SwiftCIM: A 55nm 23.2μJ/Token L-0.5 ReRAM Coupled Digital CIM Accelerator with Fully-Fused Multi-Head Attention Dataflow for FlashAttention

***Abstract*—This paper presents SwiftCIM, a novel Transformer accelerator with three key features: (1) A high-density ReRAM coupled SRAM cell: Each SRAM-CIM cell integrates a level-0.5 (L-0.5) high-density ReRAM subarray, enabling fast and robust data loading via differential sensing. (2) L-0.5 ReRAM coupled digital CIM (RC-DCIM): RC-DCIM features dual loading paths - one leveraging the high-bandwidth, low-latency weight loading from L-0.5 ReRAM for rapid projection, and the other configured for conventional external loading for attention - enabling flexible reuse and efficient dataflow scheduling. (3) HW-SW co-designed fully-fused MHA (FF-MHA) dataflow: Leveraging RC-DCIM's rapid projection, we reformulate the FlashAttention-2 dataflow, recompute intermediate tiles, and fuse operators to enable vector-wise pipelining, effectively reducing intermediate tile storage and movement overhead with limited recomputation overhead. Fabricated in 55nm CMOS with commercial ReRAM, SwiftCIM integrates 8MB L-0.5 ReRAM in 29.16mm², achieving a CIM storage density of 3.95Mb/mm² and an end-to-end energy efficiency of 23.22μJ/token, improving energy efficiency by 1.75× and throughput by 2.5× over the baseline.**

I. Introduction

Efficient acceleration of multi-head self-attention (MHA) is critical for deploying Transformers on edge devices. While FlashAttention-2 (FA-2) [1] avoids full attention matrix materialization through tiled computation, storing and transferring QKV and intermediate tiles still incurs significant external memory access (EMA) and on-chip data movement overhead that scale with sequence length. In CNN accelerators, layer fusion [2] pipelines consecutive layers to eliminate intermediate feature map storage by exploiting spatial locality. However, online softmax for a score tile in MHA requires the complete tile row vector ready before normalization, imposing a strict data dependency that prevents vector-wise pipeline fusion.

Computing-in-Memory (CIM) accelerates Transformers by performing computations directly on stored weight matrices. However, fabrication design rule limitations and the large area overhead of digital circuits prevent existing SRAM-based digital CIM technologies from storing all weights within the CIM array, leading to frequent weight loading from external memory or SRAM buffers during MHA computation. To address these, ROM-CIM [3] employs read-only, high-density 1T ROM at crossbar nodes to store multi-bit weights, enabling multiple weight tiles within a single macro but the weights cannot be modified. Additionally, [4] reduces EMA overhead and improves energy efficiency on a digital CIM accelerator via tiled FlashAttention, yet its reliance on KV cache introduces EMA overhead that scales linearly with context length.

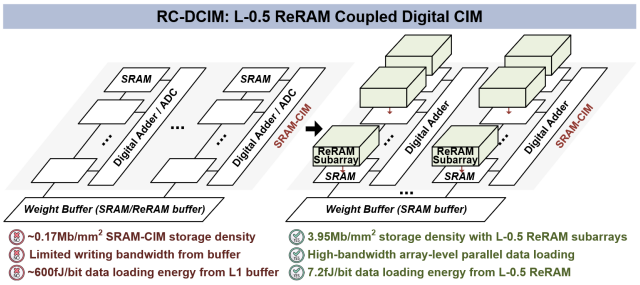

Fig. 1: Comparison between Conventional SRAM-based Digital CIM [5] and our RC-DCIM with L-0.5 ReRAM.

