Low-Latency PIM Accelerator for Edge LLM Inference

Xinyu Wang, Xiaotian Sun, Wanqian Li, Feng Min, Xiaoyu Zhang, Xinjiang Zhang, Yinhe Han, *Senior Member, IEEE*, and Xiaoming Chen, *Member, IEEE*

***Abstract*—Deploying large language models (LLMs) on edge devices has the potentials for low-latency inference and privacy protection. However, meeting the substantial bandwidth demands of latency-oriented edge devices is challenging due to the strict power constraints of edge devices. Resistive random-access memory (RRAM)-based processing-in-memory (PIM) is an ideal solution for this challenge, thanks to its low read power and high internal bandwidth. Moreover, applying quantization methods, which require different precisions for weights and activations, is a common practice in edge inference. But existing accelerators cannot fully leverage the benefits of quantization, as they lack multiply-accumulate (MAC) units optimized for mixed-precision operands. To achieve low-latency edge inference, we design an RRAM-based PIM die that integrates dedicated energy-efficient MAC units, providing both computation and storage capabilities. Coupled with a dynamic random-access memory (DRAM) die for storing the key-value (KV) cache, we propose Lyla, an accelerator for low-latency edge LLM inference. Experimental results show that Lyla achieves 3.8×, 2.4×, and 1.2× latency improvements over a GPU and two DRAM-based PIM accelerators, respectively.**

***Index Terms*—Large language model inference, processing-in-memory, edge accelerator.**

I. Introduction

RUNNING large language models (LLMs) on edge devices, rather than employing the widely used cloud services, offers better privacy protection, as edge inference avoids uploading user data. Moreover, edge inference has the potential to achieve lower latency than cloud inference, since the hardware resources of an edge device can be dedicated to a single edge user. In contrast, cloud systems have to handle numerous requests simultaneously and prioritize throughput over latency, which adopt a batched strategy at the cost of increased latency for individual users. Recent advances in LLMs, especially agents and reasoning models, have increased the demands for privacy and low-latency inference, necessitating low-latency edge accelerators. However, constructing a low-latency edge accelerator remains a non-trivial task, as it faces great challenges.

Received 7 September 2025; accepted 29 September 2025. Date of publication 7 October 2025; date of current version 24 October 2025. This work was supported in part by National Natural Science Foundation of China under Grant 62488101, Grant 62495104, and Grant 62025404, and in part by Youth Innovation Promotion Association CAS. *(Corresponding author: Xiaoming Chen.)*

Xinyu Wang, Xiaotian Sun, and Wanqian Li are with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China, and also with the University of Chinese Academy of Sciences, Beijing 101408, China (e-mail: wangxinyu22s@ict.ac.cn; sunxiaotian21s@ict.ac.cn; liwanqian20s@ict.ac.cn).

Feng Min, Xiaoyu Zhang, Xinjiang Zhang, Yinhe Han, and Xiaoming Chen are with the State Key Lab of Processors, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China (e-mail: minfeng@ict.ac.cn; zhangxiaoyu@ict.ac.cn; zhangxinjiang@ict.ac.cn; yinhes@ict.ac.cn; chenxiaoming@ict.ac.cn).

Digital Object Identifier 10.1109/LCA.2025.3618104

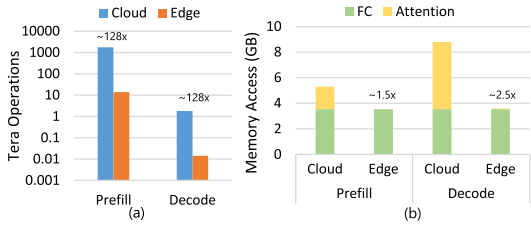

Fig. 1. Computational demands and memory access of the quantized Qwen2.5-7B model for the cloud and edge scenarios (batch size: 128 and 1, respectively). (a) Number of computation operations during the prefill phase and one decode iteration. (b) Memory access of FC and attention layers during the prefill phase and one decode iteration.

