.

B. Mapping Matrix Multiply

The accelerator paper provided an excellent figure showing the intended mapping of an instance of matrix multiply to the PE's various buffers and vector lanes, seen in 3.

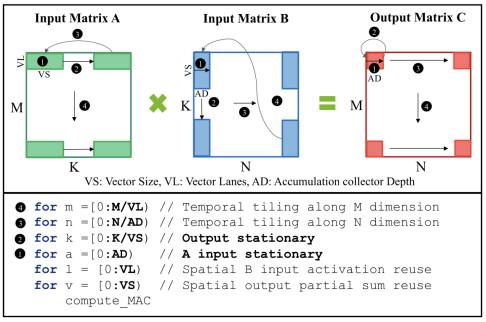

Fig. 3. Mapping MM to the Accelerator

Mapping matrix multiplication as prescribed in the photo maximizes reuse by keeping A and the Outputs stationary as much as possible. The factors responsible for the tile size, *VL*, *AD*, and *VS* are specified by the architecture as 16, 16, and $256/\#\text{bits}$, respectively. To map the problem in CiMLoop, we treated the matrix *A* as the Weights dataspace, *B* as the inputs dataspace, and *C* the Outputs dataspace. The dimensions of *A* are (M, C, X), *B* is (C, P, Y), and *C* is (M, P, Z), where X, Y, Z is the representation size of each element in bits. To allow the mapper to execute efficiently and according to the tiling above, we specified for each component except the virtualized MAC operation that there would be no iteration over the representation axes X, Y, Z. We further constrained the spatial dimensions such that the vector MACs mapped spatially along the C dimension and the A-Buffer banks mapped along the M dimension. For temporal constraints, we ensured that the buffers iterated over dimensions in the order $[C, P, M]$, corresponding to steps $(2, 3, 4)$ in 3. Step 1, the tiling, is handled by the spatial constraints. Further, for the A-Buffer we required that dimensions be maximized in the order $[C, P, M]$, to make sure that entire rows of the Weights dataspace are computed before incrementing P, and then all P's are exhausted before incrementing M, as is shown in 3. For components not used as storage elements for the dataspaces (PPU, scale factor computation, etc.) we ensured that no looping was occurring by specifying no factors or no temporal reuse.

C. System Configurations

To build the accelerator into a complete system, we opted for a simple design that can generally describe most of the important design choices. We model the accelerator as a single PE, then arrange a fixed number of PEs in a 2-dimensional spatial grid. We connect all of these PEs to a "Global PE", which in the simplest case is a common scratchpad consisting of a storage element and an address generator. Adding layerwise operations in the Global PE would be relatively straightforward without introducing much computational overhead. The Global PE reads and writes the Weights, Inputs, and Outputs dataspaces from a DRAM, modeled as 64-bits wide and Infinite depth. The system layout can be seen in 4.

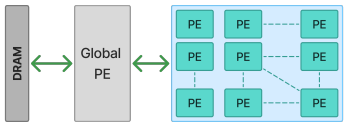

Fig. 4. System Layout

In our system we were most interested with the optimal Global PE scratchpad space and the optimal spatial arrangement of PEs. For our analysis we decided to use 1-16 PEs in the spatial grid, as this allows several interesting configurations while keeping the mapspace small enough to efficiently search for good architectures. For the scratchpad size, we tried 2^k, $k \in \{12, 14, 16, 18, 20, 22, 24\}$. In practice, these values encompassed scratchpads of reasonable sizes (from a few kilobits to $>$ 10 megabits) to see reuse across the PEs in different configurations.

D. Impact of Transistor Technology

The accelerator was designed for 5nm technology, and optimized accordingly. Additionally, we optimized our system configuration for 5nm components in CiMLoop. To see the impact of emerging hardware technologies such as 5nm compared to older, larger components, we decided to switch out the 5nm technology for 40nm and 65nm components and compare energy efficiency and area usage of the system.

V. Results and Discussion

In our experiments we were able to recreate key area and energy metrics from [1]. Using these results to ensure we were modeling the architecture accurately, we were then able to extend the accelerator into a system and scan for optimal architecture choices. We found that a mesh of $(8, 2)$ PEs with a Global scratchpad size of 2^{16} was optimal, bringing the amortized energy down to ≈ 105 fJ/MAC. In this section we will first present our results showing consistency with [1], then extend to our exploration of system sizes and finally investigate other technologies.

A. Recreating Results

Using the methodology from section IV, we were able to accurately recreate area and energy metrics from the original paper. Notably, the paper lists the relative energy of different parts of the accelerator, normalized to the energy of the 4-bit datapath, but fails to document absolute numbers for these energies. The area, however, can be deduced from the mock chip layout to be roughly $\approx 150,000 \mu m^2$. By specifying the mapping as above and tuning the per-component energy usage, we were able to create the figures 5 and 6.

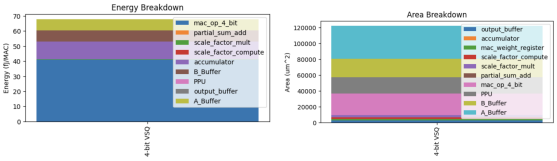

Fig. 5. 4-bit VSQ Energy per MAC and Area

```
20 output_buffer [ Outputs:1024 (1024) ]
21 PPU [ Outputs:1024 (1024) ]
22 B_Buffer [ Inputs:524288 (524288) ]
23 accumulator [ Outputs:1024 (1024) ]
24 ----------------------------------------
25 |             for P in [0:2)
26
27 inter_vec_mac_lane_spatial [ ]
28 --------------------------------
29 |               for M in [0:16) (Spatial-X)
30
31 A_Buffer [ Weights:32768 (32768) ]
32 ----------------------------------------
33 |                 for P in [0:8)
34 |                   for C in [0:128)
35
36 mac_weight_register [ Weights:256 (256) ]
37 scale_factor_compute [ ]
38 scale_factor_mult [ ]
39 partial_sum_add [ Outputs:4 (4) ]
40 inter_1bit_x_1bit_mac_spatial [ ]
41 --------------------------------
42 |                     for Z in [0:4) (Spatial-X)
43 |                       for Y in [0:4) (Spatial-X)
44 |                         for X in [0:4) (Spatial-X)
45
46 here_to_fix_a_bug [ ]
47 inter_mac_container_spatial [ ]
48 --------------------------------
49 |                           for C in [0:64) (Spatial-X)
50 |                             << Compute >>
```

Fig. 6. Example Mapping for 4-bit VSQ

There are several interesting things to note about the figures. First, as asserted by [1], the datapath energy dominates the amortized energy. The buffers take up some non-zero energy as well, and then all other operations impact the total energy usage very little on average. The total energy used is $\approx$ 69fJ/MAC. For the area, the sizes roughly match up with the footprint from the original paper. While we had to tune the MAC unit area manually, as that was a user-defined component, all other components from the library accurately reflected the area without tuning. Note that the total area consumed is $\approx 120,000 \mu m^2$, which is less than the die size $153,005 \mu m^2$. Looking at the annotated accelerator floorplan you can see that this difference is accounted for in unused area on the die. The area metrics in our simulation only show component area and have no concern for die layout concerns which may introduce inefficiencies, thus slightly increasing the area needed for the accelerator in practice.

Finally, examine the sample mapping in 6. The mapping is valid and aligns with the tiling scheme in 3. The A-Buffer goes through all columns C before repeating in the stationary dimension P, while the banks of the A-Buffer are mapped spatially along the row dimension M. The B-Buffer also iterates over the remaining non-saturated dimensions M and P, to maximize reuse. The problem used in this example is square, meaning $|inputs| = |weights| = |outputs|$, and notably this configuration with $C = 64 \cdot 128 = 2^{13}$ leads to full use of both the A-Buffer and the Accumulator (The accumulator says it is only storing 1024 bits, however the partial sums in the accumulator are larger bit representations before post-processing, so this is in fact full utilization). If the C dimension grows, or the P dimension is not large enough, then the accumulator utilization will decrease. Similarly, if $C < 2^{13}$ then the A-Buffer may be under-utilized as the accumulator becomes the mapping constraint (depth 16 means $P \leq 16$ in the A-Buffer with the CiMLoop environment). Overall, for large problems, the accelerator can perform very well and be fully utilized.