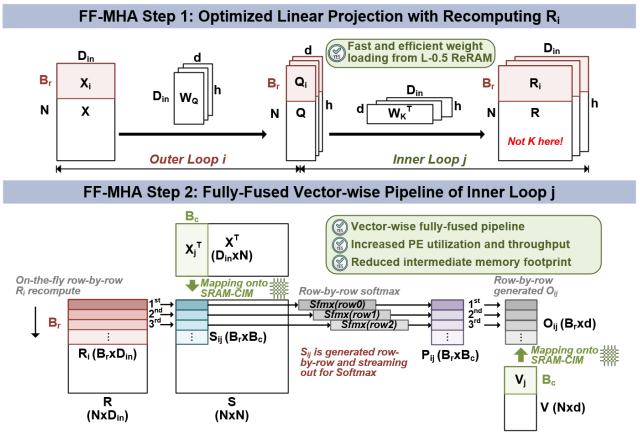

Fig. 2: Our HW-SW co-designed fully-fused multi-head attention dataflow for Flash-Attention2.

As shown in Figure 1 and 2, we propose SwiftCIM, an level-0.5 ReRAM coupled digital CIM accelerator for efficient FlashAttention-2 computation, with three key features:

- First, to improve storage density, we design an L-0.5 ReRAM coupled SRAM memory cell integrating a high-density 64-bit SLC ReRAM subarray with 1-bit SRAM. L-0.5 denotes that the ReRAM does not directly participate in computation but is tightly coupled to the SRAM-CIM cell. Robust data readout is achieved via differential sensing by SRAM itself between the storage ReRAMs and reference ReRAMs. The high-density L-0.5 ReRAM enables a storage density of 3.95Mb/mm² in CIM macro.
- Second, based on the ReRAM coupled cell, we design an L-0.5 ReRAM coupled digital CIM (RC-DCIM) architecture to enhance support for different MHA operators. RC-DCIM features dual loading paths: one leveraging the 1cycle/tile and 7.2fJ/bit weight loading from L-0.5

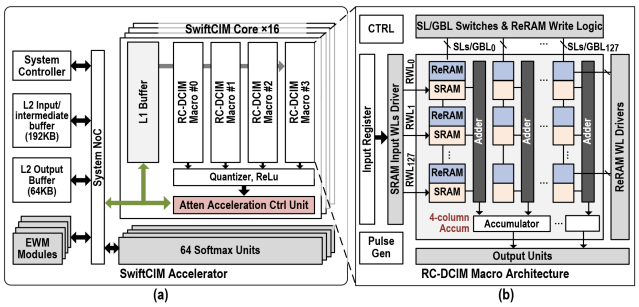

Fig. 3: Overall Architecture of SwiftCIM and RC-DCIM Macro.

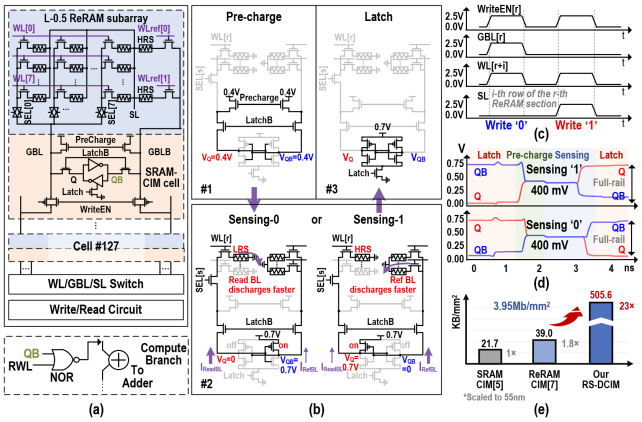

Fig. 4: ReRAM Coupled SRAM Cell with Differential Sensing.

ReRAM for rapid projection, and the other configured for conventional external loading for attention to enable flexible reuse and efficient dataflow scheduling.

- Finally, leveraging RC-DCIM's rapid projection, we design a HW-SW co-designed fully-fused MHA (FF-MHA) dataflow. As shown in Figure 2, by reformulating S_{ij}=$Q_iK_j^T$ as S_{ij}=$R_iX_j^T$ with R_i=$Q_iW_K^T$, recomputing R_i and V_j, and fusing operators to enable vector-wise pipelining, we eliminate KV off-chip loading and on-chip intermediate tile memory access. FF-MHA reduces EMA by 82%, intermediate tile memory access by 67%, on-chip storage by 36%, and improves SRAM-CIM utilization from 35% to 84%, thereby improving energy efficiency to 1.75× and throughput to 2.5×. Although R_i and V_j recomputation is introduced, the latency is hidden by pipeline, and the energy overhead is lower than that of EMA and intermediate tile movement in the baseline.

II. SwiftCIM Architecture and Dataflow

Figure 3(a) illustrates the overall SwiftCIM architecture and RC-DCIM macro. SwiftCIM comprises 16 cores, 256KB L2 buffer, a system network-on-chip (NoC), 64 row-wise softmax units, element-wise multiplier (EWM) modules and a system controller. Each core includes four RC-DCIM macros, an 84KB L1 buffer, an attention acceleration control unit for fine-grained dataflow control, and quantizer and ReLU modules.

As shown in Figure 3(b), in the 128×128 RC-DCIM macro, each crossbar node integrates a 64-bit ReRAM subarray with 1-bit SRAM, providing 1Mb ReRAM storage. Offline, data is written into ReRAM via the ReRAM write logic; at runtime, data is loaded to SRAM via differential sensing. Inputs enter

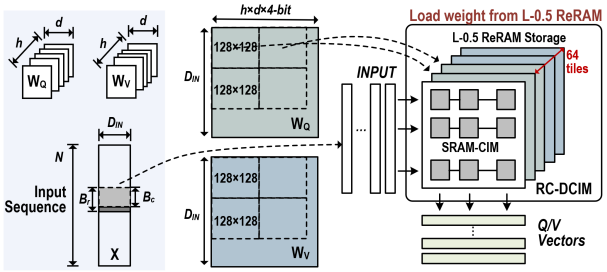

Fig. 5: Multiple Weight Tile Mapping in RC-DCIM Macro.

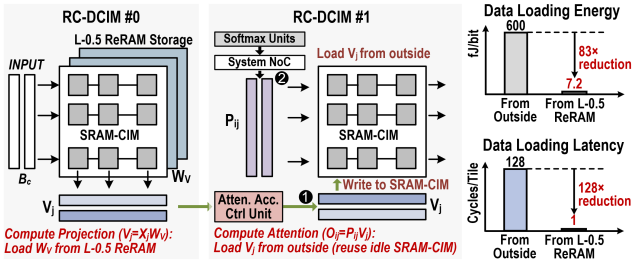

Fig. 6: RC-DCIM Dual Loading Path for Projection and Attention.

the SRAM-CIM bit-serially, and column-wise adders are for accumulation. For INT4, four column adder outputs are further accumulated over four cycles to produce the final result. The SRAM supports dual loading paths: conventional writing via the global bitline (GBL), or loading from ReRAM via differential sensing. Each macro integrates an asynchronous pulse generator that produces the necessary precharge and latch signals for ReRAM-to-SRAM data loading.

A. L-0.5 ReRAM coupled SRAM cell

Figure 4 shows the circuit schematic, differential sensing mechanism, and write/sensing waveforms of the L-0.5 ReRAM coupled SRAM cell. L-0.5 ReRAM is not directly involved in computation but serves as a high-density 'cache' for the SRAM-CIM. The subarray consists of an 8×8 1T1R SLC array with 8 WLs, 8 BLs, and a shared sourceline (SL). WLs activate the selected row, select signals (SELs) control transmission gates for BL selection, and SL provides VDD/GND for write and sensing. Two reference ReRAMs in high-resistance state (HRS) are connected in parallel to provide an intermediate reference resistance. The 7T SRAM cell includes a latch transistor controlling the latch loop. Its left side connects to the ReRAM array through Global BL (GBL), while the right side connects to reference ReRAMs through GBLB. A precharge circuit precharges Q and QB nodes. The cell connects to RC-DCIM column GBL and GBLB through two transmission gates enabled by EN_{write}.

During offline writing, a specific ReRAM cell is selected via WL, SEL, SL, and EN_{write}, and programmed at 2.5V. As shown in Figure 4(c), applying voltages in different polarities programs different resistance states. During computation, the SRAM's differential latch loop enables differential sensing. ① The Latch signal is set to 0 to break the feedback loop. ② The Precharge signal precharges both Q and QB nodes to 400mV, an intermediate voltage. ③ WL and SEL select a specific row and column, WLref is activated, and SL is

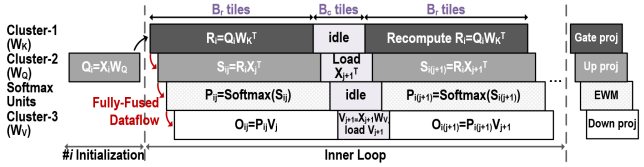

Fig. 7: Illustrations of our Fully-Fused MHA dataflow.

set to GND. This creates two discharge paths: **the left path** from Q to SL through the selected ReRAM, and **the right path** from QB to SL through the reference ReRAMs. Due to resistance differences, these paths discharge at different rates. If the selected ReRAM is in LRS (lower resistance than reference), the left path discharges faster, pulling Q to GND and turning on the PMOS of QB, charging it to VDD. When Latch is re-enabled, the SRAM latches to '0'. Conversely, if in HRS, QB discharges faster and the SRAM latches to '1'. This scheme enhances read reliability and speed.

Our tested cell has a nominal HRS of 60kΩ and LRS of 5kΩ without sensing errors, which is more reliable for LLMs compared to prior MLC ReRAM simulation work [6]. The L-0.5 ReRAM achieves a read latency of 2.7ns and a CIM macro storage density of 3.95Mb/mm², 23× higher than high-density SRAM-CIM [5] and 13× higher than ReRAM-CIM [7].

B. Proposed L-0.5 ReRAM Coupled Digital CIM (RC-DCIM)

As shown in Figure 3(b), the RC-DCIM macro consists of a 128×128 cell array, where each cell incorporates an independent read path. A pulse generator coordinates all cells within the macro to perform differential sensing in parallel, enabling a complete ReRAM-to-SRAM-CIM weight tile loading in a single cycle - 128× faster than row-by-row SRAM writing - and achieving a chip-level peak ReRAM-to-SRAM-CIM bandwidth of 29.8TB/s at 250MHz. Compared to loading from outside, ReRAM loading in RC-DCIM consumes only 7.2fJ/bit, approximately 1/83 of the outside loading energy.

As shown in Figure 5, weight matrices are partitioned into tiles and stored in L-0.5 ReRAM. SwiftCIM supports INT1-8 precision, and in this work we adopt INT4 for aggressive quantization evaluation in edge deployment. Given the 64:1 ReRAM-to-SRAM ratio, each macro can store 64 INT4 weight tiles and switch between tiles in a single cycle, significantly enhancing local weight storage and alleviating weight loading overhead. This key advantage of L-0.5 ReRAM enables the **rapid projection support** required by the FF-MHA dataflow.

As illustrated in Figure 6, RC-DCIM features dual loading paths. Beyond rapid projection support, it can also operate as a conventional SRAM-CIM loading data from GBL to support attention computation, avoiding resource underutilization. The SRAM-CIM in RC-DCIM supports two computation tasks from Figure 2: S_{ij}=$R_i X_j^T$ and O_{ij}=$P_{ij}V_j$, where X_j^T and V_j are respectively loaded from outside into the SRAM-CIM and participate in computation under a weight-stationary dataflow.

C. HW-SW co-designed fully-fused MHA (FF-MHA) dataflow

In FA-2 inference on edge devices, storing QKV off-chip for MHA incurs substantial EMA overhead. While recomputing K_j^T in the inner loop avoids EMA, the overhead on

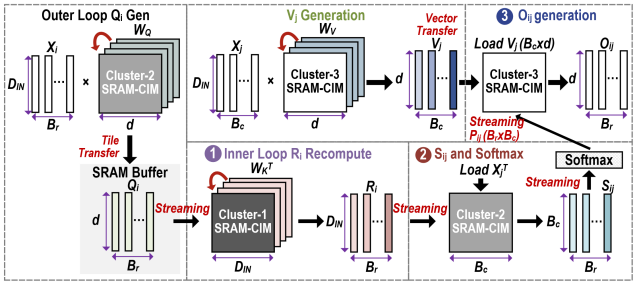

Fig. 8: Hardware Allocation in FF-MHA Dataflow.

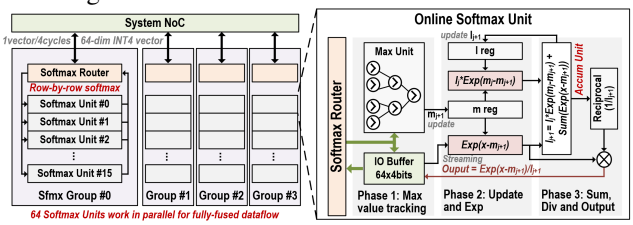

Fig. 9: Softmax Group and the Unit Design.

traditional hardware largely offsets the savings. Furthermore, the strict data dependency imposed by softmax between $Q_i K_j^T$ and V_j requires intermediate tiles to be buffered in SRAM, introducing additional on-chip movement and storage overhead. Leveraging RC-DCIM's rapid projection, we propose the FF-MHA dataflow illustrated in Figure 2, which reformulates this data dependency via row-by-row recomputation of R_i=$Q_i W_K^T$.

Figure 7 illustrates the FF-MHA pipeline. RC-DCIM is partitioned into three cluster types. In the outer loop initialization, rapid projection computes Q_i, stored in the buffer to launch the inner loop. In the j-th inner loop, Cluster-1 recomputes R_i=$Q_i W_K^T$ row-by-row along B_r, streaming resulting vectors directly to Cluster-2 for S_{ij}=$R_i W_K^T$ without SRAM buffering. Each row of S_{ij} immediately triggers row-wise softmax, and the resulting P_{ij} vectors stream to Cluster-3 for O_{ij}=$P_{ij}V_j$. All intermediate R_i, S_{ij}, P_{ij} and V_j tiles are transmitted vector-wise through the pipeline, eliminating tile-level buffering and hiding recomputation latency. Since $B_r \gg B_c$, hardware utilization remains consistently high. Without rapid projection, weight loading for recomputation would stall the pipeline, precluding effective dataflow fusion.

Figure 8 shows the FF-MHA hardware allocation: only Q_i and O_{ij} require SRAM buffering. To maximize utilization, clusters with idle weights are repurposed for attention computation: Cluster-2 (W_Q) handles S_{ij}, and Cluster-3 (W_V) handles O_{ij}. As shown in Figure 9, softmax units are organized into 4 parallel groups of 16 units with a circular pipeline, each requiring 64 cycles per computation. Since one S_{ij} vector is produced every 4 cycles, the router assigns tasks at the same rate, forming a circular pipeline with no idle slots. Each unit implements FA-2's online softmax via three phases: max tracking, update and exponentiation, and sum, division and output, while maintaining m and l values to rescale O_{ij} tiles.

III. Validations and Conclusions

Figure 10 presents the die photo, chip specifications, shmoo plot, and ReRAM TEM photo of our 55nm test chip, which has an area of 29.16mm². Measurement results

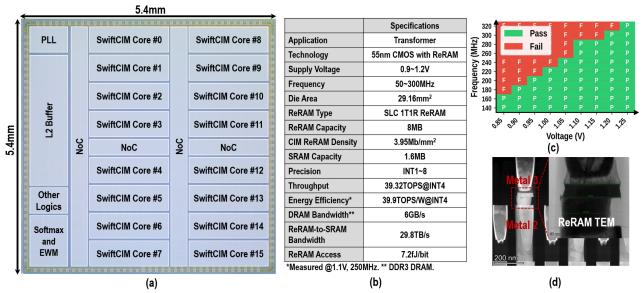

Fig. 10: Die Photo, Chip Spec, Shmoo Plot and ReRAM TEM.

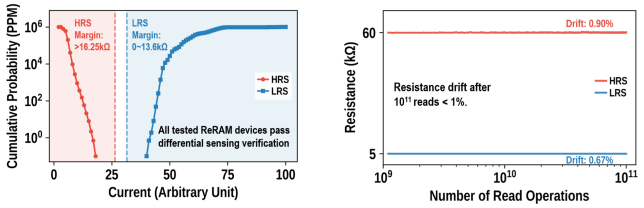

Fig. 11: ReRAM Sensing Margin and Read Disturbance Test.

show that the chip operates at 50-300MHz with a supply voltage of 0.85-1.25V. The chip achieves a typical INT4 throughput of 39.32TOPS@1.2V and an energy efficiency of 39.9TOPS/W@1.1V.

As shown in Figure 11, we analyze the differential sensing margin under CMOS and ReRAM resistance variation to ensure sensing reliability. In the worst case, the margin is $>$16.25kΩ for HRS and 0$\sim$13.6kΩ for LRS, within which ReRAM bits are correctly distinguished. Measurements on commercial ReRAM devices confirm that all tested HRS and LRS cells fall within the margin. ReRAM resistance remains stable with drift less than 1% after 10^{11} consecutive reads.

As shown in Figure 12, we conduct end-to-end experiments on INT4 quantized BERT-Base and MobileLLM. The baseline follows the FlashAttention scheduling strategy similar to [4], with specific adaptations for fair comparison: QKV stored off-chip and loaded on-chip for computation, weights stored in a conventional L-1 ReRAM buffer of equal capacity, SRAM-CIM of identical size and throughput, and the same B_r and B_c as SwiftCIM. The baseline does not employ the FF-MHA dataflow. End-to-end evaluations show that SwiftCIM achieves 2.5$\times$ to 3.6$\times$ speedup and 1.75$\times$ to 3.03$\times$ energy efficiency improvement over the baseline.

Figure 12(c) reveals the key mechanism: although recomputation increases GeMM overhead, the energy cost remains limited owing to L-0.5 ReRAM's low-latency, low-power weight loading. By enabling the FF-MHA dataflow, recomputation reduces EMA and on-chip SRAM access by 82% and 67%, respectively, significantly lowering overall energy overhead. Furthermore, RC-DCIM's rapid projection and flexible attention support enable efficient pipeline fusion, improving SRAM-CIM utilization from 35% to 84%, hiding recomputation latency, and reducing on-chip SRAM footprint by 36%, collectively yielding significant performance gains.

As shown in Figure 13, we compare SwiftCIM with state-of-the-art works [3], [5], [7]–[9]. The results show that RC-DCIM achieves a CIM storage density of 3.95Mb/mm², benefiting

Workload Specifications

Model	Layers	#Head (h)	Head Dim (d)	Embedding Dim (D_{IN})	Sequence Length (N)	Param	Row Tile (B_r)	Column Tile (B_c)	Layers/Chip
BERT-Base	12	12Q/12KV	64	768	512	81M	512	64	2
MobileLLM	30	9Q/3KV	64	576	2/4/8K	101.25M	1024	64	4

End-to-End Evaluation Results

SwiftCIM Performance	BERT-Base (N=512)	MobileLLM (N=2K)	MobileLLM (N=4K)	MobileLLM (N=8K)
Time-To-First-Token (ms)	7.64	17.53	42.97	117.76
Energy (mJ)	11.89	33.0	82.88	225.78
Energy/Token (μJ)	23.22	16.11	20.23	27.56

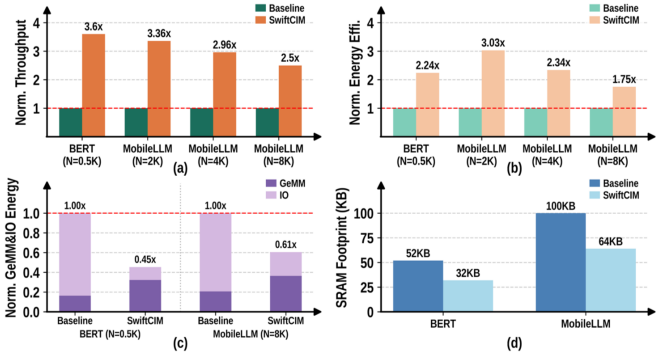

Fig. 12: End-to-End Evaluations and Validations.

Comparison with State-of-the-art Accelerators

	High-Density Macro Baselines			System Baselines		This work
	ESSCIRC'23 [5]	JSSC'23 [3]	ISSCC'24 [7]	ISSCC'23 [8]	JSSC'25 [9]	
Applications	CNN	CNN	CNN	LLM	LLM	LLM
Technology	16nm	65nm	22nm	12nm	28nm	55nm
Die Area (mm²)	0.5	12	8.2	4.6	7.098	29.16
Implementation	SRAM Digital CIM	ROM Analog CIM	ReRAM Analog CIM	Digital	SRAM Analog CIM	ReRAM Coupled SRAM Digital CIM
Voltage (V)	0.58-1.0	0.7-1.2	0.7-0.8	0.62-1.0	0.56-0.9	0.9-1.2
Frequency	20-91MHz	50-210MHz	200MHz	77-717MHz	80-275MHz	50-300MHz
Precisions	INT8	INT1-8	FP16/BF16	FP4, FP8	INT8	INT1-8
Throughput (TOPS/TFLOPS)	0.182@INT8	0.0546@INT8	0.86@BF16	0.734@FP4 0.367@FP8	1.38@INT8	39.32@INT4
SRAM Buffer Cap. (KB)	N/A	N/A	N/A	647	166	1600
NVM/CIM Cap. (KB)	128 (SRAM)	256 (ROM)	2048 (ReRAM)	N/A	80	8192 (ReRAM)
CIM Mem. Den. (1) (Mb/mm²)	0.17	5.43 (Read-only)	0.31	N/A	0.50	3.95
CIM Energy Effi. (1) (TOPS/W or TFLOP/W)	6.92@INT8	5.12@INT8	65.5@BF16	N/A	14.7@INT8	39.9@INT4(2)
System Efficiency (μJ/Token)	N/A	N/A	N/A	75.19 (BERT-Base)	22.87(3) (BERT-Base)	23.22(4) (BERT-Base)

(1) Scaled to 55nm for fair comparison. (2) Measured at 1.1V, 250MHz for best reliability of L-0.5 ReRAM differential sensing.
(3) Off-chip memory access is excluded. (4) DDR3 is included, and W4A4KV4 is adopted for best performance.

Fig. 13: Comparison with SOTA Works.

from the tight integration of the L-0.5 ReRAM. Further more, enabled by the HW-SW co-designed FF-MHA dataflow, SwiftCIM achieves a significant end-to-end energy efficiency of 23.22μJ/token on INT4 quantized BERT-Base.

REFERENCES

[1] T. Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv*, 2023.

[2] M. Alwani et al. Fused-layer cnn accelerators. In *MICRO*. IEEE, 2016.

[3] G. