CHIME: Chiplet-based Heterogeneous Near-Memory Acceleration for Edge Multimodal LLM Inference

Yanru Chen[1*], Runyang Tian[1*], Yue Pan[1], Zheyu Li[1], Weihong Xu[2#], Tajana Rosing[1]

[1]University of California San Diego, La Jolla, CA, USA

[2]Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland

{yac054, r3tian, k4fan, yup014, zhl178, tajana}@ucsd.edu; weihong.xu@epfl.ch

* Equal contribution. # Corresponding author.

Abstract—The proliferation of large language models (LLMs) is accelerating the integration of multimodal assistants into edge devices, where inference is executed under stringent latency and energy constraints, often exacerbated by intermittent connectivity. These challenges become particularly acute in the context of multimodal LLMs (MLLMs), as high-dimensional visual inputs are transformed into extensive token sequences, thereby inflating the key-value (KV) cache and imposing substantial data movement overheads to the LLM backbone. We present CHIME, a chiplet-based heterogeneous near-memory accelerator for edge MLLM inference. CHIME pairs monolithic-3D (M3D) DRAM for low-latency, bandwidth-hungry attention with M3D RRAM for dense, non-volatile weight storage, and uses a co-designed mapping framework that executes fused kernels near data to minimize cross-chiplet traffic and maximize effective bandwidth. On FastVLM (0.6B/1.7B) and MobileVLM (1.7B/3B), CHIME achieves up to $54\times$ speedup and $246\times$ energy efficiency per inference over NVIDIA Jetson Orin NX, sustaining 116.5–266.5 token/J vs. 0.7–1.1 token/J. It delivers up to $69.2\times$ higher throughput than FACIL. Compared to an M3D DRAM-only design, heterogeneous memory improves energy efficiency by 7% and performance by $2.4\times$.

Index Terms—On-device MLLMs, processing near-memory, heterogeneous memory chiplets, monolithic 3D, mapping framework

I. INTRODUCTION

On-device AI is expanding with billions of edge devices, driven by demands for latency, offline availability, privacy, and strict energy constraints. Large language models (LLMs) are reshaping human-computer interaction, and multimodal LLMs (MLLMs) extend language models to vision and audio, including medical visual question answering (VQA) and report generation [1]. However, deployment on mobile platforms is often limited to small-batch inference, which exposes a critical performance bottleneck. MLLMs further stress the memory system: visual tokens inflate the key-value (KV) cache and introduce cross-modal transfers, making throughput bandwidth-limited and energy dominated by off-chip data movement [2].

Prior work mitigates the bottleneck via software/system optimizations. Heterogeneous execution across GPUs and NPUs can improve throughput but adds synchronization, compilation overhead, and unpredictable power [3], [4]. Industry therefore reduces data movement via buffer compression, operator fusion, and low-precision quantization [5]–[8], yet the von Neumann compute–memory split remains the fundamental limiter.

Processing-in-memory (PIM) has emerged as a promising approach to mitigate the memory wall and improve energy efficiency in LLMs by embedding computation directly within memory. Early systems largely targeted single-modal LLM. Newton proposed a DRAM-based PIM architecture that integrates multiply–accumulate (MAC) units directly into DRAM and reports an average $54\times$ speedup over a Titan V GPU on BERT transformer [9]. TransPIM augments HBM with lightweight computation units and token-based dataflow, delivering $115\times$ GPU speedup and up to $667\times$ energy efficiency [10].

Recent systems extend toward multimodality. MulTCIM achieves $2.24\,\mu J/\text{Token}$ and $85.8\,\text{TOPS/W}$ by exploiting hybrid sparsity with a modal-adaptive computing in memory network [11]; StreamDCIM adopts a tile-based reconfigurable macro with a mixed-stationary dataflow, improving throughput and energy by $1.28\times$ and $1.23\times$ over layer streaming [12]; PIM–GPU collaboration further shows that mapping framework-guided task partitioning can deliver up to $15\times$ end-to-end speedup over GPU-only baselines [13].

On-device MLLMs face a unique memory bottleneck: cross-modal attention demands high bandwidth, while multimodal inputs expand the KV cache, increasing storage pressure. While DRAM-centric systems struggle to meet demands, we introduce CHIME, a chiplet-based near-memory design for edge MLLMs: monolithic 3D (M3D) DRAM serves latency-critical attention and connector kernels, and M3D RRAM provides dense storage for weights and KV under a mapping framework that reduces data movement, raises effective bandwidth, and respects RRAM endurance. Our contributions are summarized below.

- Efficient heterogeneous chiplet PIM architecture: A 2.5D advanced UCIe package links M3D DRAM for latency-critical kernels with M3D RRAM for energy-efficient storage.
- Hardware and software co-design: Our CHIME system manages weight and KV cache allocation, tiering, and migration, and schedules near-memory kernels to minimize cross-chiplet traffic and fully exploit heterogeneous memory.
- Mapping framework for general MLLMs: The mapping framework maps operators to DRAM or RRAM by access patterns, schedules processing elements (PEs), special function processing elements (SFPEs) execution, and fuses kernels to keep activations local and cut transfers.
- Speedup and energy efficiency: We compare CHIME with Jetson Orin NX on MobileVLM and FastVLM, reporting $31–54\times$ speedup and $113–246\times$ energy efficiency. Furthermore, CHIME achieves up to a $69.2\times$ higher throughput than the state-of-the-art (SOTA) PIM accelerator.

