

KVNAND: Efficient On-Device Large Language Model Inference Using DRAM-Free In-Flash Computing

Lishuo Deng, Shaojie Xu, Jinwu Chen, Changwei Yan,
Jiajie Wang, Zhe Jiang, and Weiwei Shan
Southeast University
Nanjing, China

Email: {dengls, 220246825, 230228386, yancw, 220256672, 101013615, wwshan}@seu.edu.cn

Abstract—Deploying large language models (LLMs) on edge devices enables personalized agents with strong privacy and low cost. However, with tens to hundreds of billions of parameters, single-batch autoregressive inference suffers from extremely low arithmetic intensity, creating severe weight-loading and bandwidth pressures on resource-constrained platforms. Recent in-flash computing (IFC) solutions alleviate this bottleneck by co-locating weight-related linear computations in the decode phase with flash, yet still rely on DRAM for the key-value (KV) cache. As context length grows, the KV cache can exceed model weights in size, imposing prohibitive DRAM cost and capacity requirements. Attempts to offload KV cache to flash suffer from severe performance penalties.

We propose KVNAND, the first DRAM-free, IFC-based architecture that stores both model weights and KV cache entirely in compute-enabled 3D NAND flash. KVNAND addresses the fundamental performance challenges of flash under intensive KV cache access by leveraging IFC for all memory-bound operations to reduce data transfer overhead, introducing head-group parallelism to boost throughput, and employing page-level KV cache mapping to align token access patterns with flash organization. In addition, we propose a design space exploration framework that evaluates discrete and compact KVNAND variants to balance weight and KV placement, automatically identifying the optimal design trade-off. These techniques mitigate latency, energy, and reliability concerns, turning flash into a practical medium for long-context KV storage. Evaluations on MHA 7B and GQA 70B LLMs show that KVNAND achieves $1.98\times/1.94\times/2.05\times$ geometric speedup at 128/1K/10K-token contexts compared to DRAM-equipped IFC designs and addresses out-of-memory failures at 100K context length.

I. INTRODUCTION

As Large Language Models (LLMs) integrate into daily workflows, demand increases for personalized AI agents that align with user preferences, domain knowledge, and interaction styles. Deploying such agents on edge devices offers privacy, low-latency responsiveness, and cost efficiency by eliminating cloud dependency, making on-device LLMs a compelling direction for AI democratization [81].

Realizing high-quality personal LLM agents on resource-limited edge devices faces two main bottlenecks: memory capacity and bandwidth. [67], [78], [81], [89] Modern LLMs require tens to hundreds of GBs just for weights (e.g., LLaMA2-70B needs ~ 140 GB [71]), far exceeding typical

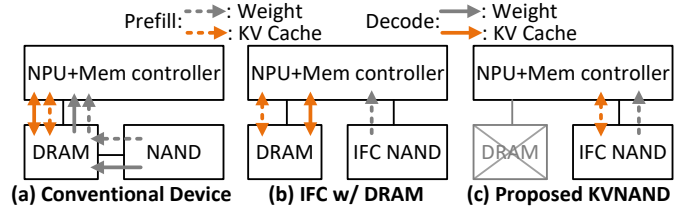


Fig. 1: Different on-device LLM inference architectures.

edge-device DRAM limits (8–16 GB on smartphones, <64 GB on laptops). The growing demand for long-context agentic workflows like long document analysis [35], multi-turn dialogue [84], and chain-of-thought reasoning [10] introduces the KV cache as another dominant consumer of this limited memory [19], [74]. Moreover, recent state-of-the-art (SoTA) models support extensive context lengths ranging from 128K (LLaMA3.1-70B [50]) to 1M (Gemini 2.5 Pro [13]). The KV cache demand scales linearly with context length; for example, a 13B model already requires ~ 8 GB KV memory at a 10K context [71], placing prohibitive pressure on edge resources. Meanwhile, unlike the compute-intensive prefill stage dominated by matrix–matrix multiplication (GEMM), decoding reduces to memory-bound matrix–vector multiplication (GEMV), exhibiting extremely low arithmetic intensity (≈ 1 OPS/Byte at FP16).

To address the challenges of memory footprint, conventional devices selectively load weights from high-capacity flash into DRAM [5], but still suffering from limited flash I/O bandwidth. Figure 1(b) summarizes more advanced in-flash computing (IFC) designs such as Cambricon-LLM [81] and Lincoln [67], which push memory-bound operations into the flash to utilize the higher inner bandwidth. Unfortunately, these designs still rely on DRAM for the KV cache, which imposes prohibitive cost and energy under long contexts, while KV-dependent attention computations remain on the NPU, constraining performance.

To overcome these bottlenecks, we propose KVNAND, a DRAM-free NPU-IFC architecture that unifies model weights and KV cache within compute-enabled 3D hybrid-bonding NAND flash. KVNAND eliminates the need for DRAM by

storing KV cache directly in flash, leveraging the large capacity and low cost. The architecture extends IFC coverage beyond model weights-related GEMVs to the KV-related GEMVs to improve performance. This approach ensures that all memory-bound computations are serviced by IFC, avoiding intensive KV cache transfers.

To accommodate diverse model scales and context length requirements, we design two complementary KVNAND variants and explore the design space, drawing inspiration from the fundamental tradeoffs inherent in pipeline-parallelism (PP) and tensor-parallelism (TP). KVNAND-Discrete places weights and KV cache in separate groups of IFC dies for KV-heavy cases, enabling head-group pipeline parallel execution that overlaps QKV generation with attention computation. KVNAND-Compact instead stores KV cache in-place within the same flash arrays as the weights for short context length, exploiting higher TP under the same hardware budget and reducing KV inter-die communication.

KVNAND further introduces a page-level KV cache mapping strategy that aligns KV placement with flash’s page-level access granularity and the distinct access patterns of KV generation and attention. With the aid of lightweight KV buffers, this scheme improves read/write efficiency and significantly reduces redundant page reads. In addition to IFC’s core benefit of significantly reducing the movement of intermediate results and large-batch KV cache, the external bandwidth of modern NAND is rapidly evolving. Per-die bandwidth has increased from 3.6 GB/s (ONFI 5.2) to 4.8 GB/s [51], [58], [79], closely approaching that of LPDDR5X DRAM (8 GB/s) [54]. Reliability concerns can also be alleviated by using periodic reclaim [67] and intensive spare flash capacity.

To the best of our knowledge, KVNAND is the first on-device LLM inference architecture that offloads all storage-intensive operands directly into compute-enabled flash dies. Our design preserves the key advantages of SoTA IFC-based LLM accelerators while further extending the role of modern 3D NAND flash as both the storage substrate and execution engine for deploying long-context LLMs on edge devices. The main contributions are as follows:

- We propose the first DRAM-free IFC-NPU heterogeneous on-device LLM inference accelerator, which stores both the KV cache and weights and executes memory-bound operations in compute-enabled 3D NAND flash.
- We introduce discrete and compact variants of KVNAND and provide a design space exploration (DSE) framework for selecting configurations under given model, context length, and hardware constraints.
- We optimize attention layer dataflow and KV cache mapping by aligning LLM inference characteristics with KVNAND flash storage properties to reduce execution time and energy while alleviating reliability stress.
- We evaluate the performance and power of KVNAND. Across MHA 7B and GQA 70B LLMs, KVNAND achieves $1.98 \times 1.94 \times 2.05 \times$ geomean speedup compared with DRAM-equipped IFC designs under 128/1K/10K-

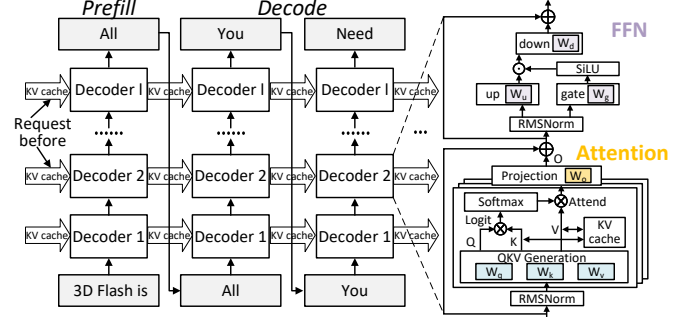


Fig. 2: General LLM architecture and inference flow.

token contexts, and $1.17 \times 1.32 \times$ geomean energy efficiency improvements under 10K/30K-token contexts. In addition, KVNAND reduces memory cost by 69% and solves the out-of-memory at 100K context length.

II. BACKGROUND

A. LLM Inference

Mainstream decoder-only LLMs [28], [70], [85] consist of stacked decoder blocks, each with an attention module and a feed-forward network (FFN) as shown in Figure 2. The attention layer generates Q/K/V, performs self-attention (Logit, Softmax, Attend), while fully connected (FC) layers dominate the FFN. Multi-head attention (MHA) [72] enables parallel attention across different aspects of the context. Group-query attention (GQA) [3] is an efficient attention architecture that strikes a balance between the high quality of MHA and the low inference cost of Multi-Query Attention (MQA) [64] by grouping multiple query heads to share a single K and V head.

For a single inference request, the input sequence is first processed in prefill phase to generate an initial output token, followed by decode phase, where tokens are generated one at a time. A widely adopted technique is KV cache [38], lowering attention complexity from $O(s^2)$ to $O(s)$. However, as context length grows, KV cache becomes both capacity-intensive and bandwidth-demanding, especially since conversational history and reasoning traces further extend effective context [14], [41], [59]. To relieve DRAM pressure, recent GPU-based inference architecture offloads KV caches to flash [61], [62], [65], enabling larger contexts on memory-constrained devices. Yet, even with SSD-side compute [61], these designs fail to exploit flash’s internal bandwidth and remain impractical for resource-constrained consumer devices.

As shown in the roofline model of Figure 4(a), decode-phase GEMV computations are memory-bound, with performance constrained by the limited bandwidth of mobile NPUs. In single-batch on-device LLM inference, the absence of batch-level parallelism eliminates weight reuse across tokens, further lowering arithmetic intensity [24], [61], [81]. Consequently, data movement accounts for over 90% of decode latency [67], leaving compute units heavily underutilized.

B. In-Flash Computing for LLM

To address the low arithmetic intensity of LLM decoding, recently, Processing-In-Memory (PIM) has emerged as

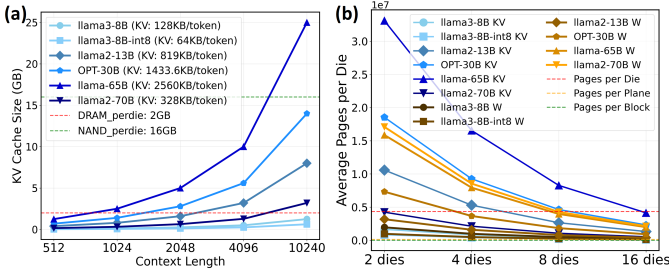


Fig. 3: (a) KV cache size of various models as the context length increases. (b) Number of NAND flash pages occupied by KV cache and model weights.

