

Focus Session: Hardware/Software Co-Design to Accelerate Generative AI Workloads on Heterogeneous Architectures

Pratyush Dhingra, Vibhanshu Sharma, Janardhan Rao Doppa, and Partha Pratim Pande
Washington State University, Pullman, WA, USA

Abstract— Effective acceleration of Generative AI workloads is constrained by heterogeneous computational and memory characteristics. Large Language Models (LLMs), for example, are composed of diverse kernels, ranging from memory-bound attention mechanisms to compute-intensive feed-forward networks. Such variability leads to poor resource utilization and renders homogeneous architectures inefficient. This paper outlines a methodology to address this challenge using two synergistic approaches. First, we discuss the design of a three-dimensional (3D) manycore architecture enabled by heterogeneous integration. Specifically, the architecture leverages emerging non-volatile memory alongside CMOS-based computing cores to create an optimized kernel-to-compute mapping. Second, we explore a hardware and software co-design framework to enable fine-tuning of unimodal and multimodal LLMs. This framework ensures thermally robust inference when leveraging emerging non-volatile memories within the heterogeneous architecture. Experimental results on multiple LLMs demonstrate the efficacy of the 3D heterogeneous architecture and the co-design framework.

Keywords— LLMs, Heterogeneous Integration, Co-design.

I. INTRODUCTION

Generative AI has revolutionized natural language processing, with Large Language Models (LLMs) delivering state-of-the-art performance across diverse applications [1]. Built on the transformer architecture, LLMs leverage the self-attention mechanisms to capture long-range dependencies, achieving exceptional accuracy and generalization capabilities [1]. However, these large models demand enormous computational and memory resources, making their deployment challenging in edge environments [2]. Consequently, conventional compute-centric platforms, such as GPUs, suffer from suboptimal performance and low hardware utilization due to the memory wall problem [3] [4].

Memory-centric architectures have emerged as an alternative computing paradigm [5]. These architectures perform energy-efficient computation within the memory to eliminate unnecessary data movement. Specifically, Processing-in-memory (PIM)-enabled architectures have been proposed for accelerating Deep Neural Networks (DNNs) [5][6] [7]. PIM technologies can be broadly classified into two categories: volatile and non-volatile PIM. Volatile PIM technologies include devices such as Static Random-Access Memory (SRAM) and Dynamic Random-Access Memory (DRAM) [8]. Non-volatile memory (NVM) technologies leverage emerging devices such as ferroelectric

field-effect transistor memory (FeFET), magnetic random-access memory (MRAM), resistive random-access memory (ReRAM), etc. [7]. Each of these technologies presents unique trade-offs in terms of power, area, latency, endurance, and device variability [9]. For instance, NVM-based PIM solutions are both area and energy-efficient but suffer from limited endurance and device variability. Conversely, volatile PIM, such as SRAM, does not suffer from endurance issues or device variability but incurs significant area overhead and high leakage power in comparison to NVM-based PIM [9].

Unlike simpler DNNs that have homogeneous computing kernels, transformer model layers exhibit significant heterogeneity. For instance, the attention mechanism needs dynamic operand multiplications even during inference, whereas the feed-forward network involves multiplication with static pre-trained weights. The dynamic computation on homogeneous NVM-based PIM architectures would necessitate multiple write operations to NVM cells. It is well known that NVMs suffer from limited write endurance [10]. Consequently, architectures relying exclusively on NVM PIM are inadequate for LLM deployment [11]. Therefore, it becomes necessary to consider heterogeneity in the computing substrate and address the limitations of standalone memory-centric or compute-centric architectures to develop a hardware accelerator for LLMs .

2.5D/3D integration enables combining multiple heterogeneous cores within a single package [12] [13]. Recently, several studies have proposed heterogeneous integration for transformer models, focusing on suitable kernel-to-device mapping [11] [13] [14] [15]. However, these architectures suffer from significant limitations when considering diverse scenarios at the edge, including model fine-tuning. Transformer based foundation models pretrained using high-coverage data need to be customized to achieve high predictive accuracy for specific applications upon deployment [4]. Some examples include fine-tuning an LLM, such as GPT for question answering, and fine-tuning LLMs for a specific domain (e.g., finance, biology). Unlike inference, fine-tuning is computationally intensive and memory-heavy, requiring back-propagation for gradient updates. Current 2.5D/3D heterogeneous architectures are not well optimized to address the challenges during fine-tuning or to ensure high model accuracy under the thermal constraints of emerging NVM based PIM.

In this paper, we discuss the current state-of-the-art for hardware and software co-design on heterogeneous manycore architectures to accelerate generative AI workloads. We first discuss the architectural characteristics of LLMs, highlighting the diverse computational and memory requirements. Building on this, we describe the design of a heterogeneous architecture and a suitable computational kernel to core mapping. Next, we discuss various challenges associated with enabling model fine-tuning at the edge and thermally robust inference. Finally, we outline the future scope and discuss challenges associated with billion-parameter model inference.

This work was supported by the US National Science Foundation (NSF) grant CSR-2308530, the Army Research Office under Grant ARO-W911NF-24-1-0240 and the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research program under project 84245—Democratization of Co-design for Energy-Efficient Heterogeneous Computing (DeCoDe) at Pacific Northwest National Laboratory (PNNL). PNNL is a multi-program national laboratory operated for the U.S. Department of Energy (DOE) by Battelle Memorial Institute under Contract No. DE-AC05-76RL01830.

II. LLM ARCHITECTURE AND ACCELERATORS

In this section, we present a brief overview of the features of LLM computational kernels and the associated diverse computational requirements. Next, we discuss the need for heterogeneity and a hardware/software co-design approach.