In contrast to the extensively studied cloud scenarios, edge inference exhibits distinct computational characteristics and hardware constraints, making it infeasible to directly transfer the design methodologies of cloud accelerators to low-latency edge accelerators. Fig. 1 shows the theoretical computational demands and memory access of the prefill and decode phases in both cloud and edge scenarios. Edge inference exhibits significantly lower computational demands than cloud inference but at a similar level of memory access volume. This indicates that edge accelerators should prioritize high bandwidth rather than high computational capability. Regarding memory access patterns in edge inference, fully-connected (FC) layers exhibit considerably higher memory access volume than attention layers. In addition, edge devices face stringent power and form factor constraints, requiring accelerators to be energy-efficient and compact. Given the computational characteristics and hardware constraints of edge inference, the *first challenge* of building low-latency edge accelerators is how to achieve high memory bandwidth with a limited power budget, particularly for static weights in FC layers.

Deploying LLMs on edge devices heavily relies on quantization [1], [2], which converts high-precision parameters to low-precision ones, significantly reducing storage overhead and memory access volume. QoQ [1] and QQQ [2] can quantize the model weights to 4 bits and the activations to 8 bits with minimal performance degradation, enabling integer arithmetic. QoQ can further compress the key-value (KV) cache to 4 bits. Although INT8-INT4 multiply-accumulate (MAC) units, which process 8-bit and 4-bit inputs, can meet the precision requirements of those quantization methods, current accelerators lack dedicated units and use high-precision units like INT8-INT8 units, causing unnecessary power consumption. Quantization methods present the *second challenge* for edge accelerators: how to design an energy-efficient mixed-precision unit that can leverage their theoretical advantages.

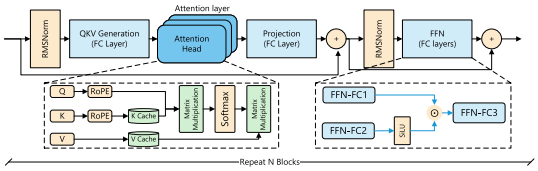

Fig. 2. Structure of a transformer block in Qwen model.

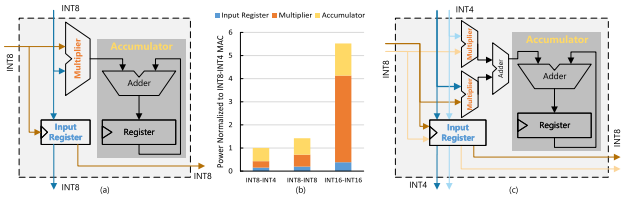

Fig. 3. (a) Structure of a trivial MAC unit. (b) Power breakdown of MAC units under different precisions. (c) Proposed energy-efficient MAC unit.

Processing-in-memory (PIM) architectures tightly integrate storage and compute units within a single die, enabling high-bandwidth and low-power intra-die data transfer, which can mitigate the bandwidth bottlenecks in edge inference. In this letter, we propose a resistive random-access memory (RRAM)-based PIM accelerator, Lyla, for low-latency edge inference. To tackle the *first challenge*, we build a PIM architecture based on RRAM devices featuring low read power consumption to meet the bandwidth demands of LLMs under low power budget. For the *second challenge*, we design an energy-efficient MAC unit placed near the RRAM array to fully exploit the benefits of quantization. Lyla consists of two types of dies: RRAM-based PIM dies for storing static weights and performing computations, and conventional dynamic random-access memory (DRAM) dies for storing the KV cache. Evaluation shows that Lyla delivers 3.8×, 2.4×, and 1.2× speedups, along with 165.1×, 10.9×, and 4.0× better energy efficiency, over a GPU and two DRAM-based PIM accelerators, respectively.

II. Background and Motivation

Model Structure: Current LLMs comprise multiple transformer blocks. Fig. 2 illustrates the structure of a Qwen2.5 [3] transformer block, shared by most open-source models. Matrix multiplications constitute the core operations within a transformer block, primarily originating from FC layers and attention layers. FC layers perform computation between the static weights and input activations, whereas attention layers require computation between activations and the continuously updated KV cache. The inference process consists of two stages: a prefill phase that processes all input tokens to produce the first output token, and a decode phase that iteratively generates one token based on the previous outputs.