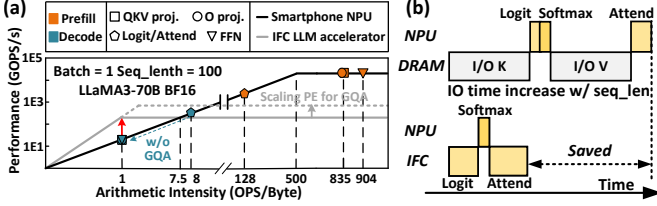


Fig. 4: (a) Roofline model analysis of IFC and smartphone NPU. (b) Comparison of attention computation workflows when KV cache is stored in DRAM and in NAND with IFC.

a promising direction [21], [24], [26], [42]. In particular, on-die in-flash computing (IFC) leverages the low cost and high storage density of 3D NAND flash to perform near-data processing [11], [44]. Recent works demonstrate IFC effectiveness for on-device LLMs, achieving up to 36.3 tokens/s on LLaMA2-7B [81] and 11.5 tokens/s on LLaMA-65B [67], sufficient for real-time inference. Moreover, the high internal bandwidth and large capacity offered by 3D stacking technologies further accentuate the advantages of IFC for on-device LLM accelerators.

Cambricon-LLM [81] adopts a chiplet-based hybrid architecture with compute-enabled flash dies connected to the NPU via high-speed die-to-die links. Lincoln [67] integrates compute logic beneath each flash plane and interfaces flash to the SoC through LPDDR PHYs. AiF [40] further optimizes flash circuits and reliability. These advances define a new architectural paradigm for on-device LLM inference, referred to as IFC-NPU. IFC subsystems accelerate memory-bound operations involving model weights in the decode phase. The NPU handles the prefill phase and nonlinear computations. In these architectures, flash is used solely for storing weights, with KV caches maintained in DRAM.

III. MOTIVATION

A. The Overlooked Crisis: Long Context KV Cache Storage

With the rapid growth of context lengths in LLM inference, the KV cache scales with sequence length and can surpass model weights as the dominant consumer of memory [29], [45]. The problem is especially severe for large-scale models preserving long conversational histories. Figure 3(a) illustrates that KV cache size grows rapidly with context length and can even exceed model weights. For instance, with a 10K

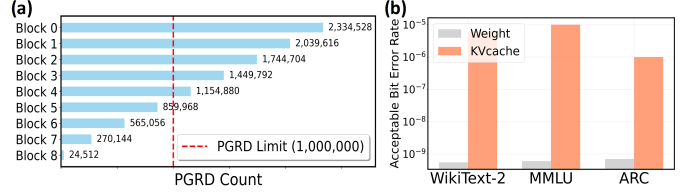


Fig. 5: (a) Page-read (PGRD) counts for LLaMA3.1-8B KV cache with a 50K-token context (1K input and 1K generated tokens per request). (b) Acceptable BER in model weights and KV cache resulting in a 10% accuracy/perplexity degradation.

context, LLaMA2-70B requires several gigabytes of KV storage despite using GQA. Figure 3(b) further shows the flash page occupancy of 16GB flash dies [67], where large models such as LLaMA2-70B demand multiple dies exclusively for KV cache at 100K tokens, while smaller GQA-based models like LLaMA3-8B [50] also needs one to two 16GB IFC dies. Compared to DRAM, 3D NAND flash provides the same capacity at about one-fifth the cost [81] and is more resilient to out-of-memory (OOM) failures. These trends highlight flash as a practical and scalable medium for long-context KV storage in on-device LLM inference.

B. The Latency Barrier: Why Naive NAND Substitution Fails

A naive substitution of NAND for DRAM to offload the KV cache, while addressing the aforementioned capacity and cost issues, introduces significant performance challenges due to the higher latency of NAND flash. First, flash’s external bandwidth trails that of DRAM. For per-die external bandwidth, a single LPDDR5X channel can sustain about 8 GB/s [54], whereas modern NAND flash (even ONFI 6.0) reaches 4.8 GB/s [51]. More critically, the internal access latency gap is fundamental. DRAM provides access at the nanosecond (ns) scale. Conversely, NAND’s internal program operations are orders of magnitude slower, operating at the microsecond (μ s) scale (e.g., a conservative 75 μ s program time [36], 1.6 GB/s internally). This stark latency divide renders simple offloading infeasible. It necessitates specialized KV cache storage and mapping schemes that align with flash access characteristics, thereby hiding fragmented access patterns and mitigating their latency impact. This aspect will be elaborated in Section IV.

Here, we first conduct an evaluation of a naive design that simply replaces DRAM with flash in Lincoln’s architecture [67], with an internal read bandwidth of 32 GB/s, assuming ideal KV cache access and 4 dies in parallel. This bandwidth level analysis provides an initial understanding of the feasibility and limitations of storing KV cache in flash. We use the popular mixtral-of-experts (MoE) model Mixtral-8 $\times$ 7B [28] as a case study. In the single-batch decode phase, generating one token with GQA yields a KV cache size of: $KV_per_tk = 2 \times l \times k \times \frac{d}{h} \times 2B = 128KB$ in BF16, where the factor of 2 accounts for both K and V tensors, l is the number of decoder block layers, d is the hidden size.

(1) Reading KV cache from flash. For larger KV cache reads, the flash page buffer allows pipelining, which hides

part of the access latency. During attention computation with a sequence length of 1K tokens, the read latency is: $t_{KV,read} = (KV_per_tk \times seq_length) / (4 \times BW_{external}^{read}) \approx 6.9 ms$.

(2) FFN acceleration in IFC. Each routed expert has about 175M parameters, or 87.5 MB under INT4 quantization. With 8 experts per layer and 32 layers, the FFN weights total roughly 22.4 GB, but only 2 experts are activated per inference step. The corresponding read time is: $t_{FFN,read} = (l \times expert_size \times num_expert) / (4 \times BW_{internal}^{read}) \approx 44 ms$.

For short contexts, FFN execution dominates end-to-end latency, with attention contributing only a minor fraction. As the context length grows, attention latency increases proportionally and the bandwidth gap between flash and DRAM becomes more pronounced. Similar trends hold across different LLM scales. Consequently, an intuitive solution to this KV movement latency is to increase the external NAND bandwidth. For instance, Lincoln [67] connects IFC dies via LPDDR-style interfaces, largely mitigating the external I/O bottleneck, while other proposals leverage high-speed custom die-to-die links [81]. However, these specialized solutions incur significant custom design costs and forego the benefits of leveraging the mature and commercial ONFI interface.

C. Unleashing Full Potential: IFC for Mem-Bound Operands

As our analysis confirms, transferring large volumes of KV cache from DRAM to the NPU incurs significant latency as sequence length increases. Rather than merely optimizing this costly I/O path, a more fundamental solution is to avoid the data movement altogether. This motivates our proposal to leverage an IFC architecture, as shown in Figure 4(b). All three SoTA IFC-based LLM inference accelerators [40], [67], [81] leverage the insight from the roofline analysis in Figure 4(a): the memory-bound operands of LLM inference should be offloaded to IFC if possible. However, in these designs, KV cache remains stored in DRAM, which forces the memory-bound Logit and Attend operands in attention layer to be executed on NPU. In contrast, storing KV cache directly in NAND allows these two operations to be executed entirely within IFC die, saving associated I/O time and energy. This enables all memory-bound operators in LLM inference to be offloaded to IFC units.

Noted that models with a GQA architecture exhibit an $h/k \times$ (typically 4 or 8) arithmetic intensity, pushing it beyond the roofline intersection and making it less memory-bound, because the number of KV heads k is smaller than the number of Q heads h . Therefore, for an IFC system to maintain a performance benefit on GQA workloads, its peak computational throughput must be correspondingly increased as illustrated by the gray dashed-line in Figure 4(a). As shown in Section V, the resulting workload remains well-suited for IFC execution in most cases.

D. Reliability Consideration

The reliability problems like retention, read disturbance, and endurance of NAND flash [7], [88], have been explicitly considered in several flash-based LLM accelerators [40], [61],

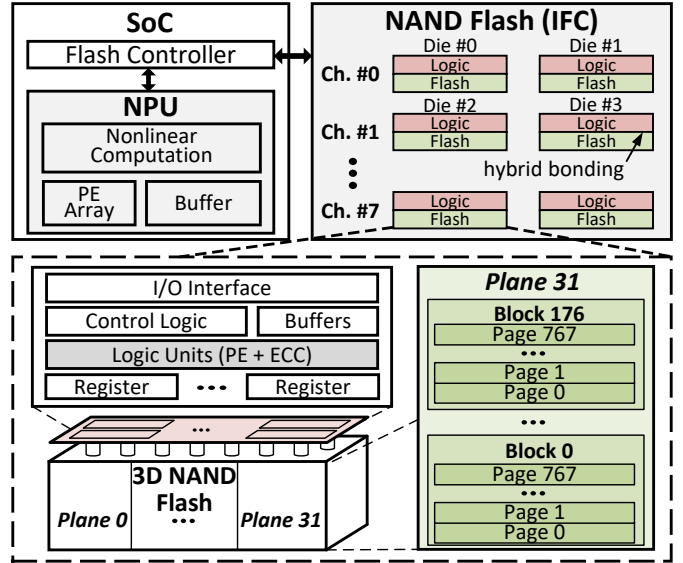


Fig. 6: KVNAND overall architecture.

[67], [81]. Lincoln identified read disturbance [6], [33], [86] as the primary reliability concern that increases the bit error rate (BER) since the continuous long-output generation. For weight storage, each page is written once and then read many times for decoding, leaving sufficient programs/erases (P/E) endurance margin to perform read reclaim operations [34].

For KV cache stored in NAND flash, two primary reliability stress factors must be considered: (1) increased P/E cycles due to KV updates, and (2) accumulated page reads. The access pattern of the KV cache differs from that of model weights, as KV entries generated during decoding are only consumed in subsequent token-generation iterations. As shown in Figure 5(a), under the flash configuration of Lincoln [67] with a 50K-token context window (1K input and 1K generated tokens per request), blocks storing early KV entries exhibit higher cumulative page-read counts, while blocks holding later KV caches remain well below the read-disturb endurance limit. Fortunately, flash capacity enables trading space for reliability by retiring high-BER blocks. Moreover, fragmented KV accesses can amplify page reads, a challenge we mitigate through the mapping schemes in Section IV.

Bit-error injection further validates the feasibility. As illustrated in Figure 5(b), experiments on LLaMA3.1-8B [50] across WikiText-2 [52], MMLU [25], ARC [12] demonstrate that accuracy degrades by over 10% at much lower BER thresholds for weights than for KV cache, indicating that KV data are inherently more error-tolerant due to the error-masking effect of the Softmax operation [77].

IV. KVNAND DESIGN

A. Overall Architecture and Dataflow

Hardware design. Figure 6 illustrates the core concept of the proposed KVNAND architecture: both model weights and KV cache are stored and computed inside compute-enabled flash, enabling fully DRAM-free inference. By leveraging

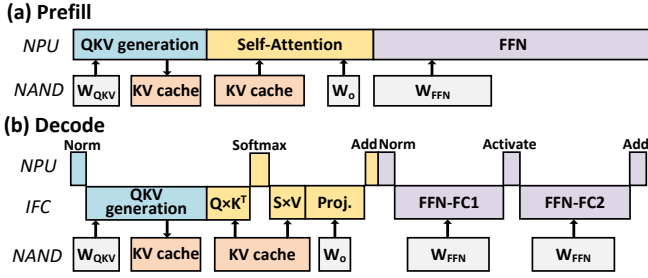


Fig. 7: Overall dataflow of (a) Prefill and (b) Decode phase.