A. LLM Computation Kernels

An LLM is primarily composed of multiple sequential layers of identical blocks. The input text is first tokenized and mapped to vectors within an embedding matrix to generate token embeddings. These embeddings, combined with positional encodings, serve as inputs (X) to the model. Structurally, each decoder layer contains two primary functional blocks: multi-head attention (MHA) and feed-forward (FF) network [1]. Fig. 1 illustrates the LLM decoder layer. The MHA module employs multiple attention heads in parallel, each computing attention scores [1]. This is followed by the FF module, a fully connected network that processes data sequentially through linear layers [1]. Table I details the fundamental computational kernels along with the associated operations in the original transformer architecture. Here, d_{model} represents the dimensionality of the model architecture and n represents the input token length. We adhere to the conventional terminology used to describe the transformer architecture. For instance, Q, K, V and S represent the query, key, value, and attention score vectors.

The MHA computation involves multiplication with trainable weight matrices (W^Q, W^K, W^V , and W^O) each typically featuring a dimension $d_{model} \times d_{model}$. The other MHA computations include multiplications with dynamic operands and non-linear softmax and normalization operations. The FF network accounts for a larger parameter count compared to MHA with dense linear layers. Here, the hidden dimension is usually much larger than the model dimension (e.g. $4 * d_{model}$ in standard transformer architecture). Additionally, each functional module is associated with a normalization layer (such as LayerNorm or RMSNorm) to ensure training stability.

The original transformer model with encoder-decoder blocks is designed for machine translation tasks [1]. However, the transformer architecture is continuously evolving with structural variations tailored to other NLP tasks. Notable architectural variations include models that are exclusively composed of decoder blocks (e.g. GPTs), which are used for text generation. Furthermore, significant structural optimizations have been introduced within individual functional blocks. One such advancement is the introduction of Multi-Query Attention (MQA) in the attention module [2]. MQA, unlike the standard MHA, shares a single set of K and V values across heads while using distinct Q vectors. MQA is explicitly designed to reduce the high memory overhead of the attention mechanism. Another variation is the Multi-Latent Attention (MLA) proposed by DeepSeek, which creates a

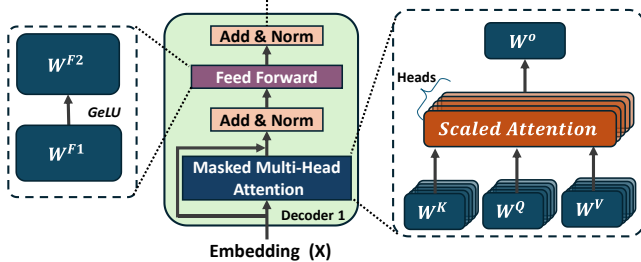


Fig. 1: Architectural breakdown of a transformer decoder layer, highlighting the FF and MHA block.

Table I. Computational Kernels in Transformer Model

	Multi-Head Attention (MHA)	Operations
MHA-1	$Q, K, V = X \cdot W^Q, X \cdot W^K, X \cdot W^V$	$3 \times O(d_{model}^2 \cdot n)$
MHA-2	$S = Q \cdot K^T$	$O(d_{model} \cdot n^2)$
MHA-3	$P = Softmax(S) \cdot V$	$O(d_{model} \cdot n^2)$
MHA-4	$H_m = P \cdot W^O$	$O(d_{model}^2 \cdot n)$
Feed-Forward (FF)		
FF-1	$X^1 = GeLU(M \cdot W^{F1})$	$4 \times O(d_{model}^2 \cdot n)$
FF-2	$X^2 = GeLU(X^1 \cdot W^{F2})$	$4 \times O(d_{model}^2 \cdot n)$

low-rank representation of the K and V matrices to improve memory efficiency [2]. Additionally, modern LLMs like Llama modify the FF layer by employing three layers instead of two with SwiGLU as the activation function [1]. These variations (encoder/decoder-only, MQA, and MLA) must be considered when designing a model-agnostic accelerator.

B. Limitations/Challenges of LLM Accelerators

Currently, GPUs and Google's Tensor Processing Units (TPUs) based on systolic arrays are the predominant platforms for LLM acceleration [1] [4] [16]. However, both these suffer from low utilization with computing cores waiting for data [3] [4]. Additionally, the cost of data transfer far exceeds that of the arithmetic operations, leading to suboptimal performance and poor energy efficiency [17]. This has led to research into alternative hardware platforms to accelerate transformer models. Recent approaches include a diverse range of implementations using FPGA, PIM, and heterogeneous systems integrating NVM-based PIM and SRAM.

Homogeneous NVM-based PIM architectures to accelerate transformer inference include ReTransformer, AccelTran and others [18] [19] [20]. As mentioned earlier, standalone NVM PIM architectures are not suitable for transformer workloads due to the limited write endurance inherent to NVM devices. Volatile PIM architectures include DRAM-based PIM accelerator (e.g. TransPIM) with compute units integrated within HBM banks [21]. However, TransPIM suffers from low overall system utilization for dynamic operand multiplication, such as attention computation. Consequently, heterogeneous accelerators have also been developed to address the shortcomings of homogeneous accelerators. HAIMA, for instance, adopts a hybrid strategy by incorporating SRAM units and DRAM [14]. Similarly, H3D-Transformer proposes a 16-tier hybrid architecture stacked vertically using TSVs. It combines FeFET, SRAM, and TPU cores in a single system for inference [15].

Despite these advancements, existing works primarily optimize for inference and fail to consider model fine-tuning, a critical requirement for domain adaptation at the edge. Model fine-tuning introduces complex data dependencies and requires gradient updates (back-propagation), which are computationally intensive compared to inference. Further, existing literature often overlooks the optimization of the Network-on-Chip (NoC), which is crucial for facilitating efficient data exchange and supporting the traffic among heterogeneous computing cores. Hence, a hardware/software co-design approach is necessary for the acceleration of emerging GenAI workloads.