MAC Unit: Systolic arrays composed of MAC units are generally used to handle matrix multiplications in LLMs. Fig. 3(a) illustrates the basic structure of a MAC unit, which consists of an input register, a multiplier, and an accumulator. In existing accelerators, such as GPUs, the multiplier within the MAC unit processes two operands of equal bit width. In LLMs' inference, although the operands can be quantized to different bit widths (such as INT8 and INT4), conventional hardware still employs multipliers with equal input bit width (such as INT8-INT8), causing extra overhead.

Why RRAM?: Various memory devices, such as DRAM and RRAM, can be employed to construct PIM accelerators. Existing works [4], [5] have applied DRAM-based PIM in LLM accelerators. IANUS [4] and H^2-LLM [5] are both heterogeneous accelerators composed of a neural processing unit (NPU) and a DRAM-based PIM, where the NPU accelerates the prefill phase and the PIM primarily handles the decode phase. IANUS enables PIM by embedding compute units within DRAM dies, while H^2-LLM leverages hybrid bonding to build high-bandwidth links between the DRAM and logic compute dies for PIM. However, RRAM has two advantages over DRAM when building PIM accelerators for edge LLM inference. First, unlike DRAM, RRAM read operations avoid charge and discharge processes, resulting in lower read power consumption (e.g., 0.256 pJ/bit [6]). This allows the RRAM-based PIM to achieve comparable internal bandwidth under a reduced power budget. Second, DRAM-based PIM requires external storage for weights when power-off and incurs reloading latency after power-on. The inherent non-volatility in the RRAM removes these overheads, benefiting intermittently powered edge devices.

LLMs impose strict demands on storage capacity, requiring memory of high storage density. 3D-RRAM can achieve 0.2 Gb/mm^2 storage density with 10 metal layers [7]. Coupled with quantization, RRAM-based PIM can satisfy the storage requirements of static weights in current LLMs. To mitigate the limited write speed and endurance of RRAM, we incorporate a conventional DRAM die to handle the small number of write operations, while RRAM is employed for the predominant static weight read operations in edge inference. To avoid the impact of inherent noise in RRAM on inference, we adopt a digital approach by using the RRAM only for storage and performing computations through near-array units, instead of using RRAM for computations in the analog domain.

III. Energy-Efficient MAC Unit

As shown in Fig. 3(b), replacing the INT8-INT8 MAC unit with a trivial INT8-INT4 unit reduces the power consumption, aligning with the power requirements of edge devices. However, the INT8-INT4 MAC unit can be further optimized to save more energy. To prevent overflow, the accumulator has to be designed with a higher bit-width than the multiplier. When the dimension of a square matrix is n, the accumulator requires at least $\log_2 n$ more bits than the product of the multiplier to ensure numerical precision. Considering that the matrices in LLMs have huge dimensions, such as 14,336 in LLaMA3-8B, an INT8-INT4 MAC unit requires at least a 26-bit accumulator to preserve accuracy. The power breakdown in Fig. 3(b) shows that, as the input precision of the MAC unit decreases, the high bit-width accumulator becomes the dominant source of power consumption. In an INT8-INT4 MAC unit, the accumulator accounts for 57% of the total power.

Fig. 3(c) shows the structure of the proposed energy-efficient MAC unit. We implement two multipliers within a single MAC unit. Their outputs are first combined by a 13-bit adder and then accumulated by a 26-bit accumulator. This design allows one