IFC beyond weight-only GEMV to KV-related operands, KVNAND eliminates intensive KV transfers in a long context window. The system consists of 3D NAND flash dies with IFC, a neural processing unit (NPU), and a multi-channel flash controller. The NPU and flash controller are integrated into the system-on-chip (SoC).

In KVNAND architecture, 3D NAND flash arrays and their peripheral circuits are connected through wafer-to-wafer hybrid bonding [55], [60], [67], [80]. On the upper CMOS die originally reserved for flash peripheral circuitry, additional logic units with processing elements (PEs) are integrated to form the logic die, enabling GEMV operations between stored weight matrices or KV cache and input vectors. Data read from the flash array first enter the data and cache registers, passes through lightweight on-die ECC, and is then fed into the PE. Computation results and input vectors are exchanged between IFC and NPU as needed.

The NPU is implemented on a separate die and integrates general-purpose PE arrays for high-throughput GEMM, residual connection (add), as well as specialized functional units for operations such as normalization, rotary position embedding (RoPE), activation, and Softmax. These vector operations are ill-suited for IFC, as they offer no bandwidth savings due to identical input/output sizes and incur prohibitive design complexity within the resource-constrained logic die. Moreover, the distributed outputs from multiple parallel PEs in IFC must be aggregated before certain computations.

The flash controller on the SoC manages communication with multiple IFC dies, each connected through an independent channel. Multiple channels enable scaling of aggregate I/O bandwidth and computation parallelism across IFC dies.

Compute Flow. Figure 7(a) depicts the prefill phase, which is dominated by compute-bound GEMM. Prefill is therefore executed on the NPU, following prior schedulers [40], [67], [81]. The NPU streams weights from flash, performs QKV projection, self-attention, and the FFN. The resulting KV tensors are stored in flash.

Figure 7(b) illustrates the decode phase, where KVNAND fully exploits the advantage of storing the KV cache in flash. All memory-bound GEMVs of LLM single-batch inference are offloaded to the IFC logic die. This includes QKV generation, Logit, Attend, O projection, and the FC layers in the FFN. In this phase, the NPU is only responsible for nonlinear functions. Both phases require bidirectional data exchange between the NPU and flash, with intermediate data and KV cache updates

passing through the interface. Unlike prior IFC accelerators that use DRAM to buffer embeddings and intermediate results, KVNAND keeps these smaller data structures in on-chip SRAM. Some works [40], [81] offload part of the decode-phase GEMV computation to NPU to balance utilization, but such techniques require additional system-level support. KVNAND establishes a baseline architectural optimization that can be seamlessly combined with these methods.

B. KVNAND-Discrete and KVNAND-Compact

When both weights and KV cache are stored in flash and attention GEMV operations are executed on IFC, the design space becomes more complex than in DRAM-equipped designs [40], [67], [81]. We propose two design variants, KVNAND-Discrete (KVNAND-D) and KVNAND-Compact (KVNAND-C), which are shown in Figure 8. These two variants are explicitly designed to accommodate diverse model scales or context lengths.

As shown in Figure 8(a), KVNAND-D partitions all IFC dies into two groups: Group 1 (G1) stores model weights and executes QKV generation, O projection and FC, reusing dataflow of prior IFC [67], while Group 2 (G2) stores KV cache. Both groups share the same die configuration for cost-effective manufacturing, while the die allocation between G1 and G2 can be flexibly tuned. For short contexts, FFN dominates latency, favoring more dies in G1. As context length grows, attention latency and KV cache size increase, making additional G2 dies more beneficial. The configuration trade-offs will be further discussed in Section V.

The execution dataflow involving KV cache in KVNAND-D is as follows. W_{QKV} is partitioned in G1 to increase per-head generation parallelism as in Lincoln [67]. Each page of multi-plane is read simultaneously (①) and multiplied with an input vector broadcast to all planes across all dies in the PE (②), with partial results accumulated in the PE registers. When a complete output is produced, it is collected in the global buffer and sent to the SoC as Q/K/V results (③). Newly generated KV cache entries are first stored in SRAM-based KV buffer in SoC, and once the accumulated size reaches a proper size, they are written to G2 flash (④⑤). For attention, G2 IFC dies load the required K vectors from flash (⑥) and multiply them with broadcasted Q vectors (⑦). The results are aggregated in the NPU (⑧) to compute the Softmax, which are then sent back to IFC dies to perform the Attend with V vectors (⑧⑦).

As shown in Figure 8(b), KVNAND-C integrates KV cache and weight storage within the same die, leveraging spare capacity after weight placement to accommodate KV data. In this design, Q/K/V generation also begins by reading the W_{QKV} pages (①) and generating Q/K/V in the PEs (②). Generated K and V are temporarily stored in the KV buffer of each plane, and once sufficient KV data is accumulated, it is written back (⑤) to KV-dedicated blocks in the same plane. Self-attention in KVNAND-C begins by loading KV cache from flash (⑥) and, as in KVNAND-D, performing the Logit and Attend in the PEs (⑦), with intermediate results exchanged with the NPU (⑧).

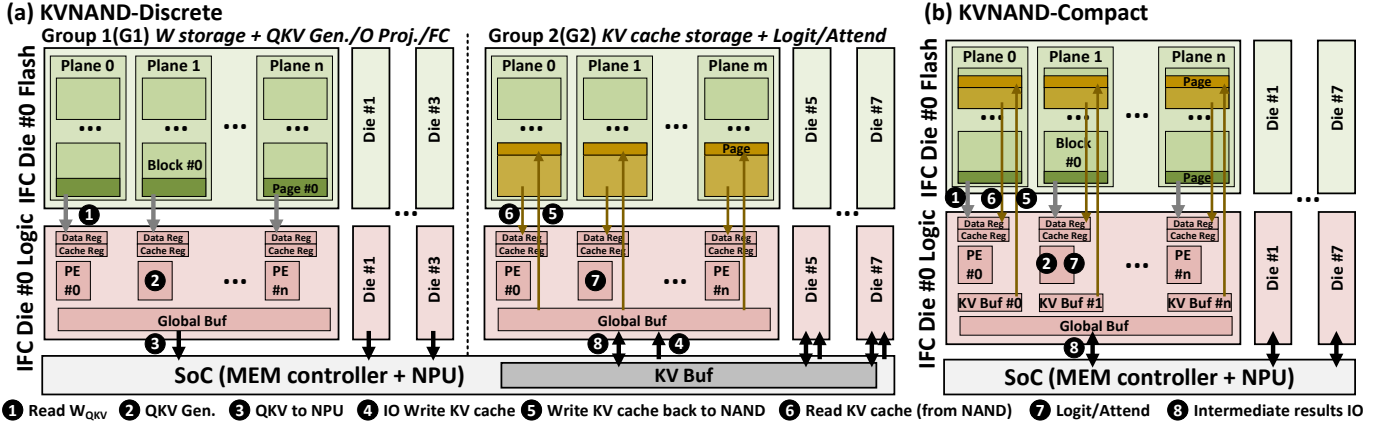


Fig. 8: (a) KV-Discrete and (b) KV-Compact architectures with the QKV-Gen. and Attention executing flow in decode phase.

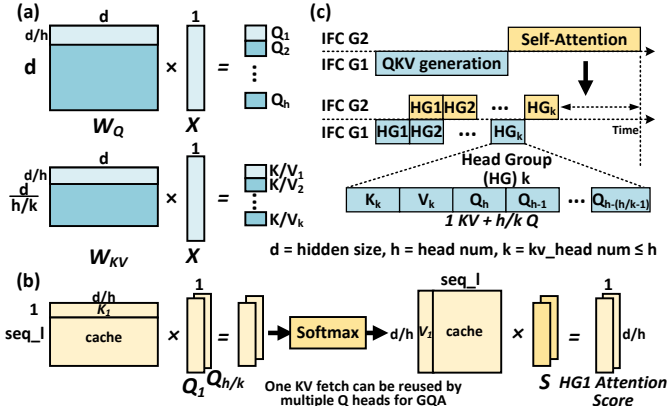


Fig. 9: (a) QKV generation and (b) Self-attention of GQA model, and (c) KVNAND-D head-group parallelism dataflow.

KVNAND-C increases tensor-parallelism by leveraging all hardware resources, whereas KVNAND-D employs pipeline-parallelism to overlap QKV-generation and Logit/Attend but sacrifices some tensor-parallelism. Compared to the discrete design, KVNAND-C also avoids cross-die data movement for KV storage. For an equivalent channel and die configuration, KVNAND-C can offer higher compute parallelism, accelerating QKV generation, attention computation, and the subsequent FFN stage, which often accounts for a large portion of total inference latency when context length is short. Another distinction of the compact design is that ECC encoding must be performed on the logic die to protect the KV data before writing to flash, rather than in the SoC flash controller as in the discrete design. This adds some area and power overhead on the logic die, though the cost is modest since ECC encoding typically requires only about one-quarter of the area of decoding [4], [20], [53].

C. Dataflow Optimization for Attention

Prior IFC-based accelerators optimized only weight-related GEMVs. With KVNAND storing KV cache in flash, the key is to accelerate attention with KV accesses, as fully utilizing IFC throughput is critical for long-context latency.

Single-batch on-device inference requires strictly sequential decoding for each token, where QKV generation, self-attention, and FFN execute in order. This dependency chain limits intra-layer parallelism. However, QKV computations across heads are independent and can run in parallel, as illustrated in Figure 9. This enables temporal overlap between QKV generation and self-attention, with maximum benefit when attention time is comparable to QKV generation time at a given sequence length. This software-level attention-head partitioning of the LLM decode process naturally aligns with the inherent hardware partitioning of our KVNAND-D architecture. As shown in Figure 9(c), this synergy allows us to leverage the G1 and G2 IFC groups to execute QKV generation and the subsequent attention computation in a decoupled, pipelined fashion. This approach effectively hides the execution latency and minimizes overhead.

To realize this execution, we introduce the head group (HG) as the fundamental unit of granularity for pipeline-parallelism scheme, as shown in Figure 9(c). Each HG consists of one KV pair and its associated Q heads: for example, in LLaMA3-8B ($h = 32, k = 8$), each HG contains one KV pair and four Q heads, while in LLaMA2-70B ($h = 64, k = 8$), each HG contains one KV pair and eight Q heads. In standard MHA where $h = k$, each HG naturally reduces to a single QKV triplet. Other parallelism forms, such as concurrent execution of gate and up-projection in FFN, are orthogonal to our approach and can complement in KVNAND.