III. HARDWARE/SOFTWARE CO-DESIGN

In this section, we discuss the 3D heterogeneous architecture and present the corresponding design strategy for accelerating both transformer inference and fine-tuning.

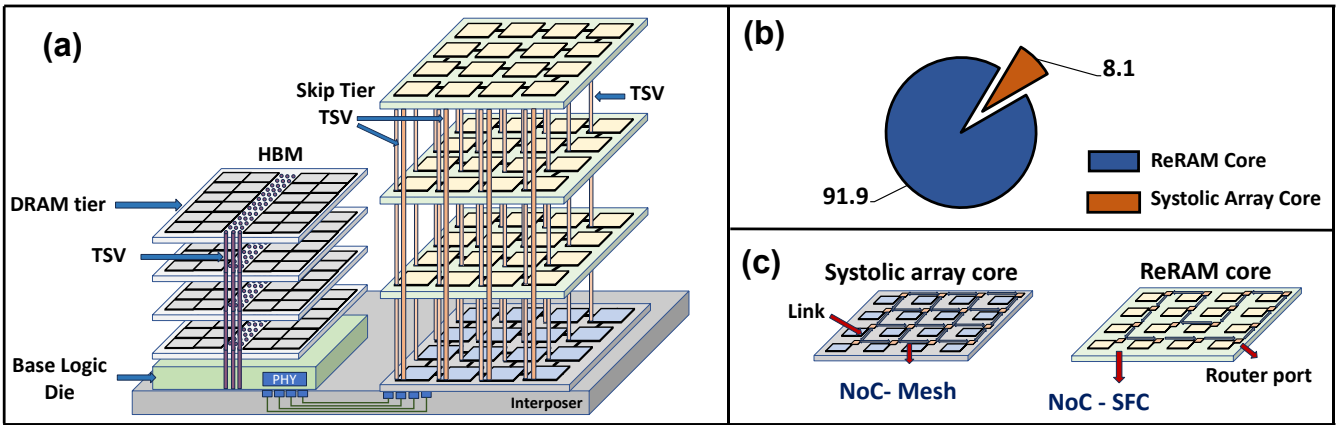


Fig. 2: (a) Atleus architecture consisting of 3 ReRAM tiers and 1 systolic array tier connected vertically through TSV-based interconnect. The memory access to DRAM is through the interposer via 2.5D integration, (b) The NoC design in Atleus for each type of computing core, and (c) The compute operations breakdown among ReRAM and systolic array cores in percentage within Atleus for GPT-2 (Medium).

A. Heterogeneous Architecture Design

An effective transformer model acceleration requires handling two distinct types of multiplication: 1) involving stationary operands; and 2) with dynamic operands. Multiplications with static weights, such as those in the FF network, can be performed effectively on high-density PIM devices (e.g. NVM or DRAM). Conversely, tasks involving dynamic operands require input-dependent computations and frequent read-modify-write operations. These are unsuitable for NVM PIM due to write endurance limitations and high write latency. Further, DRAM PIM suffers from higher access latencies compared to on-chip logic and low resource utilization for dynamic computation [14]. Thus, CMOS-based compute units, such as systolic arrays or SRAM-based PIM, should be employed for these dynamic computations.

Heterogeneous Integration (HI) offers a compelling architectural solution to address this computational diversity. The advantages of using HI for LLMs are twofold. First, the weight-stationary computation, which represents most of the compute operations, is performed in an energy-efficient manner by utilizing high-density PIM devices. Second, the limitations of NVM and DRAM cells with dynamic operand computation is mitigated by offloading these computations to CMOS-based computing cores. Building on this, several three-dimensional (3D)-HI architectures consisting of NVM devices and CMOS-based computing cores have been developed, such as H3D-Transformer and Atleus [22]. Moreover, HAIMA is a notable HI architecture that employs a system incorporating a host, SRAM PIM, and DRAM connected via a 2.5D interposer [14].

As a representative example, Fig. 2(a) shows Atleus, a 3D-HI manycore architecture consisting of ReRAM-based NVM devices and systolic array cores [22]. Here, ReRAM-based NVM is used for multiplication with weight-stationary operands. ReRAM crossbars perform N^2 multiplications in constant time making it extremely energy efficient [17]. In addition, systolic arrays are utilized for dynamic operands. The CMOS-based systolic array cores are equipped with Special Function Units (SFUs) to execute non-linear computational kernels such as layer normalization and softmax. Note that due to manufacturing constraints, these disparate technologies, i.e. the systolic array and ReRAM cores, are implemented on separate tiers. The tiers are vertically interconnected with Through Silicon Via (TSV)-

based links [12] [23] [24]. The intermediate activations generated during model inference/fine-tuning necessitate significant dynamic memory capacity. Thus, off-chip DRAM access is enabled through an interposer via 2.5D integration to store these activations [25]. The DRAM illustrated in Fig. 2(a) is High Bandwidth Memory (HBM), which is 3D-DRAM technology specifically designed to offer high bandwidth.

Fig. 2(b) illustrates the computational breakdown between the ReRAM and systolic array cores in terms of compute load on Atleus for GPT-2 (Medium) model as an example. This analysis assumes a sequence length of 1024, which is the maximum allowable sequence length for GPT-2 (Medium). From Fig. 2(b), we observe that ReRAM cores are utilized for 91.9% of the computations in Atleus. This underscores the effectiveness of HI where we leverage the advantages of PIM for most (more than 90%) of the computations and address the memory wall.