II. Motivation and Background

A. MLLM Architecture

MLLMs extend single-modal LLMs to reason over multi-modalities. As shown in Fig. 1(a), the architecture augments an LLM with a vision encoder and a connector. The encoder converts an image into feature embeddings, often with a vision transformer. The connector projects these embeddings into the language domain through multihead attention (MHA) projector or a multilayer perceptron (MLP) projector, producing pseudo-tokens. Together they form a semantic interface that transforms images into token sequences comparable to text, which are then fused and processed by the LLM backbone to generate responses. This breaks the linear scaling of single-modal LLMs by inflating the KV cache with visual tokens and increasing memory traffic during fusion.
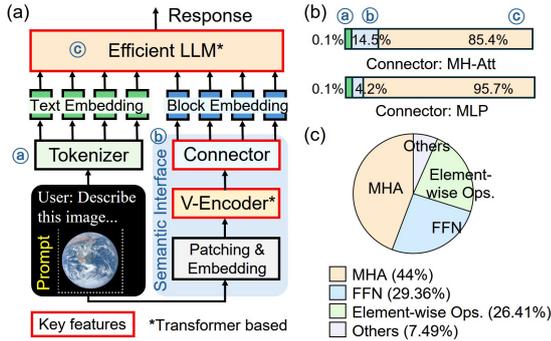

Fig. 1. MLLM software architecture overview (a) MLLM with three key features (b) Execution time breakdown of MLLMs under different connectors (c) Execution time breakdown of the GPT-2 backbone on GPU [14]

Profiling reveals that this semantic interface adds little computational load, while the LLM backbone remains the dominant cost. As Fig. 1(b) shows, the backbone accounts for 85.4%–95.7% of execution time, with the encoder and connector together consuming only 4.2%–14.5%. The dominant overhead is the transfer of large visual features from the memory hierarchy to the LLM backbone. Within the LLM backbone itself, execution is concentrated in transformer kernels, with MHA at 44% and Feed-Forward Networks (FFN) at 29.36% (Fig. 1(c)), followed by element-wise operations 26.41%.

B. Processing In Memory

PIM is broadly classified into two types: processing-using-memory (PUM) and processing-near-memory (PNM) [15]. In DRAM, PUM is achieved by carefully timed multi-row activation with charge sharing, enabling batch row-wise bitwise and majority operations [16]. PNM augments DRAM stacks with digital compute near memory, transferring data through short internal links. High-bandwidth memory (HBM) offers a natural foundation for this model, featuring a stacked architecture with a base logic die for computation.

In resistive crossbars, PUM uses stateful logic for column-parallel Boolean primitives. RACER [17] applies bit-pipelined execution across tiles to raise throughput. Analog crossbars support matrix–vector multiplication, but device variability, nonlinearity, and calibration overheads limit accuracy. Digital

PUM avoids these non-idealities, preserving exact arithmetic for MLLM inference [15]. PNM designs pair RRAM arrays with digital logic, exploiting short-range access paths and lower write energy. We adopt PNM on M3D RRAM to preserve digital accuracy while leveraging M3D stacking to raise bandwidth and capacity within thermal limits.

C. Monolithic 3D Devices

M3D sequentially builds multiple active layers on a single substrate and links them with nanometer-pitch, monolithic inter-tier vias (MIVs). For data-intensive applications, M3D integration is leveraged to develop PIM capable memories, using its fine-grain and low-latency vertical interconnects to move compute logic closer to or within memory and alleviate data movement costs [18]. Compared to through-silicon-via (TSV) stacking, MIVs provide far higher vertical interconnect density and lower capacitance and power, enabling tighter memory-compute coupling beyond 2D limits.

In this work, we focus on two complementary M3D devices. In DRAM, coarse TSV pitch limits vertical bandwidth in stacked memories such as HBM; replacing TSVs with dense MIVs in monolithic integration alleviates this bottleneck [19]–[21]. The vertical staircase layout in M3D DRAM also creates latency asymmetry across layers, enabling in-memory tiering based on access frequency. In nonvolatile memory (NVM), back end of line compatible, low-temperature processes enable RRAM stacks above CMOS logic; stacking such tiers creates short, wide data paths that enable energy-efficient near-memory computation and high-capacity storage [19], [21], [22]. CHIME combines the advantages of these two memory devices: M3D DRAM provides fast, durable access for attention and streaming, while M3D RRAM offers dense, low-leakage storage for massive weights.

D. Motivation

While SOTA PIM accelerators for LLMs mainly target DRAM bandwidth, on-device MLLMs remain memory-bound under strict power budgets. A viable system must simultaneously provide high bandwidth for attention and connector kernels, high-capacity and low-leakage storage for the growing KV cache and weights, and endurance-aware data management, while preserving digital precision and supporting variable sequence lengths. M3D DRAM and M3D RRAM provide complementary strengths for this goal. We therefore design a chiplet-based near memory platform, supported by a mapping framework that maps kernels by access locality and bandwidth, tiers the KV cache across M3D DRAM, and migrates data only when reuse outweighs transfer cost.

III. CHIME Hardware–Software Co-design

A. CHIME System Overview

CHIME integrates a heterogeneous near-memory hardware platform with a co-designed mapping framework for edge MLLM inference. The hardware platform integrates two distinct near-memory processing chiplets: M3D DRAM and M3D RRAM chiplets in a 2.5D UCIe [23] package (Fig. 2(a)), each on its own logic die with near-memory processors (NMPs). Data moves across chiplets via DMA over UCIe, while near-memory
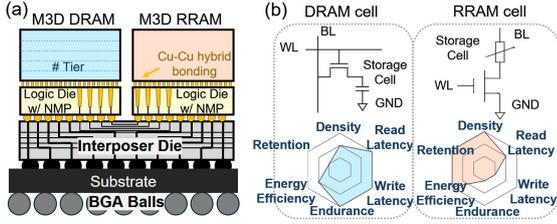
Fig. 2. CHIME system overview (a) 2.5D UCIe package integrating M3D DRAM and M3D RRAM on logic dies via interposer (b) Device-level tradeoffs of DRAM (1T1C) and RRAM (1T1R)

kernels operate in place. Our mapping framework exploits this heterogeneity with a hierarchical strategy that places data across chiplets and optimizes local execution.