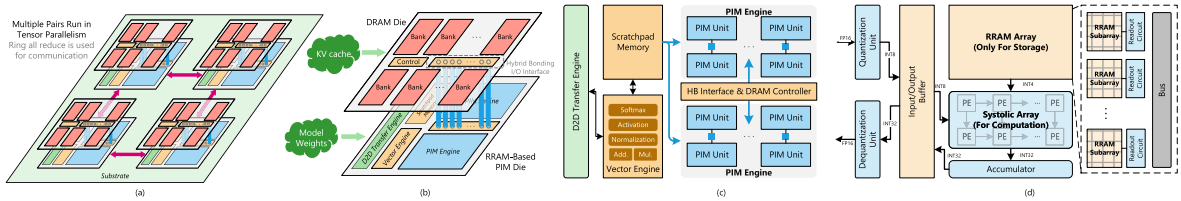

Fig. 4. (a) Overview of the accelerator Lyla, consisting of 4 pairs. (b) Architecture of a pair in Lyla. (c) Architecture of the RRAM-based PIM die in the pair. (d) Design of the PIM unit.

accumulation for every two multiplications, effectively reducing the energy consumption by halving the number of accumulation operations. Since two multiplications are performed per cycle, we reduce the frequency by half to keep the same external bandwidth requirement.

IV. RRAM-Based PIM Edge Accelerator

As shown in Fig. 4, we propose a hierarchical edge LLM accelerator. The fundamental unit is a pair that provides both storage and computation capabilities, consisting of an RRAM-based PIM die and a DRAM die. To accommodate models of different sizes, we provide three accelerators with varying capacities, Lyla-S/M/L, containing 1, 2, and 4 pairs, respectively. Tensor parallelism [8] is employed among multiple pairs in Lyla-M/L to aggregate storage and computational resources.

A. Pair Design

As shown in Fig. 4(b), a DRAM die is stacked on top of an RRAM-based PIM die through hybrid bonding (HB) [9], forming a pair. HB is a packaging technology that tightly bonds two dies vertically through Cu-to-Cu and dielectric-to-dielectric bonding, which reduces the package size by utilizing vertical space and achieves low-power data transfer through its inherent low-resistance I/O pins, aligning with the form factor and power constraints of edge devices. The PIM die is responsible for storing the model weights and computing the prefill and decode phases, while the DRAM die solely provides storage for the KV cache.

The RRAM-based PIM die consists of a PIM engine, a vector engine, a scratchpad memory, a DRAM controller, and a die-to-die (D2D) transfer engine. The PIM engine comprises multiple PIM units, each containing many RRAM subarrays that store the weights of FC layers and other static parameters. We place a systolic array composed of MAC units introduced in Section III within the PIM unit, which performs INT8-INT4 matrix multiplications for the FC and attention layers. Although integer activations are used in matrix multiplications, activations outside the PIM units remain in FP16 to preserve accuracy for other operations. Therefore, quantization and dequantization units are integrated into each PIM unit to convert between integer and floating-point data formats. The vector engine is used to execute element-wise operations and non-linear functions (e.g., vector-add, softmax, etc.). The scratchpad memory is dedicated to storing intermediate results. The DRAM controller manages the communication with the DRAM die and handles the loading and updating process of the KV cache. The D2D transfer engine has two links that connect adjacent pairs, forming a ring topology.

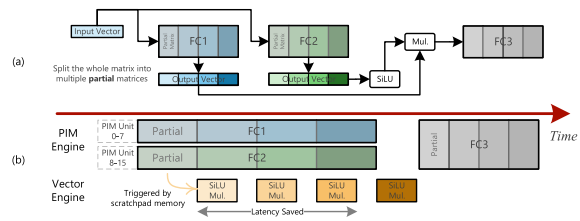

Fig. 5. (a) Computation process of the FFN block. (b) Timing diagram of PIM engine and vector engine executing the FFN block. (Only four partial results are shown for clarity.).

B. Pipeline Mechanism

To reduce execution latency, we design a pipeline mechanism. A complete computation task (e.g., matrix multiplication, element-wise vector operation) can be decomposed into multiple partial computations, each producing a partial result. The final result can be obtained by concatenating all partial results. As shown in Fig. 5(a), the weight matrix in an FC layer is divided into multiple partial matrices along the output dimension. Multiple partial matrices of the same matrix are executed sequentially. A tag manager is introduced into the scratchpad memory. When a compute engine writes the partial result to a specific address, the tag of that address will be updated, and the scratchpad memory will trigger the consumer compute engine waiting for that tag to load data for subsequent execution. Fig. 5(b) shows the pipelined execution of a feedforward network (FFN) block carried out by a pair. Initially, the PIM engine executes two partial matrices of FC1 and FC2. Once the partial results are written, the scratchpad memory notifies the vector engine to perform vector operations on them. Meanwhile, the PIM engine can process the remaining partial matrices. Consequently, a pipeline is established, allowing the PIM engine and the vector engine to operate in parallel, thereby overlapping the vector operation latency.

V. Evaluation

A. Experimental Setup

We evaluate the RRAM-based PIM die under a 12 nm process and use Synopsys Design Compiler to obtain the power and area of the proposed MAC unit. Based on the array design in [6], we scale the 5 MB bank in [6] to a 16 MB bank using 1T10R 3D RRAM without compromising energy efficiency [10]. The RRAM-based PIM die comprises 16 PIM units, each equipped with twenty 16 MB banks, resulting in a total capacity of 5 GB per die. Each PIM unit activates five banks to achieve 64 GB/s

TABLE I
COMPARISON OF DIFFERENT MAC DESIGNS

	INT8-INT8 MAC	INT8-INT4 MAC	Ours
Frequency (MHz)	1000	1000	500
Power (mW)	0.2691	0.1817	**0.1288**
Area (μm^2)	121.25	**96.88**	132.192

The bold values indicate the best results among the compared methods in terms of area and power consumption.

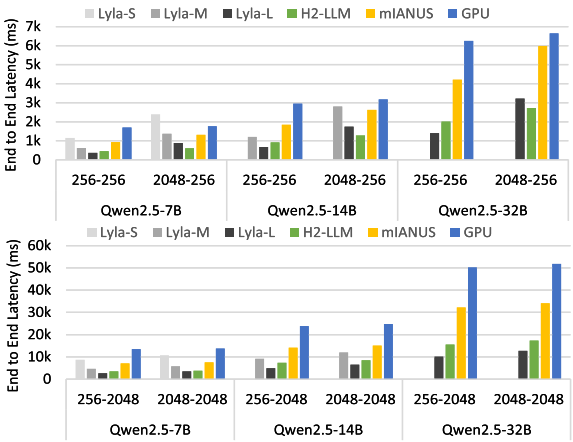

Fig. 6. End-to-End inference latency of different input and output lengths. Output length is set to 256 (top), and output length is set to 2048 (bottom).

bandwidth, delivering data to a nearby 8×128 systolic array of the proposed MAC units. We adopt the DRAM die from [9], which achieves 156 GB/s bandwidth at a power consumption of 1.2 W. We develop a cycle-accurate in-house simulator to evaluate the overall performance and power consumption of the whole accelerator.

We evaluate popular open-source models Qwen2.5-7B, Qwen2.5-14B, and Qwen2.5-32B under W4A8KV4 quantization with different input and output lengths on Lyla, a NVIDIA RTX4090 GPU, two DRAM-based PIM accelerators IANUS and H^2-LLM. The vLLM framework is used to run quantized models on the GPU. We assume that IANUS has adequate storage capacity and is equipped with INT8-INT4 MAC units to support quantized models, which is referred to as mIANUS. Effective computational capability and bandwidth are extracted from the experimental results of IANUS to estimate the performance of mIANUS. We use the same hardware configuration as in [5] to evaluate H^2-LLM. Batch size is set to 1 to better reflect real-world edge inference scenarios.

B. Experimental Results

Table I shows the evaluation of the MAC unit. As our design incorporates two multipliers, it achieves the same computational capability as a conventional MAC unit at half the frequency. Compared with the widely used INT8-INT8 MAC units, our design achieves a 52% reduction in power consumption, with only 9% extra area overhead. Compared with the trivial INT8-INT4 MAC unit, our design demonstrates a 29% advantage in power efficiency.