B. NoC Design in Atleus

It is essential to design 3D-HI enabled manycore architecture with a low-latency interconnect backbone to achieve high-performance. Specifically, the cores communicating with each other must be placed in physical proximity to minimize the communication latency. The compute mapping detailed in the previous subsection for Atleus distributes the MHA and FF computation across the heterogeneous resources. The model weights are stored on-chip and spatially distributed across the ReRAM cores. Since activations propagate sequentially from layer i to layer $i + 1$, placing consecutive layers far apart will result in long-range multi-hop communication. Therefore, it is essential to ensure continuity in the ReRAM core placement to the extent possible to reduce communication overhead. Existing work has shown that space-filling curve (SFC) effectively maintains contiguity between neural layers in 2.5D chiplet systems [26]. Specifically, the incorporation of an SFC ensures single-hop connectivity for data exchange, unlike conventional topologies such as mesh or torus. Thus, an SFC topology should be adopted to connect the cores in each ReRAM tier. Conversely, the systolic array tier requires orthogonal flow of both input data and activations, alongside frequent DRAM accesses. Hence, a conventional mesh-based NoC is best suited to effectively manage the traffic on the

systolic array. Fig. 2(c) depicts the suitable interconnection on the ReRAM and systolic array cores respectively.

The heterogeneous computing cores also need to exchange activations among each other. The ReRAM cores store the W^Q, W^K and W^V matrices to generate the corresponding Q, K and V vectors. These vectors are transmitted to the systolic array for attention score computation. Following MHA, the activations are sent to the ReRAM cores for FF network processing. The physical distance between cores in the vertical direction is significantly less compared to the horizontal direction [27]. Consequently, the intra-layer computing cores are mapped in the vertical direction to minimize the communication latency. However, this vertical mapping would generate skip-tier communication between ReRAM and systolic array cores. Specifically, the ReRAM cores may have to skip multiple ReRAM tiers to send activations to the systolic array. Therefore, we need long-range shortcuts between the ReRAM tiers and the systolic array tier to prevent multi-hop communication. Atleus addresses this by implementing TSV-based long-range links to facilitate energy-efficient skip-tier communication. Fig. 2(a) highlights the skip-tier links in the NoC design in Atleus.

We now examine the effectiveness of NoC design in Atleus against baselines. The configurations under consideration are:

- 1) **3D-mesh**: A standard 3D mesh utilizing by TSV links. It serves as a reference point for comparison.
- 2) **3D-mesh with skip**: A modified 3D-mesh design where we incorporate additional TSV-skip links connecting the top and bottom tier. These links provide shortcuts to support intra-layer traffic.
- 3) **Atleus**: Atleus utilizes an SFC to connect cores in the ReRAM tier given the unidirectional flow of activations among consecutive layers. Conversely, a mesh interconnect is employed for the systolic array tier. Similar to the second configuration, it integrates vertical skip-tier TSVs connecting the top and bottom layers.

Fig. 3(a) presents the distribution of router radices across the evaluated NoC architectures. The profile for the second configuration (3D-mesh with skip) exhibits a pronounced expansion along the axes for larger router sizes (size 5 and size 6), reflecting the increased port counts necessitated by the additional skip links. In contrast, Atleus demonstrates a contraction in the high-radix domain. Its footprint shifts away from router size 6 and skews heavily towards router size 4. This shift can be attributed to the reduced number of horizontal links from the SFC implementation to connect cores within ReRAM tiers. Specifically, each core has a planar link only to its immediate predecessor and successor along the SFC path. Next, Fig. 3(b) shows the energy-delay product (EDP) and the associated area for the three NoC configurations. We consider BERT-Large transformer model for this analysis. EDP is used as the evaluation metric to simultaneously capture both performance and energy in a single term. From Fig. 3(b), we observe that integrating additional TSV links to a mesh leads to EDP improvement by 12% but also results in an increase in area by 16%. In contrast, the Atleus NoC delivers superior efficiency, achieving a 27% reduction in EDP with a 4% area increase. The use of skip links helps in energy-efficient skip-tier communication and the reduced number of horizontal links helps to offset the area overhead incurred.

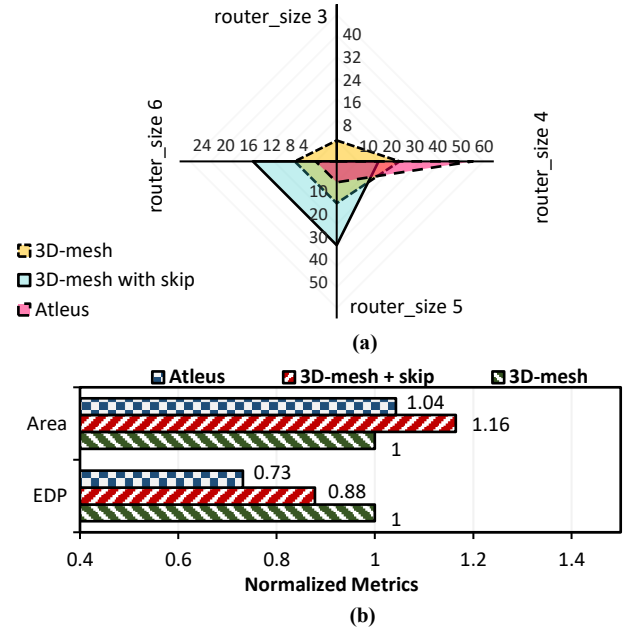


Figure 3: Quantitative comparison of NoC topologies presenting (a) router port size, (b) Normalized EDP & area.

C. LLM Fine-Tuning

Fine-tuning is essential for adapting foundation models to specific domains upon deployment. However, fine-tuning is computationally expensive and memory-intensive. Further, it involves the back-propagation algorithm, which computes gradients and updates weights. A full parameter fine-tuning would require writes due to weight updates on NVM cells in heterogeneous accelerators employing NVM PIM. This would raise endurance concerns due to the limited write endurance of emerging NVM technologies.

