DRAGON: A 28nm 5.12ms-response and 42.83token/s Digital CIM-based Multi-Phase Accelerator for Hybrid-Intensive Retrieval-Augmented Generation

Zihan Wu[1], Yiqi Wang[1], Zhen He[1], Huiming Han[1], Yang Hu[1], Fengbin Tu[2*], Shouyi Yin[1*]
[1]BNRist, Tsinghua University, Beijing; [2]The Hong Kong University of Science and Technology, Hong Kong
[*]Email: fengbintu@ust.hk, yinsy@tsinghua.edu.cn

***Abstract*—This paper presents DRAGON, a Digital compute-in-memory (DCIM)-based accelerator for Retrieval-Augmented Generation (RAG) with three features: 1) For memory-intensive *Retrieval* phase, a dual-step progressive speculation unit (DPSU) reduces EMA caused by distant database-index vectors, saving 3.53× TTFT. 2) For compute-intensive *Prefill* phase, a maximum-reuse KV cache manager (MRCM) reuses the KV cache of frequent documents, saving 2.27× TTFT. 3) For memory-intensive *Decoding* phase, an entropy-aware code searcher (EACS) enables codec-free computation on low-bit entropy-based quantization (EQ) data, reducing TPOT by 6.45×. The fabricated chip reaches 5.12ms-response and 42.83 token/s on OPT-1.3B-RAG, achieving 1.42∼7.28× speedup over state-of-the-art LLM accelerators.**

***Index Terms*—Retrieval-augmented generation (RAG), digital compute-in-memory, SRAM, reconfigurable architecture**

I. Introduction

Large language models (LLMs) have shown superb performance but still suffer from data staleness and hallucination. RAG systems combine LLM with external database to improve generation quality. As illustrated in Fig. 1, RAG consists of three phases, which are evaluated by time-to-first-token (TTFT, for *Retrieval* + *Prefill*) and time-per-output-token (TPOT, for *Decoding*). These phases exhibit different intensive properties, introducing new **challenges** for prior LLM processors [1]–[6]: **1)** *Retrieval* phase is memory-intensive since it needs to load all database-index vectors (DVs) from off-chip to search for m most relevant documents via dot-product similarity. However, the distant DVs cause unnecessary external memory access (EMA). **2)** *Prefill* is a compute-intensive phase that processes user prompt appended with m relevant documents. Although some documents are frequently retrieved [7], their corresponding KV cache still needs recomputation due to lack of cross-attention. **3)** *Decoding* phase is memory-intensive because it loads all model weights and KV cache in each iteration. Low-bitwidth non-uniform quantization (NUQ) is widely used [3]–[5], but compression potential remains due to the fixed-length coding format. DCIM has proven efficient for compute-intensive encoder-based transformers [8], [9], however, it cannot directly gain benefits on hybrid-intensive RAG system.

To address these challenges, we propose DRAGON with three key **features**: **1)** In *Retrieval* phase, DPSU first rapidly excludes distant DVs by hash method in DCIM's CAM-mode, then bit-serially loads the remaining DVs with early termination, reducing vector- and bit-level EMA. **2)** In *Prefill* phase, by comparing retrieved document sequence with pre-computed KV, MRCM reuses the KV cache of prefix-matched documents and self-related tokens, thus alleviating computational burden. **3)** In *Decoding* phase, EQ compresses NUQ by allocating shorter length to high-probability codes, yielding theoretically minimal bit usage [10]. EACS performs codec-free MAC in DCIM's CAM-mode, reducing EMA via low-bit transmission.