- **Hardware Design**: Fig. 2(b) shows the device tradeoffs: DRAM (1T1C) is volatile, fast, and high endurance, and its inherent latency heterogeneity across vertical layers enables in-memory tiering, making it well-suited for latency-critical kernels; RRAM (1T1R) is nonvolatile and dense, well-suited for large model weights, though with higher write energy and limited endurance. The detailed hardware design and implementation are presented in Section III-B.

- **Mapping Framework**: Our framework orchestrates MLLM inference through three core strategies: ❶ workload-aware data layout that statically maps model components to the optimal memory based on MLLM profiling; ❷ KV cache tiered scheduling policy that manages attention KV cache to ensure low-latency access; and ❸ kernel locality-aware fusion that combines operations to maximize data reuse within the near-memory processors, thus minimizing intermediate data movement (Section III-C).

B. Hardware Design and Implementation

1) M3D DRAM for Latency-critical Computation The M3D DRAM handles all kernels except the FFN, covering image pre-processing, KV cache, query-key-value (QKV) weight storage, the vision encoder, the connector, and attention. The weight matrices are first placed in DRAM memory, from which data are streamed via MIVs into the logic die for computation. Fig. 3(a) illustrates the organization: each channel hosts one processing unit (PU) with shared memory for attention activations, a vector register file (VRF), a reducer, a low-latency router, a 256-way SIMD SFPE, and a group of 16 PEs. Each PE includes a matrix register file (MRF), a local controller, double-buffered memory, accumulators, and a 2×2 MAC tensor core. Double-buffering enables the tensor core to compute on one tile while transferring results from the other, effectively hiding movement latency and maximizing utilization. Activations for attention stay in the local SRAM to avoid costly write-back, while KV cache blocks are written back without frequent overwrites of the same region.

Computation begins as row buffers in each bank stream tiles of QKV to the PU. These tiles pass through the SFPE–PE pipeline, where attention scoring, softmax, and post-softmax scaling are executed directly adjacent to DRAM. Fig. 3(b) shows how each bank is organized into $1,024 \times 1,024$ MATs with a 32 Kb row buffer to supply full rows at channel bandwidth. The
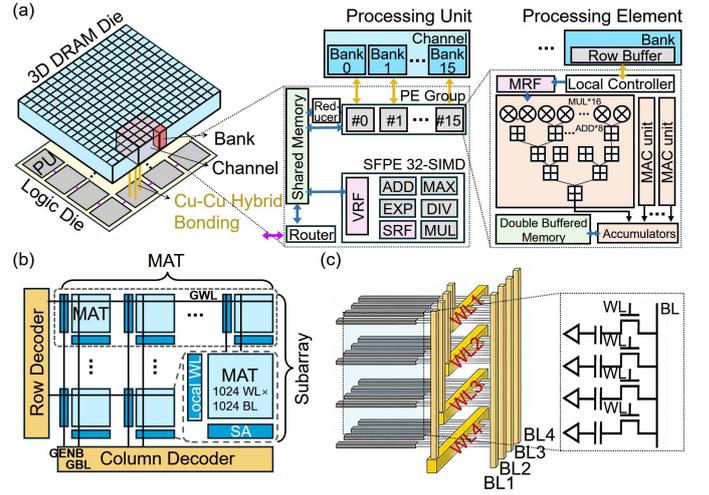

Fig. 3. M3D DRAM hardware design (a) M3D DRAM stack with NMP on logic die, organized as channels, banks, and PUs (b) Bank organization with MATs (c) Vertical M3D stacking of 1T1C subarrays
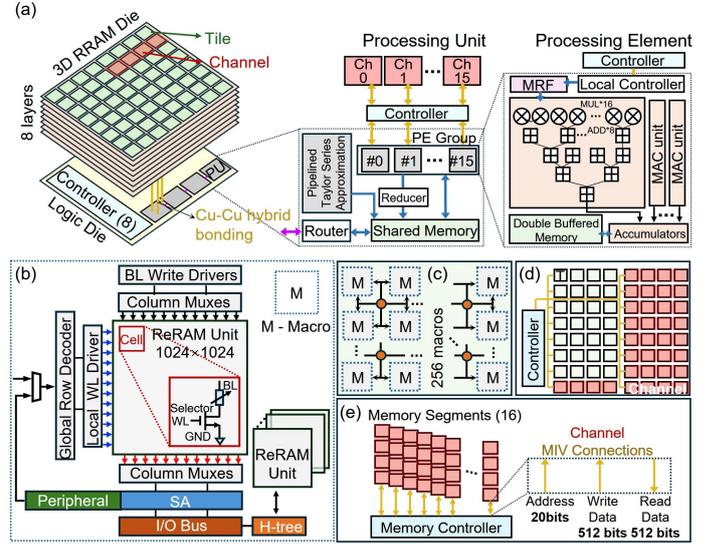

Fig. 4. M3D RRAM hardware design (a) 3D RRAM stack with NMP on logic die, organized as channels, controllers, and PUs (b) RRAM macro (c) Tile organization with local H-trees (d) Die-level layout with controllers, channels, and tiles (e) Memory subsystem with channel and tile I/O

vertical integration in Fig. 3(c) stitches bitlines (BLs) across tiers via dense monolithic links while wordlines (WLs) traverse staircases, exposing high internal bandwidth to the PU cluster. To manage the large and growing KV cache, the mapping framework employs tiered placement across the 200-layer M3D stack. Five in-memory tiers place frequently accessed blocks in lower layers, with the hottest attention data in the bottom tier (Tier-0) and connector kernels in the top tier (Tier-4). After attention computation, the attention output (AttnOut) is streamed over UCIe to the M3D RRAM, which executes the FFN kernel and returns the FFN output (FFNOut) to DRAM, enabling the next decoding step without idle cycles. Key M3D DRAM parameters are detailed in Section IV-A2.