Recently, several Parameter-Efficient Fine-Tuning (PEFT) techniques have emerged to fine-tune models efficiently. These strategies cover a broad spectrum of techniques, ranging from prompt tuning and the introduction of biases to updating input embeddings or making a subset of parameters trainable [28] [29] [30]. Among these, Low-rank Adaptation (LoRA) is a state-of-the-art technique that works by introducing trainable low-rank matrices to modulate the weights of the pre-trained model [31]. Fig. 4 illustrates a model layer with LoRA where only the low-rank matrices are updated during the fine-tuning process, keeping the original model's parameters unchanged (frozen). Eq. 1 shows the LoRA computation kernel.

$$Y = WX = (W_0 + AB)X = W_0X + ABX \quad (1)$$

Here, Y is the output with input being X , and W is the weights after fine-tuning. The fine-tuned weights are the sum

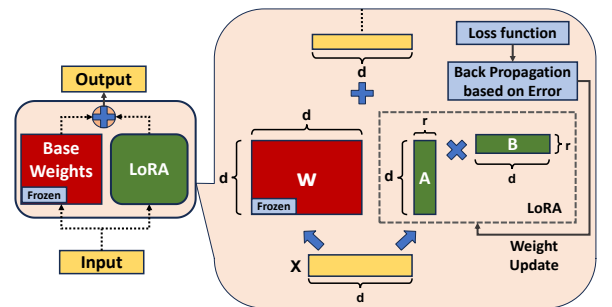


Figure 4: An overview of a model layer with low-rank adapters added to the pre-trained weights for fine-tuning.

of the frozen weights ($W_o \in R^{d \times d}$) and the product of low-rank matrices (AB). $A \in R^{d \times r}$ and $B \in R^{r \times d}$ are the two trainable low-rank matrices where " r " is the rank and " d " represents the pre-trained weight matrix dimension. These low-rank matrices are much smaller in size compared to the original weight matrix since the rank r is significantly smaller than the model dimension d [31]. This transformation changes the computational complexity of the trainable parameters from $O(d^2)$ to $O(d.r)$, reducing the memory footprint and computational load during back-propagation. For instance, existing works have shown that the value of r in the range of 8 to 32 is sufficient for fine-tuning a model with $d_{model} = 2048$ [31].

LoRA can be used to decouple the static and dynamic computation during fine-tuning similar to the mapping presented earlier for transformer inference. To address the memory and endurance challenges, Atleus can be employed with a hardware-aware mapping where frozen base-model parameters (W_o^l) remain on ReRAM cores, while trainable LoRA parameters (A^l and B^l) are mapped to systolic array cores. This mapping strategy alters the forward pass: input activations are sent to the ReRAM crossbars (where frozen weights reside) and to the systolic array cores (where low-rank matrices are mapped). Now, the output activation is obtained by the summation of the MVM operations involved in both computations. This introduces additional computation on systolic arrays, specifically MVM operations involving inputs and LoRA parameters. However, this increase in forward pass computation is minimal due to $r \ll d$.

During back-propagation, we calculate the gradient of the loss L with respect to the trainable A^l and B^l parameters. Note that the gradients for the frozen matrix (W_o^l) are zero. Thus, gradients are calculated only for parameters mapped to systolic array. This helps prevent write operations to the ReRAM cells during fine-tuning associated with weight updates. Further, the computational mapping offers significant advantages over compute-centric architectures (such as GPUs or systolic arrays), which must fetch both frozen and LoRA parameters from DRAM. In contrast, frozen parameters remain stationary on-chip in Atleus on PIM cores. Fig. 5 illustrates the mapping strategy on Atleus and highlights the difference when compared to systolic arrays.

Furthermore, the mapping strategy allows efficient switching between tasks during inference within the 3D-HI architecture. For multiple inference tasks, the computational kernel mapping remains the same as fine-tuning where the

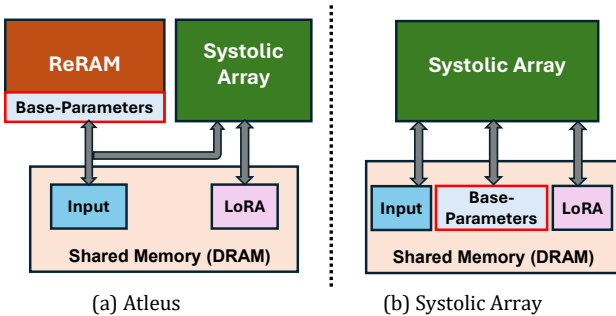


Figure 5: High-level illustration of the computational kernel to compute mapping on heterogeneous compute and memory-centric architecture (like Atleus) and compute-centric architectures such as systolic array for LoRA-based fine-tuning.

TABLE II. NORMALIZED EXECUTION TIME

	Roberta-Base		BERT-Large	
	Squad	SWAG	Squad	SWAG
ATLEUS	1 ×	1 ×	1 ×	1 ×
HAIMA	8.61 ×	12.06 ×	8.42 ×	9.95 ×
3D-TPU	13.42 ×	16.77 ×	15.54 ×	17.87 ×
GPU	19.37 ×	23.58 ×	23.54 ×	25.49 ×

TABLE III. NORMALIZED ENERGY

	Roberta-Base		BERT-Large	
	Squad	SWAG	Squad	SWAG
ATLEUS	1 ×	1 ×	1 ×	1 ×
HAIMA	11.07 ×	16.36 ×	10.83	13.50 ×
3D-TPU	20.13 ×	27.30 ×	23.31	29.17 ×
GPU	31.15 ×	40.26 ×	37.83	44.32 ×

pre-trained model weights are stored on ReRAM cores and task-specific LoRA adapters are stored off-chip in DRAM. Now, switching between inference tasks requires fetching only the lightweight LoRA parameters, which represent a negligible fraction of the total model size. This is not feasible for compute-centric architectures, which typically necessitate reloading the entire model alongside the LoRA adapters.