Figure 10(a) shows the KVNAND-D timing breakdown with HG-parallelism. HG1 generation leverages the two-stage register pipeline, common in many flash devices, to overlap QKV weight reads (1) with GEMV computation (2). Following [67], [81], the IFC logic makes 1 and 2 take roughly the same time for bandwidth matching. For instance, if each plane reads a 4 KB page in $4 \mu s$, the compute units are sized to finish the corresponding MACs in $4 \mu s$ (i.e., 2FMACs at 400MHz). Increasing the die count raises parallelism, reducing the number of 12 iterations and thus the total QKV latency. Once computation is completed, results are sent via the IFC die I/O to the NPU memory controller (3) and then written into G2 dies (4) for self-attention. Because part of the work is already

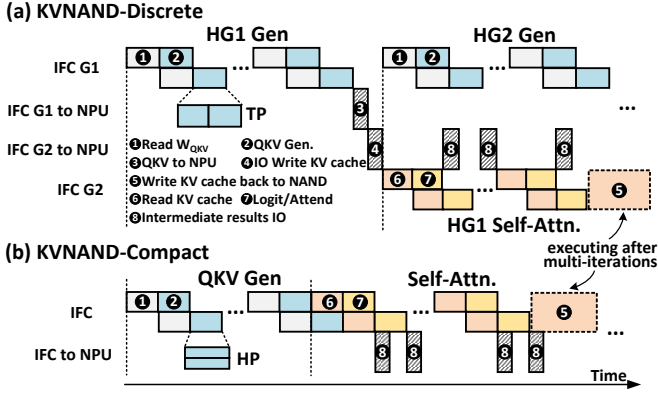


Fig. 10: Attention layer sequence flow of (a) KVNAND-D and (b) KVNAND-C.

done in IFC, the transfer volume is small and the external bandwidth is negligible. At this point, G1 can immediately start generating HG2, independent of other results. In parallel, G2 performs self-attention GEMV for HG1: KV cache is read (⑥), Logit and Attend computations are executed (⑦) using the same ①② pipeline to overlap part of the read time. For Logit, K is loaded and aggregated on the NPU (⑧), followed by Softmax and broadcasting back to G2 for Attend. The duration of the attention scales with context length, since both KV reads and computation grow with sequence length, as shown in Figure 9(b). A write to the G2 flash (⑤) occurs only after generating multiple tokens, when the KV buffer becomes full, thereby amortizing the write-overhead.

Note that in GQA models, each KV fetch is reused across multiple Q heads, enabling a single access to serve multiple attention computations. Thus, achieving a bandwidth–compute balance requires scaling the PE count by a factor of h/k to equalize the latencies of phases ① and ②. This implies that for a typical GQA model where $h/k = 8$, the required bandwidth–compute balance necessitates 16 FMACs per plane operating at 400MHz. Although this compute capability appears superfluous for weight-related GEMVs executing in G2 or KVNAND-C, it can be exploited for speculative decoding to obtain further performance gains [67]. Moreover, our evaluation in Section V demonstrates that this added compute incurs only a negligible area overhead on the logic die.

The KVNAND-C timing diagram is shown in Figure 10(b). Unlike KVNAND-D, it cannot overlap HG generation and attention, because both W_{QKV} reads (①) and KV cache reads (⑥) contend for the same flash internal read bandwidth. However, this architecture compensates by leveraging the full computational power of all IFC dies in both non-overlapped stages, thereby reducing the iteration count for phases ①② and ⑥⑦. Regarding the detailed dataflow for QKV generation, KVNAND-C requires that QKV head generation be parallelized across HGs (Head-Parallelism, HP). That is, instead of applying all multi-plane parallelism to a single head (Tensor-Parallelism, TP), multiple heads are generated concurrently. This approach resembles superscalar versus scalar execution [37]. While the overall latency remains unchanged given fixed

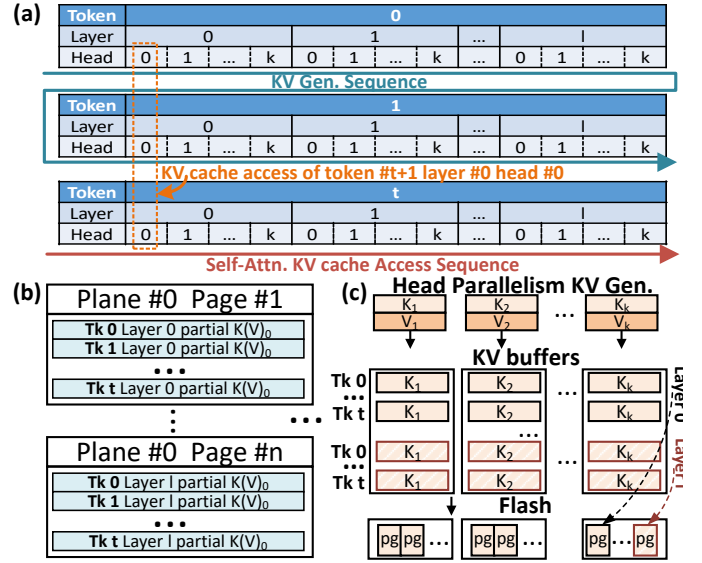


Fig. 11: KV cache storage mapping scheme.

compute resources, this HP execution model is a key prerequisite for the KV cache mapping scheme adopted by KVNAND-C in the next section. In contrast, G1 dies of KVNAND-D can employ TP-based data flow for QKV generation.

D. KV Cache Mapping Scheme

Unlike model weights, which are read-only during inference and exhibit static access patterns, the KV cache is dynamically generated during decoding and exhibits two unique storage characteristics as described in Figure 2 and Figure 11(a). (1) Temporal ordering of token generation, in which KV entries are produced sequentially as new tokens are generated. (2) Independence of KV usage across layers and heads causes conflicts between fine-grained KV cache operations and coarse-grained flash storage.

Page-level KV Cache Mapping. As shown in Figure 11(a), without optimization, KV entries are written in generation order, so each page mixes data from different layers and heads. Since attention requires KV from multiple tokens but only within the same layer and head, this misalignment reduces spatial locality and increases page reads. To avoid this, KV entries belonging to the same layer and head group should be stored contiguously in flash pages, as shown in Figure 11(b). This organization improves page-level locality, increasing the effective hit rate and thus enhancing performance and reliability. A straightforward solution is to buffer KV entries from multiple tokens until one page is filled with data for a single head in a single layer, then write it to flash. However, this requires buffer capacity on the order of $k \times l$ pages (roughly $100 \sim 1000 \times$ page size), which is infeasible for the limited on-die area in KVNAND-C and would also enlarge the SoC-side buffer in KVNAND-D, potentially reintroducing the need for costly off-chip DRAM.

To address this challenge, we propose a page-level KV cache mapping scheme and its hardware workflow. Taking KVNAND-C as an example, by parallelizing KV head gen-

eration across different planes, as illustrated in Figure 11(c), each plane’s KV buffer only needs to hold the KV data of its assigned head until enough tokens are accumulated to fill one page before writing to flash. This reduces the buffer requirement per plane by a factor of k , significantly easing the on-die area constraint. In addition, partial-page updates [30]–[32] can be applied to write data into a page at a finer granularity, further reducing the required buffer size.

Specifically, under HP generation, the number of tokens (t) required to fill a page buffer can be expressed as: $t = \text{page_size} \times (\text{num_die} \times \text{num_plane} / 2k) / \text{KV_size_unit}$, where $(\text{num_die} \times \text{num_plane} / 2k)$ is the number of planes allocated for generating either K or V after head partitioning, and $\text{KV_size_unit} = \text{precision} \times \frac{d}{h}$. For FP16 precision, this value is 256B in most mainstream models [70], [85], and remains constant across layers and heads. It is important to note that, computing $Q \times K^T$ requires accessing the K data in KV cache, while computing $S \times V$ requires only V data. Thus, in the proposed mapping scheme, K(V) refers to either K or V, with identical data layouts but stored in separate blocks.

The proposed KV cache mapping scheme is compatible with both KVNAND-C and KVNAND-D. In KVNAND-C, adopting the superscalar-like HP dataflow is a prerequisite because of the local KV buffer, whereas in KVNAND-D, it can be applied by the large KV buffer on the SoC.

Access-Aware Block Allocation. KV cache generated earlier in decoding is read at every subsequent step, leading to read disturbance comparable to weights and proportional to output length. To mitigate this stress, at the plane level, blocks are randomized across inference requests to balance wear, with counters tracking access and P/E cycles. When limits are reached, data is migrated or refreshed with the flash translation layer (FTL) table updated accordingly. The sequential page order within blocks is preserved for high read speed [40].

V. EVALUATION

A. Methodology

Baseline and Configuration. We evaluate the following systems: (1) Base-1 (Weight-only IFC + DRAM): IFC design storing all model weights in NAND flash, while KV cache resides entirely in DRAM. This architecture is consistent with existing IFC-based on-device LLM accelerators. We scale the Lincoln architecture [67], configured with one IFC die and one DRAM attached to each of the 8 LPDDR channels to ensure that the DRAM has sufficient capacity to store long text sequences. (2) Base-2 (Naive KV-in-Flash): Derived from Base-1 by simply replacing DRAM with common NAND flash for KV cache storage. (3) KVNAND-D-(G1+G2): Proposed discrete variant storing model weights in one group of IFC dies (G1) and KV cache in another (G2). The suffix (G1+G2) specifies the channel and die allocation. For example, KVNAND-D-(6+2): 8 channels $\times$ 1 die per channel, with 3 channels assigned to G1 and 1 channel to G2. (4) KVNAND-C- x : Proposed compact variant that co-locates weights and KV cache within x dies. For a fair comparison, we use the KVNAND-C-16 configuration.

TABLE I
CONFIGURATION OF KVNAND AND BASELINE

NPU Configuration	
Compute	32 TOPS (BF16 Tensor Core), 4.60 W
SRAM	5MB KV buffers for KVNAND-D, 0.36 W
Memory System Configuration	
DRAM: LPDDR5X, 16Gb, 8GB/s, 7 pJ/bit (Base-1)	
NAND Flash: ONFI 6.0, 128Gb, 4.8GB/s, 4.9 pJ/bit	
KVNAND Flash Configuration	
Flash layer	SLC, 96 wordline-layers, (4KB + 448B ECC) / page 768 pages / block, 177 blocks / plane, 32 planes 4.1484 Gb per plane, 132.75 Gb per die
	tR = 4 μ s, tP = 75 μ s, 3 pJ/bit (read), 7.5 pJ/bit (program)
Logic layer	16 FMACs + 8 KB KV buffer per plane for KVNAND-C 0.0737 mm ² , 6.98 mW per plane
	ECC: BCH(9088, 8192, 64), 14.6 Gb/s per plane dec/enc: 0.1478/0.0598 mm ² , 5.24/1.2 mW per plane
	260 KB Global buffer 0.4095 mm ² , 18.4 mW in total

KVNAND-D and KVNAND-C extend Base-2 by replacing the KV cache flash with compute-enabled IFC dies. This maintains the same number of memory dies in each channel, an advantage of eliminating DRAM, since each channel can instead host additional IFC dies. As shown later in the scalability analysis, even with the same number of IFC dies as Base-1, completely removing DRAM and adding flash capacity can still deliver comparable performance.

Table I lists hardware parameters used in our evaluation. Most parameters are derived from Lincoln [67] for consistency and fair comparison. Performance modeling follows Cambricon-LLM [81] with SSDsim [27] to capture both memory and compute behaviors of IFC subsystem, while NVSim-based [16] 3DFPIM simulators [39] validate flash latency/energy using conservative estimates. The digital logic of the IFC layer, with reduced FMAC units of Lincoln [67], is synthesized using Design Compiler and also scaled to 20 nm, while on-chip SRAM buffers are generated using the ARM memory compiler. Considering area limits, KVNAND-C provisions 8 KB KV buffers per plane, while KVNAND-D employs a 5 MB SoC-side KV buffer. DRAM access energy in Base-1 is modeled at 7.0 pJ/bit [1], [16], flash interface access at 4.9 pJ/bit [57], and internal flash reads at 3 pJ/bit [67]. Write operations are modeled with $tP = 75 \mu$ s [36] and 7.5 pJ/bit energy ($\approx 2.5 \times$ read energy [36]). We evaluate MoE FFN using TP, which shards the expert weights across the IFC dies and executes the low-overhead router on the NPU.

Benchmarks. We evaluate five full-precision LLMs (MHA-based OPT-30B [85], LLaMA2-7B [71] and GQA-based LLaMA3.1-8B [50], LLaMA3.1-70B [50], Mixtral8 $\times$ 7B [28]) spanning different scales and architectures and various mixed-precision quantization schemes for comprehensive DSE.

B. End-to-end Performance and Energy Efficiency

Performance Speedup. Figure 12 compares the decoding throughput of KVNAND-D and KVNAND-C against two baselines, while Figure 13 provides a breakdown of the major components of decode latency for the 8B model under 1K

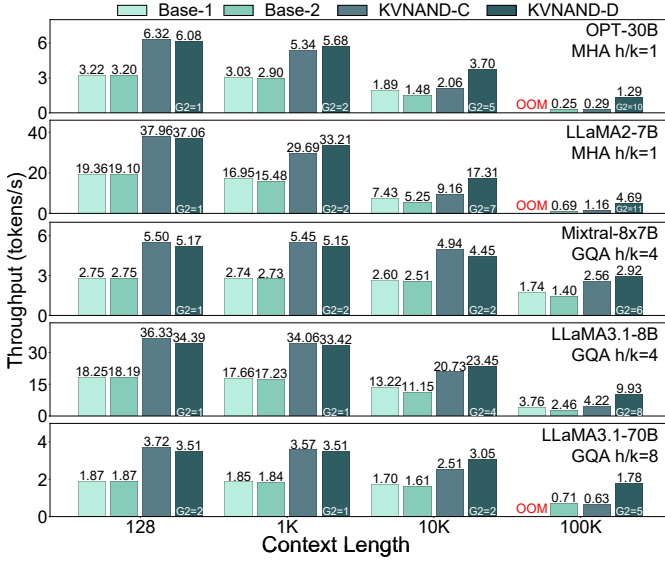


Fig. 12: Throughput of Base-1, Base-2, KVNAND-C, and KVNAND-D under 1K, 10K, 100K context length.

and 10K context lengths. To ensure optimal benefits, the hardware configuration for KVNAND-D is selected based on the results of our DSE, which is presented subsequently. Three key observations can be made:

First, KVNAND-D consistently improves throughput across most models and context lengths. At a 100K-token context, it achieves $5.2\times$, $6.8\times$, $4.0\times$, $2.5\times$, $2.1\times$ higher decode speed than Base-2 for the OPT-30B, LLaMA2-7B, LLaMA3.1-8B, LLaMA3.1-70B, and Mixtral-8x7B, respectively. The performance gains become more pronounced as context length increases, driven by the larger ratio of KV cache-related attention computations as depicted in Figure 13. Although GQA’s higher arithmetic intensity limits the IFC speedup, our design remains beneficial thanks to the additional computational power that is incorporated based on our roofline intersection analysis.

Second, KVNAND-C demonstrates higher gains at shorter context lengths. At 128 context length, it achieves $1.98\times$ and $1.05\times$ geomean speedup over Base-1 and KVNAND-D across all models. This advantage stems from its higher degree of parallelism in accelerating FFN computations as depicted in Figure 13. At shorter context length, the workload is FFN-bound, and the optimal DSE split skews heavily towards G1 (e.g., G1=15, G2=1 for LLaMA3.1-8B, 1K context-length). This large G1 allocation makes its FFN computational power comparable to that of KVNAND-C, resulting in similar FFN performance. However, for longer contexts, more frequent KV cache transfers amplify the limitation of its smaller KV buffer, diminishing performance benefits relative to KVNAND-D.

Finally, we observe that Base-1 experiences an out-of-memory (OOM) failure at the 100K context length, as its DRAM capacity is insufficient to accommodate the requisite KV cache. Even the three GQA models, which are more KV-efficient, already exhaust the DRAM capacity at context lengths of 50K. Base-2 (naive KV-in-Flash) suffers severe

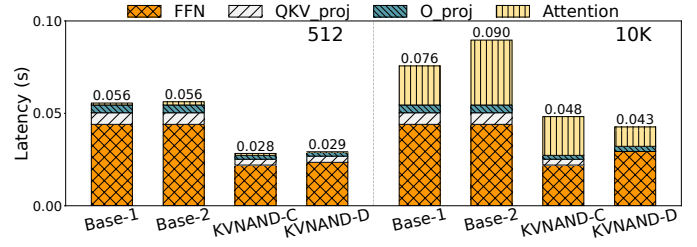


Fig. 13: Latency breakdown comparison (LLaMA3.1-8B).

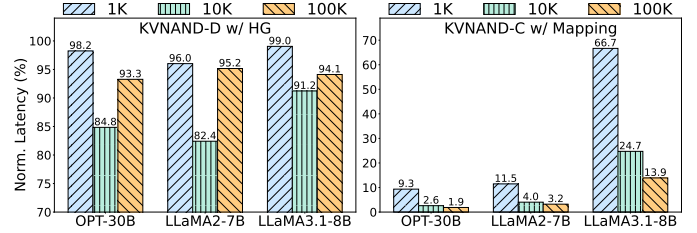


Fig. 14: Ablation study of HG parallelism in KVNAND-D and KV cache mapping scheme in KVNAND-C.

slowdowns due to flash access, resulting in significant additional latency from KV cache transfers during attention computation. This problem becomes more pronounced in longer contexts and highlights the necessity of KVNAND.

Ablation Study. Figure 14 evaluates the impact of two key optimizations. For KVNAND-D, HG parallelism reduces latency across different context lengths, normalized to a baseline without the dataflow optimization. At 10K tokens, latency drops to as low as 82.4%. The benefit is maximized when attention latency becomes comparable to QKV generation, enabling effective overlap between the two. However, at short contexts (e.g., 1K), QKV generation dominates latency, while at very long contexts (e.g., 100K), attention computation dominates, making the benefit less pronounced.

For KVNAND-C, the proposed page-level KV cache mapping optimization substantially reduces the portion of attention latency spent on flash reads. The improvement scales with context length, since larger KV caches exacerbate read overhead. At 100K tokens, the latency drops to only 1.9% for MHA 30B model compared to a design without mapping. The latency gains for GQA models are comparatively diminished. Notably, this optimization yields even greater benefits when applied to KVNAND-D, which provisions a larger KV buffer. In this analysis, we assume that the KV cache corresponding to the evaluated context length is fully stored in flash.

Scalability Study and Design Space Exploration (DSE). As shown in Figure 15, we further scale down KVNAND to an 8-die IFC configuration and explore various grouping schemes for KVNAND-D, where the number of dies in G1 ranges from 1 to 7. The study is conducted on both the 30B MHA and 70B GQA models under two widely adopted quantization settings: W8A8 (8-bit weight and 8-bit activation/KV cache) and W4A16 (4-bit weight and 16-bit activation/KV cache) [81]. Each grid cell in the heatmaps represents the inference latency under different sequence lengths, with red values

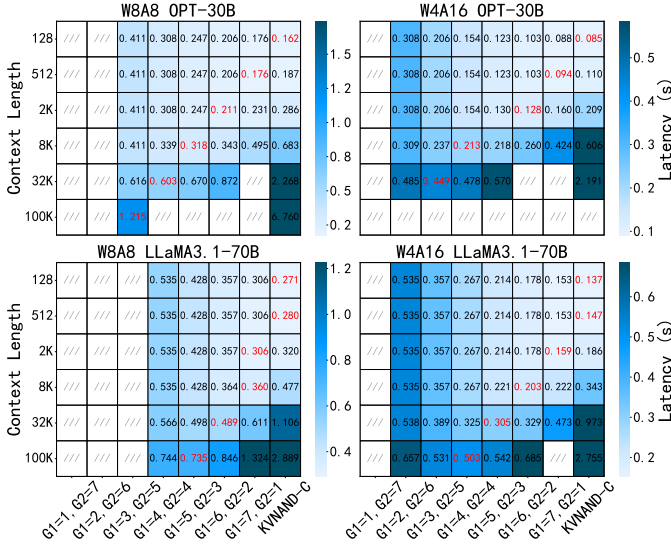


Fig. 15: Latency heatmaps for different KVnAND configurations across model scales, quantization schemes, and context lengths. A blank space indicates OOM.

indicating the best performance. All configurations must also satisfy storage constraints; i.e., the combined size of weights and KV cache must not exceed the available flash capacity. Configurations that violate this constraint are marked as OOM and denoted by blank cells. This OOM constraint is triggered by two primary factors. First, MHA models are highly susceptible to long context lengths due to their large KV cache footprint. Separately, large-scale models with their massive weight parameters are incompatible with configurations that provision too few G1 dies. Our evaluation targets system-level trade-offs of KVnAND, following prior PIM studies [42], instead of RTL simulation, to enable efficient DSE in the early design stage. This DSE-driven approach also enables adaptive on-device, software-defined reconfiguration of KVnAND. When the workload evolves (e.g., model updating or context-length mode switching), the system can be dynamically re-configured by the DSE results to apply the optimal settings. Key observations of our DSE are as follows.

First, the benefit of balanced grouping. For both 30B and 70B models, balanced die allocation between G1 and G2, or adopting the KVnAND-C scheme, generally achieves the lowest latency across a wide range of sequence lengths. For instance, in LLaMA3.1-70B (W4A16), as the required context length increases and attention computation dominates, KVnAND-D gradually outperforms KVnAND-C beyond 2K tokens. In this regime, allocating more IFC dies to G2 for KV storage and attention computation improves performance, with the optimal configuration reaching 4 dies in G2 at 100K tokens. Conversely, in shorter contexts where FFN dominates the latency, KVnAND-C or allocating more dies to G1 is more advantageous. This effect is more pronounced under GQA model, since the larger weight size and smaller KV cache size shift the bottleneck toward weight-related GEMVs.

Second, the impact of quantization. At the same model

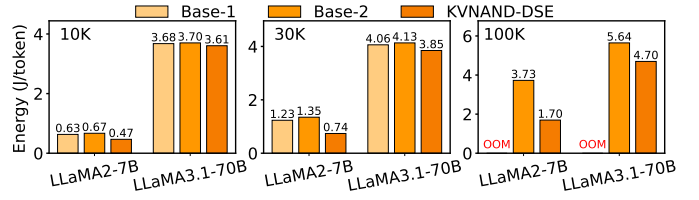


Fig. 16: Energy consumption of per-token decoding.

size and context length, W8A8 quantization favors allocating more dies to G1 due to the larger weight size, while W4A16 favors allocating more dies to G2 as activation and KV cache handling become the dominant bottleneck. In KVnAND designs, W4A16 achieves better performance than W8A8, in contrast to previous work where attention is calculated on the NPU [81] due to IFC-accelerated attention.