2) M3D RRAM for Energy-efficient Storage The M3D RRAM runs the FFN kernel, where weights are resident in

the stacked arrays and later steps access them directly without reload. As shown in Fig. 4(a), eight RRAM layers sit above the logic die, each managed by a dedicated controller. The FFN input AttnOut arrives from the M3D DRAM and is distributed across 8 controllers to 16 PUs. Each pair of PUs is assigned to one RRAM layer to maximize parallel throughput. Inside each PU, wide data slices are buffered in a 1 MB SRAM that keeps activations local. A lightweight router and reducer feed 16 PEs that perform local MAC operations with weights stored in RRAM. Computation begins from the macro in Fig. 4(b), a $1\,024 \times 1\,024$ RRAM unit with local bitline drivers and column multiplexers. Fig. 4(c) tiles 256 such macros per tile and links them with 64 local H-trees for synchronous wide reads and writes. The controllers in Fig. 4(d) schedule accesses across channels and tiles, balancing thermal load and wear while ensuring that each PU receives the required weight and activation data. At the interface in Fig. 4(e), sixteen memory segments supply a 20 bit address path and 512 bit read and write datapaths that merge at the channel through M3D vertical connections. This pipeline ensures that once AttnOUT enters the logic die, it is immediately fused with preloaded weights, and the resulting FFNOut streams back to the M3D DRAM for the next decoding stage. Key parameters related to M3D RRAM hardware configuration appear in Section IV-A 2.

C. CHIME Mapping Framework

The heterogeneous chiplet design of CHIME adds complexity in data management and scheduling. A co-designed mapping framework is required to exploit the complementary strengths of high-bandwidth M3D DRAM and high-density M3D RRAM. This mapping framework deploys general MLLMs onto CHIME platform by following three design principles that minimize cross-chiplet transfers and keep computation local. Mapping a general MLLM onto this platform first requires an understanding of its key components and dataflows. Fig. 5(a) decomposes a general MLLM into the vision encoder, the connector, and the transformer backbone. The vision encoder transforms an input image into visual tokens and can be implemented as ViT [24] without downsampling that produces N tokens, PVT [25] with a four-stage pyramid downsampling, or FastViTHD [26] with five-stage downsampling that compresses to M tokens where $M \ll N$. The connector either projects visual features to pseudo tokens via an MLP, or applies cross-attention with visual KV and text Q. The LLM backbone is a transformer with attention, LayerNorm, and FFN, backed by a KV cache that grows with context length. As illustrated in Fig. 5(b), our mapping framework orchestrates the execution of general MLLM components on CHIME platform through three core principles. Fig. 5(c) summarizes the resulting implementation.

❶ **Workload-aware Data Layout.** Our data layout is governed by a strict two-cut-point dataflow designed to minimize cross-chiplet traffic between the DRAM-NMP and RRAM-NMP. The DRAM-NMP executes fused kernels for QKV projection and FlashAttention [27] streaming attention, while the RRAM-NMP holds large FFN weights and performs a fully fused FFN computation. This partitioning creates two fixed, activation-only

transfer points: (1) AttnOut (DRAM→RRAM) and (2) FFNOut (RRAM→DRAM). This enables a pipelined execution model: for a given step t, the DRAM-NMP computes $AttnOut(t)$ and streams it to the RRAM-NMP for $FFN(t)$; the next step $Attention(t+1)$ can start only after the final $FFN(t)$ output is produced. This guarantees that only the small, final AttnOut and FFNOut activations traverse the UCIe interconnect, achieving minimal cross-chiplet traffic.

❷ **KV Cache Tiered Scheduling.**

Our scheduling strategy introduces an endurance-aware KV cache tiering policy that exploits M3D in-memory tiering. KV cache blocks are classified as hot or cold based on block-level reuse distance across autoregressive decoding steps, with hot blocks placed in faster DRAM layers (Tier-0) and cold blocks in slower ones (Tier-1,...Tier-4). For extremely long contexts, the coldest blocks are offloaded to M3D RRAM in a one-shot, write-once manner. These cold KV blocks remain part of attention computation but lie outside the latency-critical path, and are written at most once per sequence and then repeatedly read, matching the read-intensive inference model where device endurance on the order of 10^{12} writes is sufficient [28].

❸ **Kernel Locality-aware Fusion.** The hardware-aware kernel fusion enables our two-cut-point dataflow, maximizing data locality within each chiplet. Moreover, by utilizing the fast on-die SRAM to temporarily store intermediate data, these fused kernels can complete their operational sequence on the logic die, eliminating costly write-backs to memory. Table I lists the fused kernels generated by the mapping framework. On the DRAM-NMP, kernels such as `FUSED_QKV_PROJ` and `FUSED_ATTN_STREAM` avoid materializing the large attention score matrix by passing partial results directly to SFPEs for online softmax and then into subsequent GEMMs within local memory. On the RRAM-NMP, the `FUSED_FFN_ACT` kernel chains two GEMMs to complete the FFN block without offloading the intermediate tensor. The key design principle is that fusion boundaries coincide with chiplet boundaries, never splitting within kernels of the same step.

IV. Evaluation

A. Experimental Setup

1) Baselines We evaluate our CHIME system using two SOTA efficient MLLMs: FastVLM [29] and MobileVLM [30]. Table II lists their model configurations. Inference is benchmarked on a VQA task with a standard input of a 512×512 astronaut image and 128 text tokens, producing 488 output tokens by default unless otherwise specified. We then compare CHIME with a SOTA PIM LLM accelerator FACIL [31] and a representative edge GPU, NVIDIA Jetson Orin NX [32].

2) Hardware Configurations CHIME co-packages two near-memory stacks with complementary roles, a 200-layer M3D DRAM (five-tier) optimized for latency-critical computation and an 8-layer M3D RRAM optimized for energy-efficient storage. Table IV reports the device, system, and NMP parameters of the M3D DRAM stack, and Table III reports the corresponding parameters of the M3D RRAM stack.

TABLE I
FUSED NEAR-MEMORY KERNELS

| Fused PIM kernel	PE and SFPE execution flow
FUSED_QKV_PROJ$(X, W_Q, b_Q, W_K, b_K, W_V, b_V)$	PE: GEMM$(X \cdot W_Q) \rightarrow$ SFPE: Add$(b_Q) \rightarrow Q$; PE: GEMM$(X \cdot W_K) \rightarrow$ SFPE: Add$(b_K) \rightarrow K^\top$; PE: GEMM$(X \cdot W_V) \rightarrow$ SFPE: Add$(b_V) \rightarrow V$
FUSED_ATTN_STREAM$(Q, K^\top, V, s)$	for each tile $(K_t^\top, V_t)$: PE: GEMM$(Q \cdot K_t^\top) \rightarrow$ SFPE: OnlineSoftmaxUpdate $\rightarrow$ PE: GEMM$(\text{Scores}_t \cdot V_t)$ with accumulate $\rightarrow Out$
FUSED_FFN_ACT(X, W_1, b_1, W_2, b_2)	PE: GEMM$(X \cdot W_1) \rightarrow$ Add$(b_1) \rightarrow$ ACT $\rightarrow$ PE: GEMM$(Y \cdot W_2) \rightarrow$ SFPE: Add$(b_2) \rightarrow Out$
FUSED_NORM(X, g, b)	SFPE: Reduce $\rightarrow$ Normalize $\rightarrow$ Scale $(\times g) \rightarrow$ Shift $(+b) \rightarrow Out$
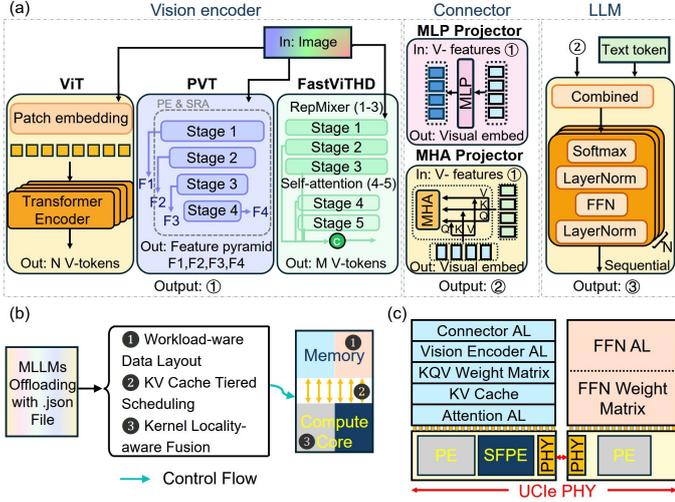
Fig. 5. MLLM dataflow and CHIME mapping (a) MLLMs abstraction with vision encoder, connector, and transformer-based LLM (b) Mapping framework pipeline with ❶ workload-aware data layout, ❷ KV cache tiered scheduling, and ❸ kernel locality-aware fusion (c) Mapping implementation

TABLE II
MLLM MODEL CONFIGURATIONS FOR EVALUATION

Model	Vision Encoder	Connector	LLM Parameters
FastVLM (0.6B)	FastViTHD	Lightweight MLP	Qwen2-0.5B
FastVLM (1.7B)	FastViTHD	Lightweight MLP	Qwen2-1.5B
MobileVLM (1.7B)	ViT	LDP	LLaMA-1.4B
MobileVLM (3B)	ViT	LDP	LLaMA-2.7B

3) Evaluation Platform We develop an in house simulator for the CHIME system. We model the 3D RRAM arrays and peripheral circuits with NeuroSim [33] to ensure device-level accuracy, and we synthesize the Register Transfer Level (RTL) design in SystemVerilog using Synopsys Design Compiler with a 40 nm CMOS PDK at 1 GHz. The synthesis included the tensor core PE, SFPE, and the on-chip controller. All results are scaled to 7nm technology node using established models [34].

B. Speedup and Energy Efficiency Analysis

Fig. 6(a) compares CHIME with Jetson Orin NX [32] on MobileVLM [30] and FastVLM [29]. CHIME achieves a $\sim$41$\times$ speedup (arithmetic-mean; 31–54$\times$ across models) and an energy-efficiency gain of $\sim$185$\times$ (arithmetic-mean; 113–246$\times$ across models). The gains are larger for the smaller variants in each family (MobileVLM 1.7B vs. 3B; FastVLM 0.6B vs. 1.7B). Their working dimension fits far more fully in the near-memory tiers with less on chip data movement, sustaining PE - SFPE pipelines and reducing UCIe data travel, while the baseline remains limited by off-chip DRAM traffic. Fig. 6(b) illustrates throughput, measured in tokens per second (TPS), and power. Jetson draws 7–11 TPS at 7 − 13 W while CHIME delivers 233–533 TPS at around 2 W, yielding roughly 40$\times$ higher throughput at 5$\times$ lower power. This indicates Jetson is

TABLE III
HARDWARE CONFIGURATIONS OF 3D RRAM

3D RRAM Device Parameters			
#Layer	8	Technology Node	28 nm CNFET
Unit Size	1k×1k	Unit/Tile	256
Read Latency	2.3 ns	Write Latency	11 ns
Read Energy	0.4 pJ/bit	Write Energy	1.33 pJ/bit
3D RRAM System Parameters			
Chip Capacity	2 GB	Internal Parallelism	128 channels
Organization	8 controllers; 16 channels/controller; 4 tiles/channel		
Interconnect	64 H-trees connect 256 units within a tile		
Interface BW	Peak BW = 512 GB/s (8 controllers × 512 bit × 1 GHz)		
Processing Element (PE)			
Tensor Core	4×4 MACs	Double Buffered SRAM	8 KB
Processing Unit (PU)			
#PEs	16	Shared Memory	80 KB
Special Function PE	None	Ring Router	128 GB/s/link
RRAM NMP Processor			
Basic	7 nm process; 0.7 V supply; 33.6 mm² die area; FP16 format.		
#PUs	16	SRAM Capacity	1 MB
Peak Performance	32 TFLOPS	Peak Power	2.584 W

TABLE IV
HARDWARE CONFIGURATIONS OF 3D DRAM

3D DRAM Device Parameters			
#layers	200	Technology Node	35 nm
MAT Size	1k×1k	#MATs/Bank	200
Bank Capacity	200 Mb	Bank Area	0.439 mm²
Row Buffer	32 Kb	Read/Write Energy/bit	0.429 pJ
Chip Area	121 mm²	Read/Write Latency/ns	(3+0.8*L) ns
3D DRAM System Parameters			
Tier Design	5 tiers (L1, L2, L3, L4, L5); 1.25 GB capacity per tier.		
Organization	16 channels per chip (64b data I/O per channel); 16 banks per channel.		
Processing Element (PE)			
Tensor Core	2×2 MACs	Double Buffered SRAM	1 KB
Processing Unit (PU)			
#PEs	16	Shared Memory	20 KB
Special Function PE	256-way SIMD	Ring Router	128 GB/s/link
DRAM NMP Processor			
Basic	7 nm process; 0.7 V supply; 121 mm² die area; FP16 format.		
#PUs	16	SRAM Capacity	512 Kb
Peak Performance	2 TFLOPS	Peak Power	0.671 W

memory-stalled on MLLM inference, whereas CHIME converts its tight power budget into effective bandwidth.

We then compare CHIME with a SOTA PIM LLM accelerator FACIL [31] and the representative edge GPU, NVIDIA Jetson Orin NX [32] (Table V). The comparison with the SOTA PIM accelerator FACIL further underscores CHIME's advantages (see Table V). While FACIL reaches 19.3 token/s at 38.5 W, CHIME's throughput of $233 - 533$ token/s represents a $12.1\times$ to $69.2\times$ throughput leap at a lower power envelope of ~ 2 W. This significant improvement highlights the high performance of CHIME's heterogeneous memory architecture and co-designed mapping framework in alleviating the memory bottleneck more effectively than other PIM-based approaches.

C. Area and Power Overhead

Fig. 7(a) and (b) show the logic die area breakdown. In M3D DRAM, peripherals use 51.5%, the UCIe PHY 22.3%, and PUs 26.2%. In M3D RRAM, larger tensor cores and double buffered SRAM raise the PU share to 34.0%. The total logic die area is smaller, $24.85 \, \text{mm}^2$ versus $28.71 \, \text{mm}^2$, with lower

TABLE V
PERFORMANCE COMPARISON OF EDGE AI PLATFORMS

Specification	Jetson Orin NX [32]	FACIL [31]	CHIME
Design	GPU	Near-bank DRAM	Our work
Technology Node (nm)	8	15	28&35
Frequency (GHz)	≤0.92	≤3.2	1
Die Area (mm²)	∼200	∼200	28.71&24.85
Power (W)	10-40	5.7-38.5	2
Throughput (Token/s)	7.4-11	7.7-19.3	233-533
Energy Eff. (Token/J)	0.28-0.74	0.50-1.35	116.5-266.5
Hardware Eff. (Token/s/mm²)	0.037-0.055	0.039-0.097	4.35-9.95
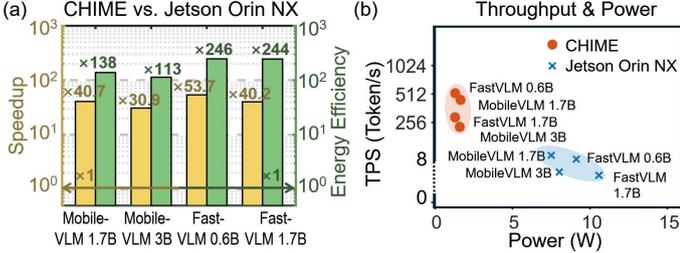
Fig. 6. Performance comparison over Jetson Orin NX [32] baseline across MobileVLM [30] and FastVLM [29] models (a) Speedup and energy efficiency comparison (b) Throughput and power comparison

peripheral cost. The split matches roles: DRAM PEs emphasize buffering for attention and connector kernels. RRAM PEs devote more compute and registers for high throughput FFN streaming. Fig. 7(c) and (d) show the power breakdown. RRAM dominates because it runs the data-intensive FFN. DRAM runs attention at lower power. Power stays stable across models, which implies utilization drives power more than model size.

D. Sensitivity Analysis

Real deployments face wide variability in prompt length and memory hierarchy choices, both directly determine KV-cache traffic, bandwidth pressure, and thus performance on edge devices. We therefore study two sensitivities of CHIME: input sequence length and memory configuration, to expose the interaction between workload scaling and memory organization.

1) Sequence Length. Fig. 8(a) and (b) show accelerator-centric latency (end-to-end delay in ms) and total energy per inference (in Joules) as the text length grows from 128 to 4k tokens for MobileVLM 1.7B/3B and FastVLM 0.6B/1.7B. Across models, both metrics increase almost linearly with length (roughly an order-of-magnitude from 128 to 4k), consistent with decoding that streams the KV cache every step. Larger models exhibit steeper slopes: MobileVLM 3B and FastVLM 1.7B incur the higher latency and energy due to wider QKV/FFN matrices and a larger KV footprint per token, while FastVLM 0.6B is lowest. At short contexts the gaps narrow as vision encoding and connector dominate, while at long contexts decoding dominates and model size is the main driver.