We also recreated a graph in the paper which demonstrated the efficiency of the accelerator in TOPS/W as voltage varied, an interesting metric demonstrating the energy efficiency over many operations. We recreated the graph successfully in our model, found in 7. Note that as V decreases to 0.5, the TOPS/W approaches 85. As voltage increases to 1, TOPS/W is roughly 20. For all other metrics in this paper, we use the standard voltage of $0.67V$.

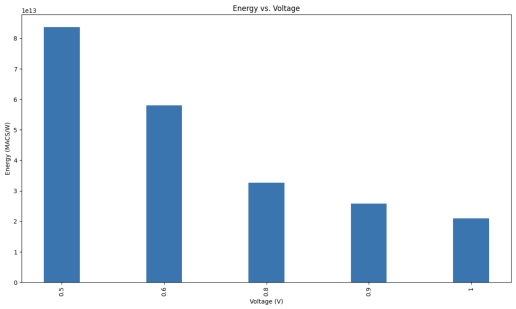

Fig. 7. TOPS/W vs. V

B. Architectural Exploration

We began our exploration by looking at PE counts of 16 or less to iterate our mesh dimensions over, as it provided reasonable square and rectangular meshes to test; the dimensions of the mesh must be powers of two per CiMLoop's restrictions. We mapped the X dimension to P fanout, and Y to M. Our energy results across these configurations can be seen in figure 10; all configurations had a global PE size of 65,536 bits. The results show that the optimal mesh size for the accelerator was an 8 x 2; as the second best was 4 x 2, we theorize that these meshes were more efficient due to temporal reuse in P, as the global PE followed the same map constraints as defined in 3, with P tending to be the innermost loop as C was all present in the A-Buffer.

We then decided to explore the impact of scratchpad size on energy per compute and the area of the accelerator. Taking the best mesh arrangement of 8x2 and holding it constant, we iterated over scratchpad sizes of different magnitudes seen in

figures 8 and 9. As shown in figure 9, beyond the size where the global PE fits enough data for reuse by the buffers to occur rather than go to DRAM, increasing the size of the scratchpad only increases the energy cost of accesses, which conforms to our ideas of larger memories incurring higher costs per access. From these results, we see that a scratchpad larger than 65,536 bits only hurts energy efficiency, and shows how the global scratchpad's size should be tailored for the data-flow mapping and buffer sizes to maximize efficiency.

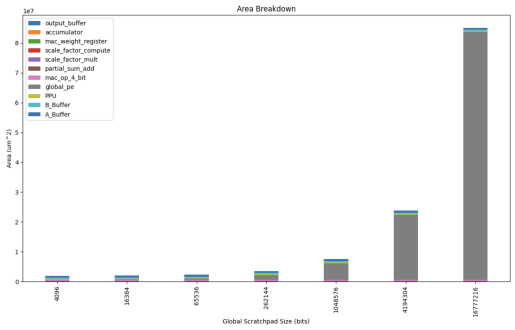

Fig. 8. Area vs. Scratchpad Size

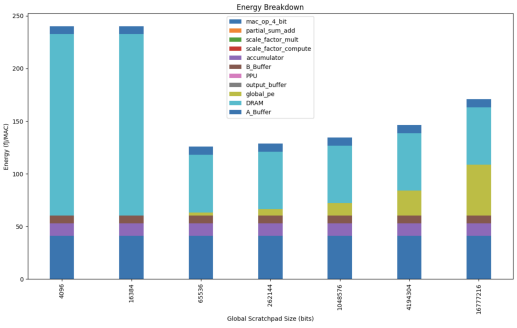

Fig. 9. Energy vs. Scratchpad Size

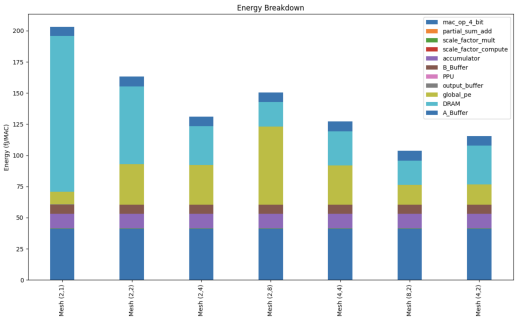

Fig. 10. Energy vs. Mesh Configurations

C. Technology Scaling

The final piece of our exploration, once we had an optimal architecture, was to scale the technology and see the impact of using 5nm components on the energy and area of the accelerator. We used technology scaling in CiMLoop to model the system with 40nm and 65nm components, shown in 11 and 12, respectively. As expected, scaling up the technology greatly increased the area of the accelerator, and it also increased the energy significantly. This makes sense, as larger components are goign to consume more latent energy during operation and while idle, raising the energy consumed per MAC.

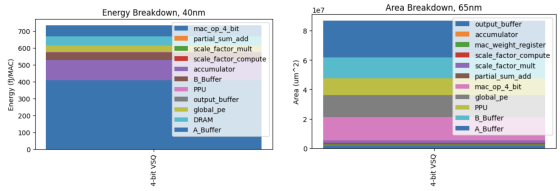

Fig. 11. 40nm Technology Energy and Area

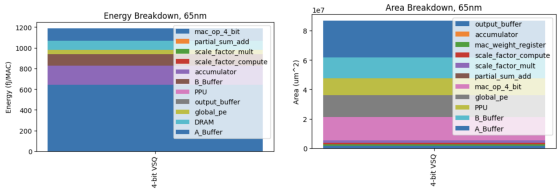

Fig. 12. 65nm Technology Energy and Area

REFERENCES

[1] Ben Keller, Rangharajan Venkatesan, Steve Dai, Stephen G. Tell, Brian Zimmer, Charbel Sakr, William J. Dally, C. Thomas Gray, and Brucek Khailany, "A 95.6-TOPS/W Deep Learning Inference Accelerator With Per-Vector Scaled 4-bit Quantization in 5nm," *IEEE Journal of Solid-State Circuits*, vol. 58, no. 4, April 2023.

[2] C. Sakr, Y. Kim, and N. Shanbhag, "Analytical guarantees on numerical precision of deep neural networks," in *Proc. 34th Int. Conf. Mach. Learn.*, Jul. 2017, pp. 3007–3016.

[3] K. Prabhu, R. M. Radway, J. Yu, K. Bartolone, M. Giordano, F. Peddinghaus, Y. Urman, W.-S. Khwa, Y.-D. Chih, M.-F. Chang, S. Mitra, and P. Raina, "MINOTAUR: A posit-based 0.42–0.50-TOPS/W edge transformer inference and training accelerator," *IEEE Journal of Solid-State Circuits*, vol. 60, no. 4, pp. 1311–1323, 2025, doi: 10.1109/JSSC.2025.3545731.

[4] D. Kam, M. Yun, S. Yoo, S. Hong, Z. Zhang, and Y. Lee, "Panacea: Novel DNN Accelerator using Accuracy-Preserving Asymmetric Quantization and Energy-Saving Bit-Slice Sparsity," in *Proc. 2025 IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, pp. 701–715, 2025, doi: 10.1109/HPCA61900.2025.00059.

[5] J. Park, Y. Kim, S. Bae, M. Kim, C. Park, B. Kang, J. Hwang, Y. Jeon, S. Lee, and H. Shin, "OPAL: Outlier-Preserved Microscaling Quantization Accelerator for Generative Large Language Models," *arXiv preprint arXiv:2409.05902*, 2024. [Online]. Available: https://arxiv.org/abs/2409.05902