Third, scaling with sequence length. At short contexts ($\leq 5K$), performance is relatively insensitive to configuration because KV cache overhead is small and weight-related GEMVs dominate. As sequence length increases to 50K–100K, latency rises sharply for unbalanced groupings due to intensified KV cache traffic. This highlights the importance of tuning the G1/G2 allocation according to target context length.

Takeaway 1: Balanced die allocation is key. KVnAND-D is better suited for long-context applications, offering higher performance and reliability after careful design-space configuration, while KVnAND-C is more favorable at shorter contexts.

Takeaway 2: Quantization shifts bottlenecks. W4A16 favors G2-heavy allocations and yields better performance under IFC-accelerated attention, whereas W8A8 favors G1-heavy allocations due to larger weight size.

Energy Efficiency. Figure 16 reports the energy consumption for two representative models under different context lengths. Overall, KVnAND after the proposed DSE achieves consistently lower energy consumption than both baselines, with the advantage becoming more pronounced at longer contexts. At 10K tokens, KVnAND reduces energy consumption to $0.75\times$ and $0.98\times$ of Base-1 for the LLaMA2-7B and LLaMA3.1-70B models, respectively, and at 100K tokens, the savings further improve to $0.46\times$ and $0.83\times$ of Base-2. These reductions stem from significantly less data movement: IFC computation eliminates repeated transfers of KV cache data. GQA models inherently reduce energy consumption for long-context inference owing to smaller KV head numbers.

KVnAND-D still requires some external data transfers, but the overall volume is much smaller due to offloaded QKV generation and reduced intermediate data movement. In addition, flash write energy is amortized across multiple accumulated updates, keeping the overall cost modest.

In terms of thermal throttling, prior analysis of 3D-stacked IFC dies [67] has shown that their power density is acceptable. KVnAND adopts the same flash configuration and maintains comparable compute resources on the logic die, so its thermal profile is expected to remain similar. The additional buffer

required in KVNAND-C introduces only a minor overhead as listed in Table 1, confirming that thermal impact is negligible.

C. Cost Analysis

The primary cost components of KVNAND come from SLC-mode 3D NAND flash, hybrid bonding, and peripheral dies. We base our estimation on YTMC’s 128-layer TLC flash (8.5 Gb/mm²) [69], which already includes hybrid bonding and peripheral die costs, priced at \$0.11/GB. Converting to SLC mode (1.8 Gb/mm² density) results in \$0.52/GB. Considering the $1.22\times$ area and $2\times$ hybrid bonding I/O per die required for page buffers, the yield can be conservatively estimated as 58% (normal, 80% yield base), leading to \$0.72/GB [67].

For 8-channel, 8-die configuration of KVNAND-D-(4+4), flash cost is approximately \$92.16 for combined weight and KV cache storage. This is significantly lower than DRAM-based IFC designs; for the same model and context-length KV cache capacity, replacing flash with LPDDR5 (\$4.62/GB [56]) would increase the memory cost by more than $2\times$ (\$295.68).

The additional logic in IFC-enabled KV dies incurs minimal area overhead ($< 3\%$ per die) and negligible cost compared to the flash array. While KVNAND requires both weight and KV dies to integrate IFC logic, this reuse of the existing peripheral die layout ensures that the incremental manufacturing cost remains small. The adoption of multi-die packages and HB interconnects follows established 3D NAND manufacturing practices, and the associated packaging cost is estimated to be $< 15\%$ of raw die cost [17].

D. Reliability Evaluation

To evaluate the long-term reliability, we consider a 65B-parameter model [71] running on KVNAND at 3 tokens/s throughput. Under the pessimistic assumption of continuous decoding for 5 years, the total KV cache generated amounts to approximately 143 TB. For the 8-die, 16 GB configuration of KVNAND-D, which supports up to $\sim 50K$ context length, this corresponds to about 1K P/E cycles, well below the nominal endurance budget of 100K P/E cycles in SLC flash [2]. Conservatively, we leave roughly 50K cycles as usable margin for page-read disturbance refresh.

We model the flash-based KV cache read/write process and track the cumulative page-read counts across blocks. At a 50K context length (with 1K input and 1K output tokens per request), the total page read counts per block can reach 4×10^6 , which exceeds the intrinsic disturbance limit of 10^6 [83]. However, KVNAND’s data mapping and parallelization schemes substantially mitigate this stress. In KVNAND-C-16, head-parallel KV generation reduces maximum per-block page-read counts by approximately $128\times$ ($\sim \frac{k \times \text{page_size}}{KV\text{buf_size}}$). In KVNAND-D, separating weight and KV dies further lowers PGRD amplification, achieving an estimated $2560\times$ reduction. These reductions provide ample reliability margin for sustained long-context inference workloads, ensuring that KV cache storage remains robust throughout device lifetime.

VI. RELATED WORK

In-Flash Computing for LLMs. To enable the deployment of large-scale LLMs such as LLaMA-70B on memory-constrained consumer devices, recent research has explored integrating compute capabilities directly into flash storage, a trend commonly referred to as in-flash computing [40], [67], [81]. IFC addresses two major bottlenecks of SSD-based offloading [5], [62], [65]: the limited transmission bandwidth between flash and accelerators, and the low internal read throughput of flash arrays. The common principle is to offload memory-bound GEMV operations to logic placed close to the flash array. Among them, Lincoln [67] enhances performance with speculative decoding, reducing iteration latency by generating multiple tokens per step. They further propose a round-robin row distribution layout for weight data, tailored to the distinct compute and memory access characteristics of the NPU and flash. Cambricon-LLM [81] strategically partitions GEMV operations, keeping some linear layers on the NPU while mapping others to flash, thereby balancing channel utilization. AiF [40] further improves throughput at the circuit level by exploiting the sequential access pattern of weights, introducing charge-recycling reads that bypass conventional wordline precharge/discharge cycles to reduce read latency. These techniques are orthogonal to our work and can be directly incorporated into KVNAND. Distinct from prior IFC architectures, KVNAND offloads the KV cache to flash, eliminating the high cost and energy overhead of DRAM for KV storage and transfers. Even on devices that inevitably integrate DRAM for system software and embeddings, KVNAND remains applicable and reduces DRAM pressure, enabling efficient long-context LLM inference. In particular, KVNAND-D can leverage available DRAM as a large KV buffer, reducing on-die SRAM requirements.

LLM Acceleration. Numerous quantization-based techniques [15], [18], [22], [23], [46], [47], [73], [75], [82] have been proposed to reduce model size and enable LLM deployment on edge devices. These methods are orthogonal to KVNAND and are incorporated into our DSE framework as input configurations for design optimization. Beyond quantization, sparse attention has emerged as another widely adopted approach to lower KV cache access and computation demand [8], [9], [43], [48], [49], [63], [66], [68], [76], [87]. Sparse attention is based on the observation that not all tokens contribute equally to the final attention output. Determining which tokens to prune often introduces algorithmic complexity and potential accuracy degradation. Building on its ability to mitigate KV cache fragmentation, the KVNAND architecture has the potential to support diverse sparsity schemes as well.

VII. CONCLUSION

This paper presents KVNAND, a DRAM-free NPU-IFC architecture that stores both model weights and the KV cache in compute-enabled NAND flash. By co-designing dataflow, head-group parallelism, and page-level KV cache mapping, KVNAND effectively addresses the challenges of KV storage in flash. Evaluation across diverse models and context lengths

shows that KVNAND achieves up to $2.3\times$ speedup and $0.75\times$ energy consumption relative to SoTA DRAM-based IFC designs at 10K context length compared to baseline. Furthermore, KVNAND fully resolves the OOM barrier at 100K context lengths, delivering an inference throughput of 10 tokens/s on LLaMA3.1-8B. Beyond performance, the low cost and energy efficiency make KVNAND a critical enabling technology for deploying long-context LLMs on edge devices. We hope KVNAND motivates further exploration of IFC for next-generation LLM systems.

REFERENCES

- [1] “Keynote Talks,” in *2019 IEEE/ACM Workshop on Memory Centric High Performance Computing (MCHPC)*, 2019, pp. 7–7.
- [2] “New Storage Class Memory Solution Accelerates Non-Relational Database Performance,” 2022.
- [3] J. Ainslie, J. Lee-Thorp, M. d. Jong, Y. Zemlyanskiy, F. Lebrón, and S. Sanghai, “GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints,” 2023, [_eprint: 2305.13245](https://arxiv.org/abs/2305.13245). [Online]. Available: <https://arxiv.org/abs/2305.13245>
- [4] I. Alam and P. Gupta, “COMET: On-die and In-controller Collaborative Memory ECC Technique for Safer and Stronger Correction of DRAM Errors,” in *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2022, pp. 124–136.
- [5] K. Alizadeh, I. Mirzadeh, D. Belenko, K. Khatamifard, M. Cho, C. C. D. Mundo, M. Rastegari, and M. Farajtabar, “LLM in a flash: Efficient Large Language Model Inference with Limited Memory,” 2024, [_eprint: 2312.11514](https://arxiv.org/abs/2312.11514). [Online]. Available: <https://arxiv.org/abs/2312.11514>
- [6] Y. Cai, Y. Luo, S. Ghose, and O. Mutlu, “Read Disturb Errors in MLC NAND Flash Memory: Characterization, Mitigation, and Recovery,” in *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 2015, pp. 438–449.
- [7] Y. Cai, G. Yalcin, O. Mutlu, and E. F. Haratsch, “Error Analysis and Retention-Aware Error Management for NAND Flash Memory,” vol. 17, no. 1, 2013.
- [8] S. Chaudhari, V. Mithal, G. Polatkan, and R. Ramanath, “An attentive survey of attention models,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 12, no. 5, pp. 1–32, 2021.
- [9] B. Chen, T. Dao, E. Winsor, Z. Song, A. Rudra, and C. Ré, “Scatterbrain: Unifying sparse and low-rank attention,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 17413–17426, 2021.
- [10] Q. Chen, L. Qin, J. Liu, D. Peng, J. Guan, P. Wang, M. Hu, Y. Zhou, T. Gao, and W. Che, “Towards reasoning era: A survey of long chain-of-thought for reasoning large language models,” *arXiv preprint arXiv:2503.09567*, 2025.
- [11] M. Chun, J. Lee, S. Lee, M. Kim, and J. Kim, “PiF: in-flash acceleration for data-intensive applications,” in *Proceedings of the 14th ACM Workshop on Hot Topics in Storage and File Systems*, ser. HotStorage ’22. New York, NY, USA: Association for Computing Machinery, 2022, pp. 106–112, event-place: Virtual Event. [Online]. Available: <https://doi.org/10.1145/3538643.3539742>
- [12] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord, “Think you have Solved Question Answering? Try ARC, the A12 Reasoning Challenge,” *arXiv:1803.05457v1*, 2018.
- [13] G. Comanici, E. Bieber, M. Schaeckermann, I. Pasupat, N. Sachdeva, I. Dhillon, M. Blistein, O. Ram, D. Zhang, E. Rosen *et al.*, “Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities,” *arXiv preprint arXiv:2507.06261*, 2025.
- [14] DeepSeek-AI, “DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning,” 2025, [_eprint: 2501.12948](https://arxiv.org/abs/2501.12948). [Online]. Available: <https://arxiv.org/abs/2501.12948>
- [15] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, “Qlora: Efficient finetuning of quantized llms,” *Advances in neural information processing systems*, vol. 36, pp. 10088–10115, 2023.
- [16] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, “NVSim: A Circuit-Level Performance, Energy, and Area Model for Emerging Nonvolatile Memory,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 7, pp. 994–1007, 2012.
- [17] Y. Feng and K. Ma, “Chiplet actuary: a quantitative cost model and multi-chiplet architecture exploration,” in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, ser. DAC ’22. ACM, Jul. 2022, pp. 121–126. [Online]. Available: <http://dx.doi.org/10.1145/3489517.3530428>
- [18] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, “Gptq: Accurate post-training quantization for generative pre-trained transformers,” *arXiv preprint arXiv:2210.17323*, 2022.
- [19] Z. Fu, X. Guo, W. Zeng, S. Zhong, Y. Zhang, P. Chen, R. Wang, L. Ye, and M. Li, “H2eal: Hybrid-bonding architecture with hybrid sparse attention for efficient long-context llm inference,” *arXiv preprint arXiv:2508.16653*, 2025.
- [20] W. Gao and S. Simmons, “A study on the VLSI implementation of ECC for embedded DRAM,” in *CCECE 2003 - Canadian Conference on Electrical and Computer Engineering. Toward a Caring and Humane Technology (Cat. No.03CH37436)*, vol. 1, 2003, pp. 203–206 vol.1.
- [21] Y. Gu, A. Khadem, S. Umesh, N. Liang, X. Servot, O. Mutlu, R. Iyer, and R. Das, “PIM Is All You Need: A CXL-Enabled GPU-Free System for Large Language Model Inference,” in *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ser. ASPLOS ’25. ACM, Mar. 2025, pp. 862–881. [Online]. Available: <http://dx.doi.org/10.1145/3676641.3716267>
- [22] Z. Guan, H. Huang, Y. Su, H. Huang, N. Wong, and H. Yu, “Aptq: Attention-aware post-training mixed-precision quantization for large language models,” in *Proceedings of the 61st ACM/IEEE Design Automation Conference*, 2024, pp. 1–6.
- [23] C. Guo, J. Tang, W. Hu, J. Leng, C. Zhang, F. Yang, Y. Liu, M. Guo, and Y. Zhu, “Olive: Accelerating large language models via hardware-friendly outlier-victim pair quantization,” in *Proceedings of the 50th Annual International Symposium on Computer Architecture*, 2023, pp. 1–15.
- [24] Y. He, H. Mao, C. Giannoula, M. Sadrosadati, J. Gómez-Luna, H. Li, X. Li, Y. Wang, and O. Mutlu, “PAPI: Exploiting Dynamic Parallelism in Large Language Model Decoding with a Processing-In-Memory-Enabled Computing System,” in *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ser. ASPLOS ’25. New York, NY, USA: Association for Computing Machinery, 2025, pp. 766–782, event-place: Rotterdam, Netherlands. [Online]. Available: <https://doi.org/10.1145/3676641.3716009>
- [25] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, “Measuring Massive Multitask Language Understanding,” *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.
- [26] G. Heo, S. Lee, J. Cho, H. Choi, S. Lee, H. Ham, G. Kim, D. Mahajan, and J. Park, “NeuPIMs: NPU-PIM Heterogeneous Acceleration for Batched LLM Inferencing,” in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ser. ASPLOS ’24. New York, NY, USA: Association for Computing Machinery, 2024, pp. 722–737, event-place: La Jolla, CA, USA. [Online]. Available: <https://doi.org/10.1145/3620666.3651380>
- [27] Y. Hu, H. Jiang, D. Feng, L. Tian, S. Zhang, J. Liu, W. Tong, Y. Qin, and L. Wang, “Achieving page-mapping FTL performance at block-mapping FTL cost by hiding address translation,” in *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, 2010, pp. 1–12.
- [28] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. d. l. Casas, E. B. Hanna, F. Bressand, G. Lengyel, G. Bour, G. Lample, L. R. Lavaud, L. Saulnier, M.-A. Lachaux, P. Stock, S. Subramanian, S. Yang, S. Antoniak, T. L. Scao, T. Gervet, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed, “Mixtral of Experts,” 2024, [_eprint: 2401.04088](https://arxiv.org/abs/2401.04088). [Online]. Available: <https://arxiv.org/abs/2401.04088>
- [29] H. Jin, X. Han, J. Yang, Z. Jiang, Z. Liu, C.-Y. Chang, H. Chen, and X. Hu, “LLM Maybe LongLM: Self-Extend LLM Context Window Without Tuning,” Jul. 2024, [arXiv:2401.01325 \[cs\]](https://arxiv.org/abs/2401.01325). [Online]. Available: <http://arxiv.org/abs/2401.01325>
- [30] M. Kang, W. Lee, and S. Kim, “Subpage-Aware Solid State Drive for Improving Lifetime and Performance,” *IEEE Transactions on Computers*, vol. 67, no. 10, pp. 1492–1505, 2018.
- [31] T. Kawaura, “Non-Volatile Memory and Control with Improved Partial Page Program Capability,” Sep. 2006, published: U.S.

- Patent Application Publication. [Online]. Available: <https://www.freepatentsonline.com/y2006/0203561.html>
- [32] J.-H. Kim, S.-H. Kim, and J.-S. Kim, "Subpage programming for extending the lifetime of NAND flash memory," in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2015, pp. 555–560.
 - [33] M. Kim, J. Song, and C. H. Kim, "Reliability Characterization of Logic-Compatible NAND Flash Memory based Synapses with 3-bit per Cell Weights and 1uA Current Steps," in *2020 IEEE International Reliability Physics Symposium (IRPS)*, 2020, pp. 1–4.
 - [34] S. Kim, Y. Jin, G. Sohn, J. Bae, T. J. Ham, and J. W. Lee, "Behemoth: A Flash-centric Training Accelerator for Extreme-scale DNNs," in *19th USENIX Conference on File and Storage Technologies (FAST 21)*. USENIX Association, Feb. 2021, pp. 371–385. [Online]. Available: <https://www.usenix.org/conference/fast21/presentation/kim>
 - [35] H. Y. Koh, J. Ju, M. Liu, and S. Pan, "An empirical survey on long document summarization: Datasets, models, and metrics," *ACM computing surveys*, vol. 55, no. 8, pp. 1–35, 2022.
 - [36] T. Kouchi, N. Kumazaki, M. Yamaoka, S. Bushnaq, T. Kodama, Y. Ishizaki, Y. Deguchi, A. Sugahara, A. Imamoto, N. Asaoka, R. Iso-mura, T. Handa, J. Sato, H. Komai, A. Okuyama, N. Kanagawa, Y. Kajiyama, Y. Terada, H. Ohnishi, H. Yabe, C. Hsu, M. Kakoi, and M. Yoshihara, "13.5 A 128Gb 1b/Cell 96-Word-Line-Layer 3D Flash Memory to Improve Random Read Latency with tPROG=75us and tR=4us," in *2020 IEEE International Solid-State Circuits Conference - (ISSCC)*, 2020, pp. 226–228.
 - [37] B. C. Kuszmaul, D. S. Henry, and G. H. Loh, "A comparison of scalable superscalar processors," in *Proceedings of the Eleventh Annual ACM Symposium on Parallel Algorithms and Architectures*, ser. SPAA '99. New York, NY, USA: Association for Computing Machinery, 1999, pp. 126–137, event-place: Saint Malo, France. [Online]. Available: <https://doi.org/10.1145/305619.305633>
 - [38] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, and I. Stoica, "Efficient Memory Management for Large Language Model Serving with PagedAttention," 2023, eprint: 2309.06180. [Online]. Available: <https://arxiv.org/abs/2309.06180>
 - [39] H. Lee, M. Kim, D. Min, J. Kim, J. Back, H. Yoo, J.-H. Lee, and J. Kim, "3D-FPIM: An Extreme Energy-Efficient DNN Acceleration System Using 3D NAND Flash-Based In-Situ PIM Unit," in *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2022, pp. 1359–1376.
 - [40] J. Lee, H. Kim, S. Oh, M. Chun, M. Kim, and J. Kim, "Aif: Accelerating on-device llm inference using in-flash processing," in *Proceedings of the 52nd Annual International Symposium on Computer Architecture*, 2025, pp. 529–543.
 - [41] W. Lee, J. Lee, J. Seo, and J. Sim, "InfiniGen: Efficient Generative Inference of Large Language Models with Dynamic KV Cache Management," 2024, eprint: 2406.19707. [Online]. Available: <https://arxiv.org/abs/2406.19707>
 - [42] C. Li, Y. Yin, X. Wu, J. Zhu, Z. Gao, D. Niu, Q. Wu, X. Si, Y. Xie, C. Zhang, and G. Sun, "H²-LLM: Hardware-Dataflow Co-Exploration for Heterogeneous Hybrid-Bonding-based Low-Batch LLM Inference," in *Proceedings of the 52nd Annual International Symposium on Computer Architecture*. Tokyo Japan: ACM, Jun. 2025, pp. 194–210. [Online]. Available: <https://dl.acm.org/doi/10.1145/3695053.3731008>
 - [43] 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," *arXiv preprint arXiv:2412.19442*, 2024.
 - [44] J. Li, J. Xu, S. Huang, Y. Chen, W. Li, J. Liu, Y. Lian, J. Pan, L. Ding, H. Zhou, Y. Wang, and G. Dai, "Large Language Model Inference Acceleration: A Comprehensive Hardware Perspective," 2025, eprint: 2410.04466. [Online]. Available: <https://arxiv.org/abs/2410.04466>
 - [45] B. Lin, C. Zhang, T. Peng, H. Zhao, W. Xiao, M. Sun, A. Liu, Z. Zhang, L. Li, X. Qiu, S. Li, Z. Ji, T. Xie, Y. Li, and W. Lin, "Infinite-LLM: Efficient LLM Service for Long Context with DistAttention and Distributed KVCache," 2024, eprint: 2401.02669. [Online]. Available: <https://arxiv.org/abs/2401.02669>
 - [46] J. Lin, J. Tang, H. Tang, S. Yang, W.-M. Chen, W.-C. Wang, G. Xiao, X. Dang, C. Gan, and S. Han, "Awq: Activation-aware weight quantization for on-device llm compression and acceleration," *Proceedings of machine learning and systems*, vol. 6, pp. 87–100, 2024.
 - [47] S.-y. Liu, Z. Liu, X. Huang, P. Dong, and K.-T. Cheng, "Llm-fp4: 4-bit floating-point quantized transformers," *arXiv preprint arXiv:2310.16836*, 2023.
 - [48] Z. Liu, A. Desai, F. Liao, W. Wang, V. Xie, Z. Xu, A. Kyrillidis, and A. Shrivastava, "Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time," *Advances in Neural Information Processing Systems*, vol. 36, pp. 52342–52364, 2023.
 - [49] Z. Liu, J. Wang, T. Dao, T. Zhou, B. Yuan, Z. Song, A. Shrivastava, C. Zhang, Y. Tian, C. Re *et al.*, "Deja vu: Contextual sparsity for efficient llms at inference time," in *International Conference on Machine Learning*. PMLR, 2023, pp. 22137–22176.
 - [50] A. LlamaTeam, "The Llama 3 Herd of Models," 2024, eprint: 2407.21783. [Online]. Available: <https://arxiv.org/abs/2407.21783>
 - [51] M3I technology corporation, "ONFI I/O," [Online]. Available: <https://www.m3itech.com/zh-hans/product/onfi/>
 - [52] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer Sentinel Mixture Models," 2016, eprint: 1609.07843.
 - [53] R. Micheloni, L. Crippa, and A. Marelli, *Inside NAND flash memories*, Jan. 2010, publication Title: Inside NAND Flash Memories. [Online]. Available: <https://doi.org/10.1007/978-90-481-9431-5>
 - [54] Micron Technology, Inc., "Automotive LPDDR5X SDRAM," Jun. 2025. [Online]. Available: <https://www.micron.com/content/dam/micron/global/secure/products/data-sheet/dram/mobile-dram/lpddr5/auto-embedded/561b-y5bp-qdp-8dp-auto-lpddr5x.pdf>
 - [55] V. Moschiano, "Tutorial: 3D Flash Memory from Technology to the System: Past, Present and Future Developments," in *2024 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. TUTORIAL, 2024, pp. 1–1.
 - [56] Mouser Electronics, "SDRAM - LPDDR5 DRAM," [Online]. Available: https://www.mouser.com/c/semiconductors/memory-ics/?srsltid=AfmBOoqCBw73OQ14CcLZ_mF-YBy_6kUDk7FTcDk7tatBvFdGQeQDR-3N&type=SDRAM%20-%20LPDDR5&utm_source=chatgpt.com
 - [57] D. Na, J.-w. Lee, S.-K. Lee, H. Cho, J. Lee, M. Yang, E. Song, A. Kavala, T. Kim, D.-S. Jang, Y. Jo, J.-Y. Shin, B.-K. Chun, T.-s. Lee, B. Jeong, C.-W. Yoon, D. Kang, S. Lee, J. Ihm, D. S. Byeon, J. Lee, and J. H. Song, "A 1.8-Gb/s/Pin 16-Tb NAND Flash Memory Multi-Chip Package With F-Chip for High-Performance and High-Capacity Storage," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 4, pp. 1129–1140, 2021.
 - [58] Open NAND Flash Interface (ONFI) Workgroup, "Onfi specifications," <https://onfi.org/specs.html>, accessed: 2025-11-16.
 - [59] OpenAI, "Introducing GPT-5 Our smartest, fastest, most useful model yet, with built-in thinking that puts expert-level intelligence in everyone's hands," Aug. 2025. [Online]. Available: <https://openai.com/zh-Hans-CN/index/introducing-gpt-5/>
 - [60] Y. Ouyang, S. Yang, D. Yin, X. Huang, Z. Wang, S. Yang, K. Han, and Z. Xia, "Excellent Reliability of Xtacking™ Bonding Interface," in *2021 IEEE International Reliability Physics Symposium (IRPS)*, 2021, pp. 1–6.
 - [61] X. Pan, E. Li, Q. Li, S. Liang, Y. Shan, K. Zhou, Y. Luo, X. Wang, and J. Zhang, "InstAttention: In-Storage Attention Offloading for Cost-Effective Long-Context LLM Inference," in *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2025, pp. 1510–1525.
 - [62] J. Rasley, S. Rajbhandari, O. Ruwase, and Y. He, "DeepSpeed: System Optimizations Enable Training Deep Learning Models with Over 100 Billion Parameters," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 3505–3506, event-place: Virtual Event, CA, USA. [Online]. Available: <https://doi.org/10.1145/3394486.3406703>
 - [63] L. Ribar, I. Chelombiev, L. Hudlass-Galley, C. Blake, C. Luschi, and D. Orr, "Sparq attention: Bandwidth-efficient llm inference," *arXiv preprint arXiv:2312.04985*, 2023.
 - [64] N. Shazeer, "Fast transformer decoding: One write-head is all you need," *arXiv preprint arXiv:1911.02150*, 2019.
 - [65] Y. Sheng, L. Zheng, B. Yuan, Z. Li, M. Ryabinin, D. Y. Fu, Z. Xie, B. Chen, C. Barrett, J. E. Gonzalez, P. Liang, C. Ré, I. Stoica, and C. Zhang, "FlexGen: High-Throughput Generative Inference of Large Language Models with a Single GPU," 2023, eprint: 2303.06865. [Online]. Available: <https://arxiv.org/abs/2303.06865>
 - [66] H. Sun, L.-W. Chang, W. Bao, S. Zheng, N. Zheng, X. Liu, H. Dong, Y. Chi, and B. Chen, "Shadowkv: Kv cache in shadows for high-throughput long-context llm inference," *arXiv preprint arXiv:2410.21465*, 2024.