2) Memory Configuration. Fig. 9 validates the heterogeneous architecture by comparing CHIME to a DRAM-only baseline, demonstrating speedups of 2.38–2.49× and energy efficiency gains of 1.04–1.07×. The speedup is significant for larger models, especially MobileVLM 3B whose FFN weights overwhelm DRAM-centric M3D DRAM. By offloading these capacity-oriented weights to RRAM, CHIME preserves critical DRAM bandwidth and fast access capability for latency-critical kernels, keeping the PE - SFPE fed to sustain higher throughput.
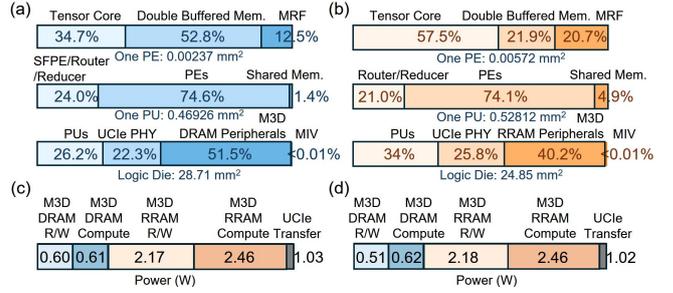

Fig. 7. CHIME logic-die area and power breakdown (a) Area breakdown of M3D DRAM (b) Area breakdown of M3D RRAM (c) Power breakdown of FastVLM (0.6B) (d) Power breakdown of MobileVLM (1.7B)
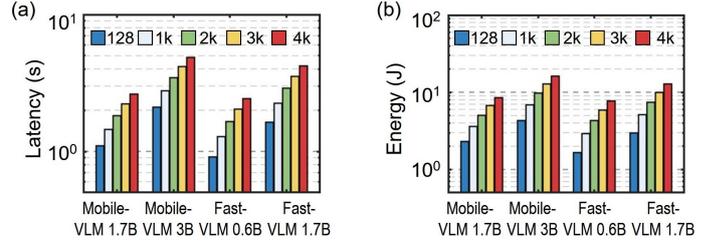

Fig. 8. Impact of sequence length: from 128 to 4k (a) Latency vs. sequence length (b) Energy vs. sequence length
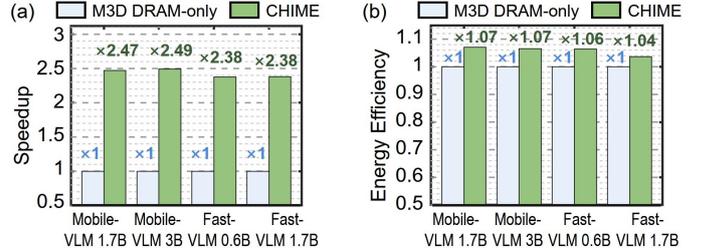

Fig. 9. Impact of memory configuration: CHIME vs. M3D DRAM-only (a) Speedup comparison (b) Energy efficiency comparison

V. CONCLUSION

CHIME tackles the fundamental memory bottleneck of edge MLLMs with a 2.5D near-memory design that pairs M3D DRAM and M3D RRAM. DRAM executes latency-critical kernels with tiered KV placement, while RRAM stores dense FFN weights for non-volatile, low-leakage access. A workload-aware mapping framework co-optimizes data layout, KV scheduling, and kernel fusion to maximize locality and effective bandwidth. On FastVLM and MobileVLM, CHIME delivers up to 41.4× speedup and 185.3× energy efficiency over Jetson Orin NX (up to 266.5 token/J), and outperforms FACIL by up to 69.2× in TPS. Compared to an M3D DRAM-only design, CHIME improves performance by 2.4× and energy efficiency by 7%, showing that reserving DRAM bandwidth for attention while offloading weights to RRAM is the right edge tradeoff.

REFERENCES

[1] G. Qu, Q. Chen, W. Wei, Z. Lin, X. Chen, and K. Huang, "Mobile edge intelligence for large language models: A contemporary survey," 2025.

[2] H. Li, Y. Li, A. Tian, T. Tang, Z. Xu, X. Chen, N. Hu, W. Dong, Q. Li, and L. Chen, "A survey on large language model acceleration based on kv cache management," 2025.

[3] L. Chen, D. Feng, E. Feng, R. Zhao, Y. Wang, Y. Xia, H. Chen, and P. Xu, "Heterollm: Accelerating large language model inference on mobile socs platform with heterogeneous ai accelerators," *arXiv preprint arXiv:2501.14794*, 2025.

[4] D. Xu, H. Zhang, L. Yang, R. Liu, G. Huang, M. Xu, and X. Liu, "Fast on-device llm inference with npus," in *ASPLOS '25*. Rotterdam, Netherlands: ACM, 2025, pp. 1–18.

[5] Y. Wang, H. Li, and X. Li, "Re-architecting the on-chip memory sub-system of machine-learning accelerator for embedded devices," in *ICCAD*, 2016.

[6] Apple Machine Learning Research, "Apple neural engine (ane) transformers," GitHub repository, 2022, "up to $10\times$ faster and $14\times$ lower peak memory".

[7] "Unlocking on-device generative ai with an npu and heterogeneous computing," Qualcomm Technologies, Inc., Tech. Rep., 2024.

[8] "Generative ai phone industry whitepaper," MediaTek Inc. and Counterpoint Research, Tech. Rep., 2024, emphasizes heterogeneous execution, APU support for INT4/INT8/INT16/FP16, and memory capacity/bandwidth needs for on-device LLMs.

[9] M. He, C. Song, I. Kim, C. Jeong, S. Kim, I. Park, M. Thottethodi, and T. Vijaykumar, "Newton: A dram-maker's accelerator-in-memory (aim) architecture for machine learning," in *MICRO*. IEEE, 2020, pp. 372–385.

[10] M. Zhou, W. Xu, J. Kang, and T. Rosing, "Transpim: A memory-based acceleration via software-hardware co-design for transformer," in *HPCA*. IEEE, 2022, pp. 1071–1085.

[11] F. Tu, Z. Wu, Y. Wang, W. Wu, L. Liu, Y. Hu, S. Wei, and S. Yin, "Multcim: Digital computing-in-memory-based multimodal transformer accelerator with attention-token-bit hybrid sparsity," *JSSC*, vol. 59, no. 1, pp. 90–101, 2023.