Fig. 6 compares the end-to-end inference latency. Lyla-L achieves average speedups of 3.8×, 2.4×, and 1.2× over the GPU, mIANUS, and H^2-LLM, respectively. For tasks with long output lengths, Lyla-L demonstrates up to 5.0× improvement over the GPU, 3.1× over the mIANUS, and 1.5× over the H^2-LLM. Fig. 7(a) and (b) illustrate the latency of the prefill phase and one decode iteration, respectively. Lyla-L achieves the lowest decode latency owing to its high internal bandwidth. For the compute-intensive prefill phase, the GPU and H^2-LLM equipped with a high-performance NPU exhibit better performance. However, for recent reasoning models that generate long outputs, low decode latency is more critical than low prefill latency. Fig. 7(c) shows the energy efficiency results of different accelerators during the decode phase. Thanks to the low read energy of RRAM and the design of energy-efficient MAC units, Lyla-L achieves 25.4 tokens/J energy efficiency during the decode phase, delivering 165.1×, 10.9×, and 4.0× improvements over the GPU, mIANUS, and H^2-LLM, respectively.

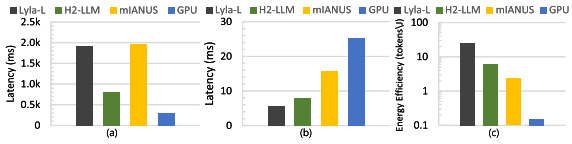

Fig. 7. (a) Latency of the prefill phase. (b) Latency of one decode iteration. (c) Energy efficiency of Qwen2.5-32B with a sequence length of (2048, 2048).

VI. CONCLUSION

From this study, we identify the key factors that influence the design of edge LLM accelerators. Providing sufficient bandwidth for FC layers is crucial for low-latency edge inference. The widely used quantization methods necessitate dedicated MAC units to fully exploit their benefits. By constructing an RRAM-based PIM die and integrating the energy-efficient MAC units, the proposed accelerator, Lyla, achieves 3.8×, 2.4×, and 1.2× speedups in performance, as well as 165.1×, 10.9×, and 4.0× improvements in energy efficiency over the GPU, modified IANUS, and H^2-LLM, respectively.

REFERENCES

[1] Y. Lin et al., "QServe: W4A8KV4 quantization and system co-design for efficient LLM serving," 2024, *arXiv:2405.04532*.
[2] Y. Zhang et al., "QQQ: Quality quattuor-bit quantization for large language models," 2024, *arXiv:2406.09904*.
[3] Qwen et al., "Qwen2.5 technical report," 2025. [Online]. Available: https://arxiv.org/abs/2412.15115
[4] M. Seo et al., "IANUS: Integrated accelerator based on NPU-PIM unified memory system," in *Proc. 29th ACM Int. Conf. Architect. Support Program. Lang. Operating Syst.*, 2024, pp. 545–560.
[5] C. Li et al., "H2-LLM: Hardware-dataflow co-exploration for heterogeneous hybrid-bonding-based low-batch LLM inference," in *Proc. 52nd Annu. Int. Symp. Comput. Archit.*, 2025, pp. 194–210.
[6] S. D. Spetalnick et al., "An edge accelerator with 5 MB of 0.256-pJ/bit embedded RRAM and a localization solver for bristle robot surveillance," *IEEE J. Solid-State Circuits*, vol. 60, no. 1, pp. 35–48, Jan. 2025.
[7] Y.-H. Huang et al., "High-density embedded 3-D Stackable via RRAM in 16-nm FinFET CMOS logic process," *IEEE Trans. Electron Devices*, vol. 71, no. 6, pp. 3614–3619, Jun. 2024.
[8] M. Shoeybi et al., "Megatron-LM: Training multi-billion parameter language models using model parallelism," 2019, *arXiv: 1909.08053*.
[9] D. Niu et al., "184QPS/W 64Mb/mm^2 3D logic-to-DRAM hybrid bonding with process-near-memory engine for recommendation system," in *Proc. 2022 IEEE Int. Solid-State Circuits Conf.*, 2022, pp. 1–3.
[10] M. Xie et al., "Monolithic 3D integration of 2D transistors and vertical RRAMs in 1T–4R structure for high-density memory," *Nature Commun.*, vol. 14, no. 1, 2023, Art. no. 5952.