We now show a comparative analysis of Atleus with the above-mentioned mapping against HAIMA, a 3D-TPU (systolic array), and an NVIDIA Tesla V100 GPU during model fine-tuning. The 3D-TPU baseline is a four-tier system featuring 128×128 systolic array connected via a 3D-torus interconnect following Google TPU-v4 [16]. To ensure consistency, we maintain the same DRAM data access latency and energy parameters across our evaluation.

Table II and Table III illustrate the normalized execution time and energy, respectively for Roberta-Base and BERT-Large models as examples on two fine-tuning workloads [32]. The results highlight significant inefficiencies in the baseline architectures. HAIMA utilizes the host for executing the score computation, specifically the softmax operation on the output resulting from $Q.K^T$ multiplication performed in HBM [14]. This design choice results in substantial communication latency during fine-tuning. Unlike inference, multiple transformer layers share the same interposer links for data exchange with the host or SRAM cores leading to high communication delay. Considering compute-centric GPU architecture, we observe less than 50% core utilization highlighting the memory bottlenecks. The 3D-TPU performs $\sim 2 \times$ faster than GPUs as the 3D architecture performs better with the complex dataflow during fine-tuning. However, 3D-TPU also suffers from sub-optimal utilization and high data transfer costs from memory. Conversely, Atleus demonstrates consistent performance gains over these baselines, driven by its optimized interconnect backbone and strategic compute mapping.

D. Thermally Robust LLM Inference

While 3D-HI architectures offer compelling advantages for LLM acceleration, they introduce significant thermal management challenges. The vertical stacking of compute tiers creates both horizontal and vertical thermal gradients across the system, with tiers farther from the heat sink experiencing elevated temperatures [13]. This thermal variation is particularly problematic for NVM-based PIM devices, where temperature-dependent conductance drift distorts stored weight values, leading to degraded predictive accuracy. This necessitates robust noise-mitigation

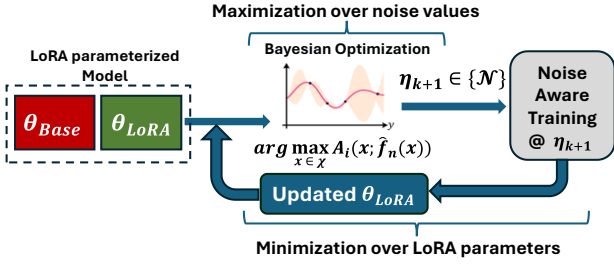


Fig. 6: An illustration of the min-max optimization framework for thermal NAT. Here, η denotes the noise level, as determined by the Bayesian optimizer.

methodologies. However, current state-of-the-art noise-aware training (NAT) approaches fail to provide consistent inference accuracy across the entire operating temperature range (e.g. 300K – 400K) [33] [34] [35] [36] [37].

As discussed in Section III, the LoRA parameters reside on thermally-resilient CMOS-based cores (e.g. systolic arrays in Atrius), while base model weights are mapped to NVM cores. This heterogeneous placement presents an opportunity for thermal robustness. The key insight is that thermal robustness can be formulated as a secondary objective during the fine-tuning process. One can jointly optimize LoRA parameters for both fine-tuning and thermal noise resiliency. Since LoRA matrices reside on thermally-stable tiers, they can learn to compensate for temperature-induced variations in the base model weights stored on NVM crossbars.

The challenge lies in the continuous nature of the temperature range. Since there are infinitely many temperature points within this range, training the LoRA parameters at every possible temperature point (thermal NAT) is computationally prohibitive. Moreover, naive sequential training across discrete temperature points suffers from catastrophic forgetting, where the model loses robustness at earlier trained temperature conditions [38] [39]. Additionally, choosing which temperatures to train on and in what sequence become critical considerations. To address these challenges, tools such as Bayesian Optimization (BO) can be leveraged to efficiently navigate the thermal noise space during fine-tuning [40]. We can frame thermal-NAT problem as a min-max optimization shown in Eq. 2, where the outer minimization operates over LoRA parameters and the inner maximization searches across thermal noise configurations. The objective is to identify LoRA adaptations that maintain minimal prediction error under adversarial noise conditions (see Fig. 6). This approach alternates between employing BO to identify worst-case noise instances that maximize model error and updating the LoRA parameters to minimize loss under these adversarial scenarios. The approach is formulated as:

$$\min_{\theta_{LoRA}} \max_{\eta \in \mathcal{N}} \mathcal{L}(f(X; \theta_{base} + \eta, \theta_{LoRA}), Y) \quad (2)$$

Here, θ_{LoRA} are the trainable low-rank parameters, θ_{base} are the original model parameters, η represents the noise in the noise space \mathcal{N} and X, Y are the input and target output for the model. Formulation in Eq. 2 ensures that LoRA parameters mitigate the most adversarial noise. This approach eliminates the inefficiency of training across all thermal noise levels by adaptively selecting a representative subset that is sufficient for robustness across the entire range [38].

By storing these thermally-aware LoRA parameters on CMOS tiers, the architecture achieves robust inference regardless of NVM tier temperature variations. The base model weights on NVM remain frozen, avoiding write endurance concerns, while the LoRA parameters (trained to

compensate for thermal effects) maintain consistent accuracy. This co-design approach of combining architectural heterogeneity with intelligent training enables practical deployment of LLMs at the edge [38].

IV. FUTURE SCOPE

In this section, we identify future key areas of research for enabling multi-billion parameter LLM inference at the edge.

A. High-Density PIM

Existing NVM PIM capabilities are limited to a few hundred MBs, which is significantly smaller than the GBs required by modern LLMs [41]. As a result, when tasked with running billion-parameter LLMs, these accelerators are forced to rely heavily on off-chip DRAM [42]. FeFET-based NAND architectures can help alleviate this on-chip storage bottleneck [43] [44]. The FeFET-based NAND architecture leverages the mature, high-density, and cost-effective vertical stacking technology of Flash memory, enabling multi-GB on-chip storage capacity [43] [45]. Importantly, FeFET operates at significantly lower voltages ($\sim 3V$) compared to Flash memory ($\sim 20V$) [43] [46]. This voltage reduction directly translates to lower power consumption and faster operation, with FeFET achieving write speeds of around 50-100 ns ($\sim \mu s$ for Flash) [43] [47]. Further, recent 3D FeFET NAND architectures have shown excellent thermal stability and retention capabilities [48] [49]. Future work should explore FeFET-based NAND to enable multi-billion parameter LLMs.

B. Phase-Aware Mapping

LLM inference involves a compute-intensive "prefill" phase and a memory-intensive "decode" phase. The prefill step is computationally intensive due to full-sequence attention across all tokens [50]. Here, the Key (K) and Value (V) tensors for all tokens are calculated and stored in the Key-Value (KV) cache. Following prefill, the model generates output tokens one at a time in an autoregressive loop in the decode phase [50]. During this phase, the attention mechanism retrieves the K and V tensors from memory for all previously generated tokens. Afterwards, it computes new Q vectors and produces the next token. Crucially, each decode step requires access to the entire KV cache values stored in DRAM. For instance, a 7 billion parameter model requires 14 gigabytes (GB) of memory for weights alone, and its dynamic KV cache can consume over 2.45 GB for every 5,000 tokens of context. This results in frequent DRAM access during decode phase, which makes it memory bound [3]. As model size grows, the memory and compute requirement differences are further compounded in the two phases. Hence, a phase-aware compute mapping and dynamic KV cache management will be critical for addressing these scaling challenges.

V. CONCLUSION

The deployment of GenAI workloads at the edge is currently hindered by the diverse compute and memory requirements. In this paper, we presented the design of heterogeneous memory and compute-centric architecture for accelerating GenAI models. By synergizing the complementary strengths of emerging PIM technology with the conventional CMOS-based compute units, we can create a suitable computing substrate for Generative AI workloads. We also discussed a hardware/software co-design framework that leverages LoRA to enable model fine-tuning and thermally robust inference, supporting practical deployment of LLMs at the edge.

REFERENCES

- [1] T. Lin, Y. Wang, X. Liu and X. Qiu, "A Survey of Transformers," in *arXiv preprint arXiv:2106.04554*, 2021.
- [2] DeepSeek-AI, "DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model," *arXiv preprint arXiv:2405.04434*, 2024.
- [3] M. Khairy, A. G. Wassal and M. Zahran, "A survey of architectural approaches for improving GPGPU performance, programmability and heterogeneity," *J. Parallel Distrib. Comput.*, vol. 127, p. 65–88, 2019.
- [4] N. P. Jouppi et al., "In-datacenter performance analysis of a tensor processing unit," *ISCA*, 2017.
- [5] A. Gebregiorgis et al., "A Survey on Memory-centric Computer Architectures," *ACM JETC*, vol. 18, 2022.
- [6] P. Dhingra et al., "ERGo: Energy-Efficient Hybrid Graph Neural Network Training on Processing-in-Memory Architectures," *ACM TECS*, vol. 24, 2025.
- [7] H. Tsai et al., "Architectures and Circuits for Analog-memory-based Hardware Accelerators for Deep Neural Networks," *ISCAS*, 2023.
- [8] K. Roy, I. Chakraborty, M. Ali, A. Ankit and A. Agrawal, "In-Memory Computing in Emerging Memory Technologies for Machine Learning: An Overview," in *DAC*, 2020.
- [9] Y. Yu et al., "Compute-in-Memory Chips for Deep Learning: Recent Trends and Prospects," *IEEE Circuits and Systems Magazine*, 2021.
- [10] P. Dhingra et al., "FARe: Fault-aware GNN training on ReRAM-based PIM accelerators," in *DATE*, 2024.
- [11] S. Sridharan, J. Stevens, K. Roy and A. Raghunathan, "X-Former: In-Memory Acceleration of Transformers," *IEEE TVLSI*, 2023.
- [12] R. Agarwal et al., "3D Packaging for Heterogeneous Integration," *ECTC*, 2022.
- [13] P. Dhingra et al., "HeTraX: Energy Efficient 3D Heterogeneous Manycore Architecture for Transformer Acceleration," in *ISLPED*, 2024.
- [14] Y. Ding et al., "HAIMA: A Hybrid SRAM and DRAM Accelerator-in-Memory Architecture for Transformer," in *DAC*, 2023.
- [15] Y. Luo and S. Yu, "H3D-Transformer: A Heterogeneous 3D (H3D) Computing Platform for Transformer Model Acceleration on Edge Devices," in *ACM TODAES*, 2024.
- [16] N. P. Jouppi et al., "TPU v4: An Optically Reconfigurable Supercomputer for Machine Learning with Hardware Support for Embeddings," *arXiv preprint arXiv:2304.01433*, 2023.
- [17] A. Shafiee et al., "ISAAC: a convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in *ISCA*, Seoul, Korea, 2016.
- [18] X. Yang, B. Yan, H. Li and Y. Chen, "ReTransformer: ReRAM-based Processing-in-Memory Architecture for Transformer Acceleration," in *ICCAD*, San Diego, 2020.
- [19] S. Tuli and N. K. Jha, "AccelTran: A Sparsity-Aware Accelerator for Dynamic Inference with Transformers," *IEEE TCAD*, 2023.
- [20] M. Kang, H. Shin and L. -S. Kim, "A Framework for Accelerating Transformer-Based Language Model on ReRAM-Based Architecture," *TCAD*, 2022.
- [21] M. Zhou, W. Xu, J. Kang and T. Rosing, "TransPIM: A Memory-based Acceleration via Software-Hardware Co-Design for Transformer," in *HPCA*, 2022.
- [22] P. Dhingra, J. R. Doppa and P. P. Pande, "Atleus: Accelerating Transformers on the Edge Enabled by 3D Heterogeneous Manycore Architectures," *IEEE TCAD*, 2025.
- [23] S. Van Huylenbroeck et al., "Small Pitch, High Aspect Ratio Via-Last TSV Module," *IEEE ECTC*, 2016.
- [24] M. F. Chen, F. C. Chen, W. C. Chiou and D. C. H. Yu, "System on integrated chips (SoIC(TM)) for 3D heterogeneous integration," *ECTC*, 2019.
- [25] C. -C. Lee et al., "An Overview of the Development of a GPU with Integrated HBM on Silicon Interposer," *ECTC*, 2016.
- [26] H. Sharma et al., "Florets for Chiplets: Data Flow-aware High-Performance and Energy-efficient Network-on-Interposer for CNN Inference Tasks," *ACM Transactions on Embedded Computing Systems*, 2023.
- [27] W. R. Davis et al., "Demystifying 3D ICs: the pros and cons of going vertical," *IEEE Design & Test of Computers*, 2005.
- [28] X. Lisa Li and P. Liang, "Prefix-Tuning: Optimizing Continuous Prompts for Generation," *arXiv preprint arXiv:2101.00190*, 2021.
- [29] Z. Lin, A. Madotto and P. Fung, "Exploring Versatile Generative Language Model Via Parameter-Efficient Transfer Learning," *arXiv preprint arXiv:2004.03829*.
- [30] E. Ben Zaken, S. Ravfogel and Y. Goldberg, "BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models," *arXiv preprint arXiv:2106.10199*, 2022.
- [31] E. J. Hu et al., "LoRA: Low-Rank Adaptation of Large Language Models," in *ICLR*, 2022.
- [32] P. Rajpurkar, J. Zhang, K. Lopyrev and P. Liang, "SQUAD: 100,000+ Questions for Machine Comprehension of Text," *arXiv preprint arXiv:1606.05250*, 2016.
- [33] M. V. Beigi and G. Memik, "Thermal-aware optimizations of reRAM-based neuromorphic computing systems," in *DAC*, 2018.
- [34] H. Shin et al., "A thermal-aware optimization framework for ReRAM-based deep neural network acceleration," in *ICCAD*, 2020.
- [35] Z. He et al., "Noise Injection Adaption: End-to-End ReRAM Crossbar Non-ideal Effect Adaption for Neural Network Mapping," in *DAC*, 2019.
- [36] J. Meng et al., "Temperature-Resilient RRAM-Based In-Memory Computing for DNN Inference," *MICRO*, vol. 42, 2022.
- [37] X. Liu et al., "HR3 AM: A Heat Resilient Design for RRAM-based Neuromorphic Computing," in *ISLPED*, 2019.
- [38] V. Sharma et al., "ThRive: Thermally Robust CNN Inference via Low-Rank Adaptation in Heterogeneous PIM Architectures," *ACM TODAES*, vol. 31, 2025.
- [39] X. Li et al., "Learn to Grow: A Continual Structure Learning Framework for Overcoming Catastrophic Forgetting," in *arXiv preprint arXiv:1904.00310*, 2019.
- [40] M. Binois and N. Wycoff, "A Survey on High-dimensional Gaussian Process Modeling with Application to Bayesian Optimization," *Transactions on Evolutionary Learning and Optimization*, vol. 2, 2022.
- [41] D. Bridarolli et al., "High-Density Multilevel 3D Vertical Resistive Switching Memory (VRRAM) for Massively Parallel in-Memory Computing," in *IEEE International Electron Devices Meeting*, 2024.
- [42] H. Tsai et al., "Analog AI Accelerators for Transformer-based Language Models: Hardware, Workload, and Power Performance," in *IEEE International Memory Workshop*, 2025.
- [43] W. W. Shim and S. Yu, "Ferroelectric Field-Effect Transistor-Based 3-D NAND Architecture for Energy-Efficient on-Chip Training Accelerator," *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, 2021.
- [44] H. Zhong et al., "Fe-GCN: A 3D FeFET Memory Based PIM Accelerator for Graph Convolutional Networks," in *ISVLSI*, 2023.
- [45] A. Goda, "3-D NAND Technology Achievements and Future Scaling Perspectives," *IEEE Transactions on Electron Devices*, 2020.
- [46] J. V. Houdt, "3D Memories and Ferroelectrics," in *IEEE International Memory Workshop*, 2017.
- [47] K. Florent et al., "Vertical Ferroelectric HfO₂ FET based on 3-D NAND Architecture: Towards Dense Low-Power Memory," in *IEEE International Electron Devices Meeting*, 2018.
- [48] S. Kabuyanagi et al., "A Vertical Channel-All-Around FeFET with Thermally Stable Oxide Semiconductor Achieving High $\Delta I_{on} > 2\mu A/cell$ for 3D Stackable 4F2 High Speed Memory," in *IEEE Symposium on VLSI Technology and Circuits*, 2024.
- [49] S. -H. Kuk et al., "Superior QLC Retention (10 Years, 85°C) and Record Memory Window (12.2 V) by Gate Stack Engineering in Ferroelectric FET: from "MIFIS" to "MIKFIS"," in *IEEE International Electron Devices Meeting*, 2024.
- [50] Y. Zhong et al., "DistServe: Disaggregating Prefill and Decoding for Goodput-optimized Large Language Model Serving," *arxiv preprint arxiv:2401.09670*, 2024.