[12] S. Qin, Z. Qiang, Z. Fan, W. Li, X. An, X. Ye, and D. Fan, "Streamdcim: A tile-based streaming digital cim accelerator with mixed-stationary cross-forwarding dataflow for multimodal transformer," in *ISCAS*. IEEE, 2025.

[13] S. Ji, C. Liu, Y. Ding, Q. Liao, and Z. Tang, "A real-time execution system of multimodal transformer through pim-gpu collaboration," in *DAC*, 2024.

[14] W. Han, H. Cho, D. Kim, and J.-Y. Kim, "Sal-pim: A subarray-level processing-in-memory architecture with lut-based linear interpolation for transformer-based text generation," *IEEE TC*, 2025.

[15] O. Mutlu, S. Ghose, J. Gómez-Luna, and R. Ausavarungnirun, "A modern primer on processing in memory," in *Emerging computing: from devices to systems: looking beyond Moore and Von Neumann*. Springer, 2022, pp. 171–243.

[16] V. Seshadri, D. Lee, T. Mullins, H. Hassan, A. Boroumand, J. Kim, M. A. Kozuch, O. Mutlu, P. B. Gibbons, and T. C. Mowry, "Ambit: In-memory accelerator for bulk bitwise operations using commodity dram technology," in *MICRO*, 2017, pp. 273–287.

[17] M. S. Truong, E. Chen, D. Su, L. Shen, A. Glass, L. R. Carley, J. A. Bain, and S. Ghose, "Racer: Bit-pipelined processing using resistive memory," in *MICRO*, 2021, pp. 100–116.

[18] A. K. Ramanathan, S. S. Rangachar, H. T. Govindarajan, J.-M. Hung, C.-Y. Lee, C.-X. Xue, S.-P. Huang, F.-K. Hsueh, C.-H. Shen, J.-M. Shieh *et al.*, "Cim3d: Comparator-in-memory designs using monolithic 3-d technology for accelerating data-intensive applications," *JxCDC*, vol. 7, no. 1, pp. 79–87, 2021.

[19] A. M. Felfel, K. Datta, A. Dutt, H. Veluri, A. Zaky, A. V.-Y. Thean, and M. M. S. Aly, "Quantifying the benefits of monolithic 3d computing systems enabled by tft and rram," in *2020 DATE*. IEEE, 2020, pp. 43–48.

[20] K. Dhananjay, P. Shukla, V. F. Pavlidis, A. Coskun, and E. Salman, "Monolithic 3d integrated circuits: Recent trends and future prospects," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 3, pp. 837–843, 2021.

[21] Y. Pan, Z. Xia, P. Hsu, L. Hu, H. Kim, J. Sharda, M. Zhou, N. S. Kim, S. Yu, T. Rosing, and M. Kang, "Stratum: System-hardware co-design with tiered monolithic 3D-stackable DRAM for efficient MoE serving," in *MICRO*, New York, NY, USA, Oct. 2025.

[22] Y. Du, J. Tang, Y. Li, Y. Xi, Y. Li, J. Li, H. Huang, Q. Qin, Q. Zhang, B. Gao *et al.*, "Monolithic 3d integration of analog rram-based computing-in-memory and sensor for energy-efficient near-sensor computing," *Advanced Materials*, vol. 36, no. 22, p. 2302658, 2024.

[23] M.-S. Lin, C.-C. Tsai, S. Li, W.-C. Chen, W.-H. Huang, Y.-C. Chen, Y.-J. Huang, A. Drake, C.-H. Wen, P. Ranucci *et al.*, "36.1 a 32gb/s 10.5 tb/s/mm 0.6 pj/b ucie-compliant low-latency interface in 3nm featuring matched-delay for dynamic clock gating," in *ISSCC*, vol. 68. IEEE, 2025, pp. 586–588.

[24] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[25] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao, "Pyramid vision transformer: A versatile backbone for dense prediction without convolutions," in *ICCV*.

[26] P. K. A. Vasu, J. Gabriel, J. Zhu, O. Tuzel, and A. Ranjan, "Fastvit: A fast hybrid vision transformer using structural reparameterization," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 5785–5795.

[27] T. Dao, D. Fu, S. Ermon, A. Rudra, and C. Ré, "Flashattention: Fast and memory-efficient exact attention with io-awareness," *Advances in neural information processing systems*, vol. 35, pp. 16344–16359, 2022.

[28] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, "Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 14–26, 2016.

[29] P. K. A. Vasu, F. Faghri, C.-L. Li, C. Koc, N. True, Antony *et al.*, "Fastvlm: Efficient vision encoding for vision language models," in *CVPR*, 2025, pp. 19769–19780.

[30] X. Chu, L. Qiao, X. Zhang, S. Xu, F. Wei, Y. Yang, Y. Hu, X. Lin, B. Zhang *et al.*, "Mobilevlm v2: Faster and stronger baseline for vision language model," *arXiv preprint arXiv:2402.03766*, 2024.

[31] S. H. Seo, J. Kim, D. Lee, S. Yoo, S. Moon, Y. Park, and J. W. Lee, "Facil: Flexible dram address mapping for soc-pim cooperative on-device llm inference," in *HPCA*. IEEE, 2025, pp. 1720–1733.

[32] NVIDIA Corporation, "Jetson Orin NX Series Datasheet," NVIDIA Corporation, Tech. Rep. DS-10712-001_v0.5, September 2022, accessed: 2025-09-13. [Online]. Available: https://developer.nvidia.com/docs

[33] X. Peng, S. Huang, H. Jiang, A. Lu, and S. Yu, "Dnn+ neurosim v2. 0: An end-to-end benchmarking framework for compute-in-memory accelerators for on-chip training," *IEEE TCAD*, vol. 40, no. 11, pp. 2306–2319, 2020.

[34] A. Stillmaker and B. Baas, "Scaling equations for the accurate prediction of cmos device performance from 180 nm to 7 nm," *Integration*, vol. 58, pp. 74–81, 2017.