- [67] W. Sun, M. Gao, Z. Li, A. Zhang, I. Y. Chou, J. Zhu, S. Wei, and L. Liu, "Lincoln: Real-Time 50T/100B LLM Inference on Consumer Devices with LPDDR-Interfaced, Compute-Enabled Flash Memory," in *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, Las Vegas, NV, USA: IEEE, Mar. 2025, pp. 1734–1750. [Online]. Available: <https://ieeexplore.ieee.org/document/10946816/>
- [68] H. Tang, Y. Lin, J. Lin, Q. Han, S. Hong, Y. Yao, and G. Wang, "Razorattention: Efficient kv cache compression through retrieval heads," *arXiv preprint arXiv:2407.15891*, 2024.
- [69] TechPowerUp, "Zhitai TiPRO7000 2 TB," Aug. 2025. [Online]. Available: <https://www.techpowerup.com/ssd-specs/zhitai-tipro7000-2-tb.d1192>
- [70] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "LLaMA: Open and Efficient Foundation Language Models," 2023, _eprint: 2302.13971. [Online]. Available: <https://arxiv.org/abs/2302.13971>
- [71] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom, "Llama 2: Open Foundation and Fine-Tuned Chat Models," 2023, _eprint: 2307.09288. [Online]. Available: <https://arxiv.org/abs/2307.09288>
- [72] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," 2023, _eprint: 1706.03762. [Online]. Available: <https://arxiv.org/abs/1706.03762>
- [73] H. Wang, S. Ma, L. Dong, S. Huang, H. Wang, L. Ma, F. Yang, R. Wang, Y. Wu, and F. Wei, "Bitnet: Scaling 1-bit transformers for large language models," *arXiv preprint arXiv:2310.11453*, 2023.
- [74] T. Wang, M. Huang, F. Li, L. Chen, J. Zhang, and J. Ren, "Dynakv: Enabling accurate and efficient long-sequence llm decoding on smartphones," *arXiv preprint arXiv:2511.07427*, 2025.
- [75] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han, "Smoothquant: Accurate and efficient post-training quantization for large language models," in *International conference on machine learning*. PMLR, 2023, pp. 38 087–38 099.
- [76] G. Xiao, Y. Tian, B. Chen, S. Han, and M. Lewis, "Efficient streaming language models with attention sinks," *arXiv preprint arXiv:2309.17453*, 2023.
- [77] T. Xie, J. Zhao, Z. Wan, Z. Zhang, Y. Wang, R. Wang, R. Huang, and M. Li, "ReaLM: Reliable and Efficient Large Language Model Inference with Statistical Algorithm-Based Fault Tolerance," *ArXiv*, vol. abs/2503.24053, 2025. [Online]. Available: <https://api.semanticscholar.org/CorpusID:277468166>
- [78] J. Xu, Z. Li, W. Chen, Q. Wang, X. Gao, Q. Cai, and Z. Ling, "On-Device Language Models: A Comprehensive Review," 2024, _eprint: 2409.00088. [Online]. Available: <https://arxiv.org/abs/2409.00088>
- [79] K. Yanagidaira, M. Sako, Y. Hirashima, J. Matsuno, Y. Higashi, Y. Shimizu, A. Imamoto, K. Kawaguchi, K. Tabata, T. Nakano *et al.*, "A 1tb 3b/cell 3d-flash memory with a 29%-improved-energy-efficiency read operation and 4.8 gb/s power-isolated low-tapped-termination i/os," in *2025 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 68. IEEE, 2025, pp. 1–3.
- [80] D. B. L. Yolanda, "Wafer to wafer bonding to increase memory density," in *2022 China Semiconductor Technology International Conference (CSTIC)*, 2022, pp. 1–4.
- [81] Z. Yu, S. Liang, T. Ma, Y. Cai, Z. Nan, D. Huang, X. Song, Y. Hao, J. Zhang, T. Zhi *et al.*, "Cambricon-llm: A chiplet-based hybrid architecture for on-device inference of 70b llm," in *2024 57th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2024, pp. 1474–1488.
- [82] Z. Yuan, L. Niu, J. Liu, W. Liu, X. Wang, Y. Shang, G. Sun, Q. Wu, J. Wu, and B. Wu, "Rptq: Reorder-based post-training quantization for large language models," *arXiv preprint arXiv:2304.01089*, 2023.
- [83] C. Zambelli, P. Olivo, L. Crippa, A. Marelli, and R. Micheloni, "Uniform and concentrated read disturb effects in mid-1X TLC NAND flash memories for enterprise solid state drives," in *2017 IEEE International Reliability Physics Symposium (IRPS)*, 2017, pp. PM–5.1–PM–5.4.
- [84] C. Zhang, X. Dai, Y. Wu, Q. Yang, Y. Wang, R. Tang, and Y. Liu, "A survey on multi-turn interaction capabilities of large language models," *arXiv preprint arXiv:2501.09959*, 2025.
- [85] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, T. Mihaylov, M. Ott, S. Shleifer, K. Shuster, D. Simig, P. S. Koura, A. Sridhar, T. Wang, and L. Zettlemoyer, "OPT: Open Pre-trained Transformer Language Models," 2022, _eprint: 2205.01068. [Online]. Available: <https://arxiv.org/abs/2205.01068>
- [86] Y. Zhang, L. Jin, D. Jiang, X. Zou, H. Liu, and Z. Huo, "A Novel Read Scheme for Read Disturbance Suppression in 3D NAND Flash Memory," *IEEE Electron Device Letters*, vol. 38, no. 12, pp. 1669–1672, 2017.
- [87] Z. Zhang, Y. Sheng, T. Zhou, T. Chen, L. Zheng, R. Cai, Z. Song, Y. Tian, C. Ré, C. Barrett *et al.*, "H2o: Heavy-hitter oracle for efficient generative inference of large language models," *Advances in Neural Information Processing Systems*, vol. 36, pp. 34 661–34 710, 2023.
- [88] X. Zhao, S. Yang, K. Xie, Y. Feng, Q. Wang, P. Sang, X. Zhan, J. Wu, and J. Chen, "Error Bits Recovering in 3D NAND Flash Memory: A Novel State-Shift Re-Program (SRP) Scheme," in *2023 IEEE International Conference on Integrated Circuits, Technologies and Applications (ICTA)*, 2023, pp. 99–100.
- [89] Z. Zhou, X. Ning, K. Hong, T. Fu, J. Xu, S. Li, Y. Lou, L. Wang, Z. Yuan, X. Li, S. Yan, G. Dai, X.-P. Zhang, Y. Dong, and Y. Wang, "A Survey on Efficient Inference for Large Language Models," 2024, _eprint: 2404.14294. [Online]. Available: <https://arxiv.org/abs/2404.14294>