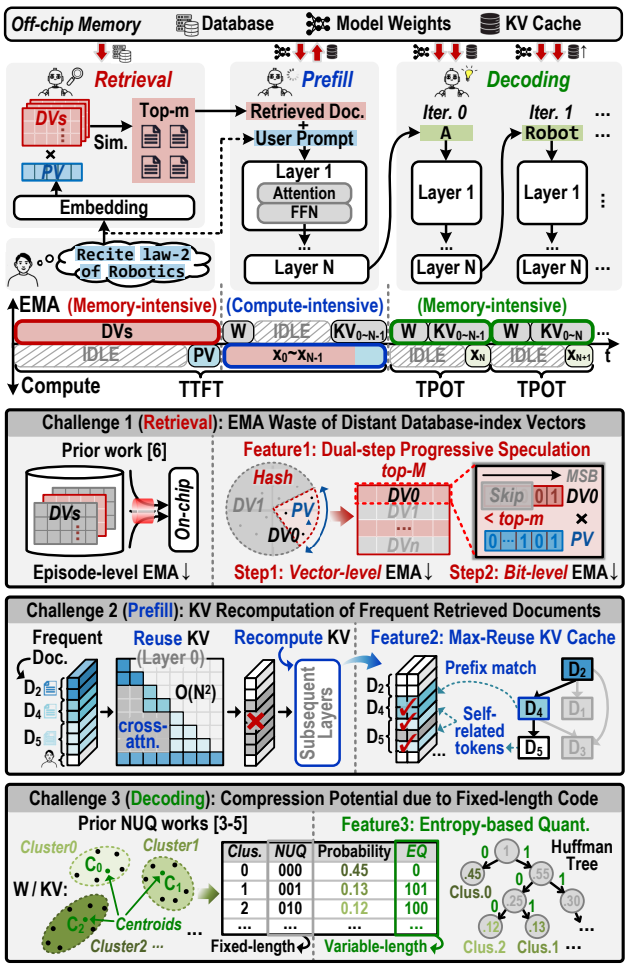


Fig. 1. Multiple phases of RAG system exhibit different compute/memory-intensive properties. Three challenges and our key ideas are highlighted.

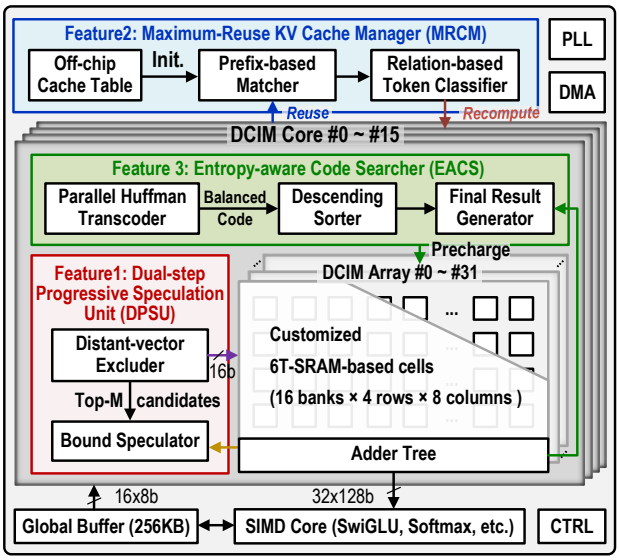


Fig. 2. Overall architecture of DRAGON and its three main features.

II. Proposed DRAGON Architecture

The hybrid-intensive properties of the RAG system drive us to rethink the design philosophy of DCIM. Our key insight is that DCIM with MAC/CAM reconfigurability can address the challenges across different phases. Specifically, MAC-mode is used for the compute-intensive phase just like prior DCIM accelerators, while CAM-mode is for the memory-intensive phase to support memory-efficient algorithms. Inspired by this, we design a DCIM macro with customized 6T-SRAM-based cells. Fig. 2 shows the overall architecture, including 16 DCIM cores and an MRCM. Each core has 32 arrays, a DPSU, and an EACS. The database (original and hash), weights (EQ format), and pre-computed KV cache (frequent documents) are stored off-chip. The detailed workflow is described below.

A. DPSU for Retrieval Phase

Fig. 3 shows DPSU for memory-intensive *Retrieval* phase, saving TTFT by dual-step EMA reduction of the distant DVs. At step 1 (vector-level), distant DVs are rapidly excluded by hash method. All DVs are projected offline to low-dimensional hash codes (HC_{DV}) and stored in DCIM. For example, DV_i is hashed to a 16-dimensional binary vector HC_{DVi}. The prompt vector (PV) is projected to HC_{PV} and fed to DCIM's wordlines for bit-wise comparison using CAM-mode. Match signals are NORed with 0 and then added up by the adder tree to get the Hamming distance between HC_{PV} and HC_{DVi}. DVs with the top-M smallest Hamming distances ($M>m$) are selected as candidates. At step 2 (bit-level), PV is stored in DCIM and the original DVs of the M candidates are fetched bit-serially from off-chip for precise similarity computation. Early termination is triggered if the per-cycle-predicted upper-bound is lower than current top-m similarity. DV is divided into positive group (PG) and negative group (NG) by the sign of partial products (PP). Upper-bound of similarity is reached when non-fetched

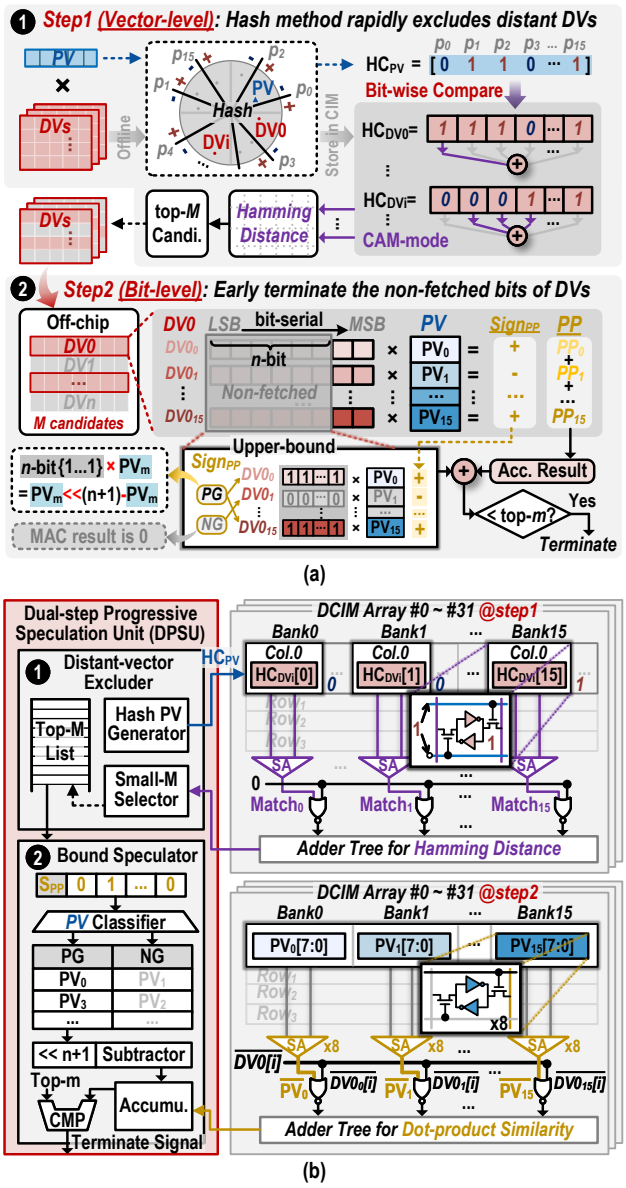


Fig. 3. DPSU for Retrieval phase. (a) Algorithm of dual-step progressive speculation. (b) Hardware of dual-step progressive speculation unit.

n-bit DVs of PG are all-1 and of NG are all-0. The n-bit all-1 value multiplied by PV equals $\{PV \ll (n+1)\}$-PV, facilitating efficient speculation via shift and subtraction. DPSU reduces latency by 6.69× and energy by 5.82× over *Retrieval* in [6].

B. MRCM for Prefill Phase

Fig. 4 shows MRCM for compute-intensive *Prefill* phase, saving TTFT by maximizing KV cache reuse. Initially, the KV cache of frequently retrieved documents (e.g., D_2, D_4, D_5) is precomputed by self-document-attention (SDA) and kept in DRAM. When a document sequence $\{D_4, D_2, D_3\}$ arrives, it is searched in DRAM and the matched KV cache (of D_4 and

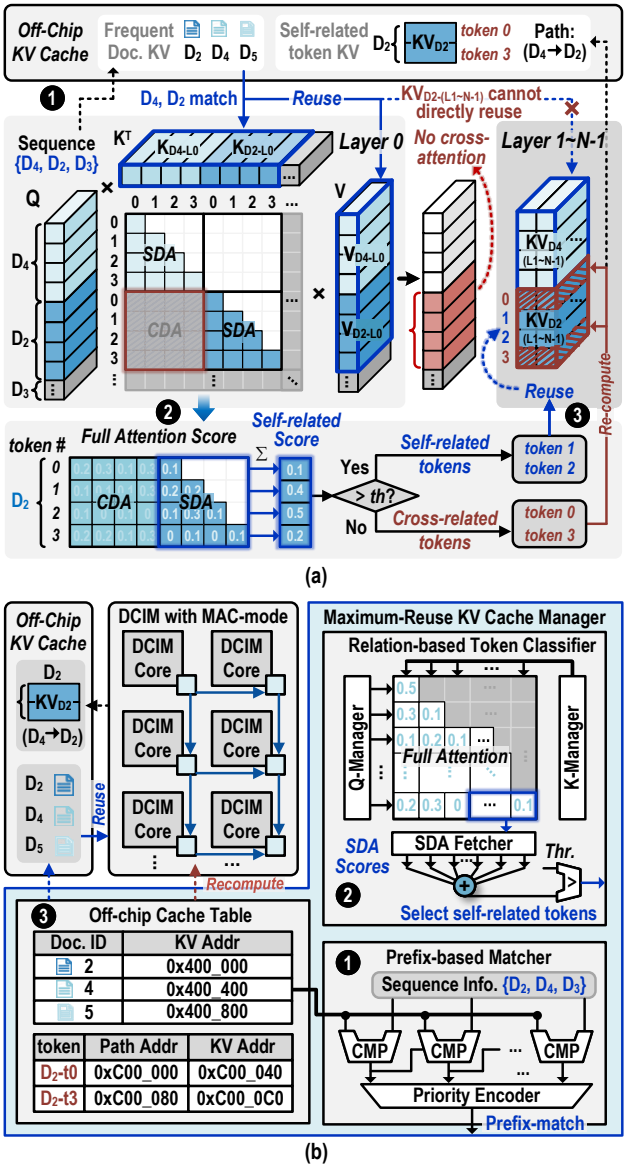


Fig. 4. MRCM for Prefill phase. (a) Algorithm of maximum-reuse KV cache management. (b) Hardware of maximum-reuse KV cache manager.

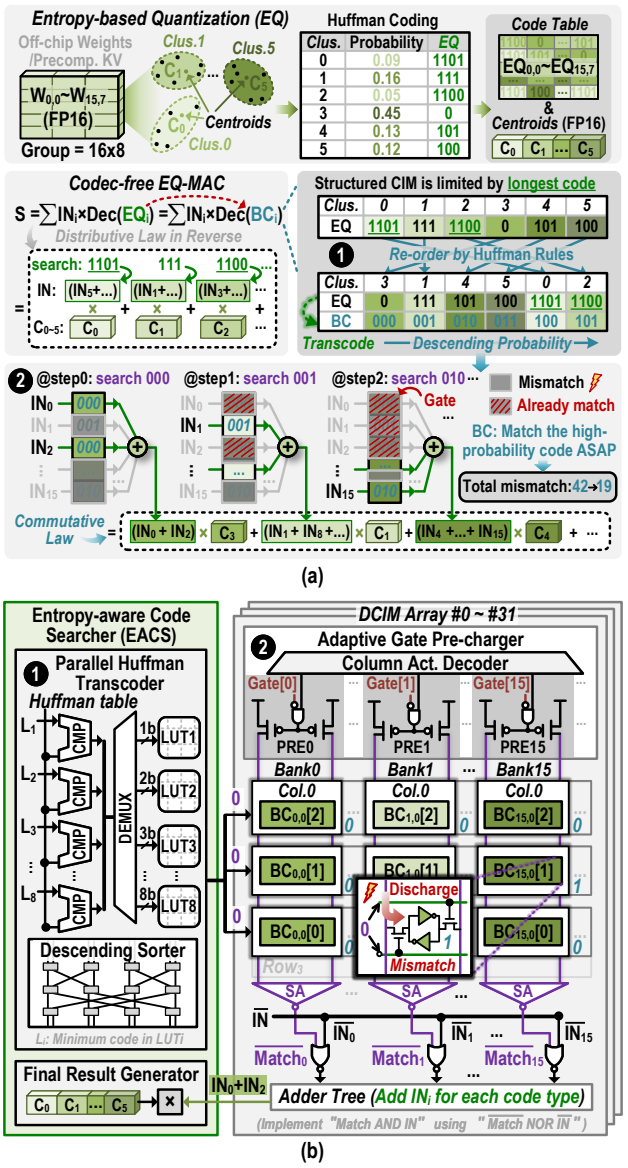


Fig. 5. EACS for Decoding phase. (a) Algorithm of entropy-based quantization and codec-free EQ-MAC. (b) Hardware of entropy-aware code searcher.

D_2) is fetched onto the chip. The KV cache of $layer_0$ can be directly reused, yet for subsequent layers, D_2's KV cache needs recomputation due to lack of cross-document-attention (CDA). D_2's full-attention in $layer_0$ is computed to get score matrix. SDA scores of D_2 are row-wisely accumulated, and tokens whose sums exceed a threshold are identified as self-related tokens (e.g., token1, 2). Self-related means strong correlation within its own document, thus the corresponding KV cache in subsequent layers can be reused with negligible loss [7]. The rest cross-related tokens (e.g., token0, 3) have more correlation with other documents, whose KV need recomputation together with the unmatched D_3. The recomputed KV of D_2's cross-related tokens are kept in DRAM with a path tag (D_4-to-D_2), which can be reused for future prefix-matched sequences {D_4, D_2, D_x}. MRCM reduces 4.35× latency and 3.51× energy.

C. EACS for Decoding Phase

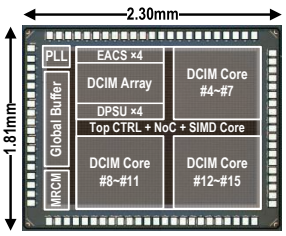
Fig. 5 shows EACS for memory-intensive *Decoding* phase, saving TPOT by loading data with low-bitwidth EQ. Weights and KV are partitioned into k-means clusters represented by centroids, and then assigned Huffman codes according to their probabilities. In the example, 16×8 FP16 data are quantized into a code table and 6 centroids. Rather than dequantizing codes and storing centroids, DCIM directly performs codec-free computation by CAM-mode. Specifically, all inputs corresponding to each code are accumulated separately via code

search and then multiplied by the centroid. Although Huffman coding minimizes total bit usage, low-probability clusters incur longer code lengths, lowering DCIM's utilization. Note that codes are only cluster identifiers and do not affect final results. Thus, EQ is first reordered by Huffman rules (i.e., shorter-first with 1-leading tie-break) to place high-probability codes earlier, and then transcoded into 3-bit balanced-code (BC) for efficient DCIM storage. Codes are searched in DCIM for input accumulation. For example, code 000 is searched and all inputs are ANDed with their corresponding match signal then added up by the adder tree. As a result, all inputs corresponding to cluster 3 are accumulated (e.g., IN_0+IN_2). Many energy-costly mismatches can be gated as matched codes no longer need to be searched. With BC in probability-descending order, high-probability codes will be matched in earlier steps, minimizing total mismatches. Compared to NUQ-based accelerators [3]–[5], EACS reduces latency by 31.5% and energy by 28.1%.

III. Measurement Results and Conclusion

Fig. 6 shows the fabricated DRAGON chip that operates at 0.6-1.0V, 125-500MHz. Unlike previous LLM accelerators [1]–[4] that focus on high efficiency at low frequency, we aim to improve performance for edge deployment. For various-sized RAG systems, it reduces TTFT by 5.08∼5.81× and TPOT by 5.98∼6.45×, reaching 5.12ms-response (response time per token) and 42.83 token/s at 1.0V, 500MHz. Compared with the state-of-the-art LLM/agent accelerators in Table I, our work reduces TTFT/TPOT by 7.28×/1.76× over [1] and 2.84×/6.07× over [6]. Thanks to the all-phase optimization, it achieves 4.97×/1.42× speedup over DCIM accelerator [5].

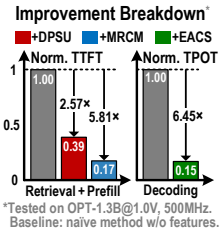
In conclusion, we propose a DCIM-based RAG accelerator with speedup across all phases. By utilizing DCIM's reconfigurable MAC/CAM-mode for compute/memory-intensive phases, we reduce TTFT/TPOT for practical edge deployment.



Technology	TSMC 28nm CMOS
Die Area	4.16 mm²
Supply Voltage	0.6V-1.0V
Frequency	125MHz-500MHz
Capacity	256KB Buffer + 32KB DCIM
Data Precision	A: FP16, W: INT8/EQ
Work Mode	MAC-/CAM-mode
Power	11.90mW @0.6V, 125MHz
	89.08mW @1.0V, 500MHz
TTFT*	5.12 ms-response
1 / TPOT*	42.83 token/s

Model	GPT-2 Large	OPT-1.3B	LLaMA2-7B
Database*1	*Wiki2*-based Collections		
Dataset	Natural Questions (NQ)		
PPL(↓) / Loss*2	28.76 (+0.87)	13.54 (+0.73)	4.38 (+0.16)
TTFT*3*4 (ms-response↓)	2.63	5.12	27.39
1 / TPOT*3*5 (token/s↑)	76.63	42.83	7.82
TTFT Speedup*6	5.08×	5.81×	5.33×
TPOT Speedup*6	5.98×	6.45×	6.12×

*1:Constructed by text-embedding-3-small. *2: Compared with FP16. No perplexity loss over NUQ [3–5].
*3: Highest performance point, 1.0V, 500MHz. *4: "-response" is average response time per input token.
*5: Generation time is averaged due to KV expansion. *6: Baseline: w/o the three proposed techniques.



*Tested on OPT-1.3B@1.0V, 500MHz. Baseline: naïve method w/o features.

Fig. 6. Die photo, summary table, and measurement results.

TABLE I
Comparison with state-of-the-art LLM/Agent accelerators

	ISSCC'25 23.7 [6]	ISSCC'25 23.1 [3]	ISSCC'25 23.8 [1]	VLSI'25 C17-3 [5]	This work
Implementation	Digital Logic	Digital Logic	Digital Logic	Digital CIM	**Digital CIM**
Phase Optimization	Only Retrieval	Only Prefill	Only Decoding	Pref. and Dec.	✓ **All Phases**
Work Mode	MAC	MAC	MAC	MAC/CAM	✓ **MAC/CAM**
Precision	INT2-8 (A) INT8 (W)	INT4/8/16 (A) INT6/NUQ4(W)	BF16 (A) INT8 (W)	FP16 (A) INT8/NUQ(W)	✓ **FP16 (A) INT8/EQ (W)**
Technology	28nm	16nm	28nm	28nm	**28nm**
Die Area (mm²)	20.25	10.15	3.52	12.65	**4.16**
Frequency (MHz)	50-200	60-450	50-460	110-500	**125-500**
Supply Voltage (V)	0.7-1.1	0.45-0.85	0.63-1.0	0.6-1.0	**0.6-1.0**
Power (mW)	52.4-559.2	7.12-152.5	16.95-105.0	56.74-335.04	**11.90-89.08**
Peak Performance (TOPS or TFLOPS)	0.41-1.64 (INT8)	0.81-2.15 (INT8)	1.40-7.16*3 (Norm. INT8)	0.24-5.96*3 (Norm. INT8)	**0.51-3.28***1*4
Energy Efficiency (TOPS/W or TFLOPS/W)	2.93-7.82 (INT8)	15.2-40.3 (INT8)	20.14-74.25*3 (Norm. INT8)	1.70-31.2*3 (Norm. INT8)	**10.13-95.34***2*4
TTFT*5*7(ms-response↓)	14.56 (2.84×)*6	21.33 (4.17×)*6	37.27 (7.28×)*6	25.45 (4.97×)*6	**5.12***1
1 / TPOT*5*8(token/s↑)	7.06 (6.07×)*6	7.39 (5.80×)*6	24.39 (1.76×)*6	30.21 (1.42×)*6	**42.83***1

*1: At the highest performance point, 1.0V, 500MHz. *2: At the highest efficiency point, 0.65V, 180MHz.
*3: For fairness, max-report in [1,5] is normalized to INT8. *4: Low: w/o all features; High: w/ three features.
*5: Evaluated on OPT-1.3B. Input token = 1024, output token = 512. Top@3 is retrieved. DDR3 is included.
*6: Unsupported phases (e.g., Retrieval in [1,3,5], Decoding in [3,6]) are supplemented with naïve methods.
*7: "-response" means average response time per input token. *8: Average token generation throughput.

Acknowledgements

This work was funded in part by National Key R&D Prog. 2023YFB4403100; Beijing S&T Proj. Z251100008425010; NSFC Grant 62422407, 62125403, 92464302, U24B20164.

References

[1] Y. Qin *et al.*, "An 88.36tops/w bit-level-weight-compressed large-language-model accelerator with cluster-aligned int-fp-gemm and bi-dimensional workflow reformulation," in *2025 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 68, 2025, pp. 420–422.

[2] S. Kim *et al.*, "Slim-Llama: A 4.69mw large-language-model processor with binary/ternary weights for billion-parameter llama model," in *2025 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 68, 2025, pp. 421–423.

[3] S. Moon *et al.*, "T-REX: A 68-to-567μs/token 0.41-to-3.95μJ/token transformer accelerator with reduced external memory access and enhanced hardware utilization in 16nm finfet," in *2025 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 68, 2025, pp. 406–408.

[4] S. Um *et al.*, "Dial: An energy-efficient dram in-memory computing accelerator with compact partial product lut and twisted differential adc," in *Symposium on VLSI Technology and Circuits (VLSI)*, 2025, pp. 1–3.

[5] Z. Wu *et al.*, "CELLA: A 28nm compute-memory co-optimized real-time digital cim-based edge llm accelerator with 1.78ms-response in prefill and 31.32token/s in decoding," in *2025 Symposium on VLSI Technology and Circuits (VLSI)*, 2025, pp. 1–3.

[6] W. Jo *et al.*, "BROCA: A 52.4-to-559.2mw mobile social agent system-on-chip with adaptive bit-truncate unit and acoustic-cluster bit grouping," in *2025 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 68, 2025, pp. 418–420.

[7] J. Yao *et al.*, "CacheBlend: Fast large language model serving for RAG with cached knowledge fusion," in *Proceedings of the European Conference on Computer Systems (EuroSys)*, 2025, pp. 94–109.

[8] F. Tu *et al.*, "MulTCIM: A 28nm 2.24μJ/token attention-token-bit hybrid sparse dcim accelerator for multimodal transformers," in *International Solid-State Circuits Conference (ISSCC)*, 2023, pp. 248–250.

[9] S. Liu *et al.*, "A 28nm 53.8tops/w 8b sparse transformer accelerator with in-memory butterfly zero skipper for unstructured-pruned nn and cim-based local-attention-reusable engine," in *2023 IEEE International Solid-State Circuits Conference (ISSCC)*, 2023, pp. 250–252.

[10] F. Cheng *et al.*, "Ecco: Improving memory bandwidth and capacity for LLMs via entropy-aware cache compression," in *Proceedings of the 52nd Annual International Symposium on Computer Architecture (ISCA)*, New York, NY, USA, 2025, pp. 793–807.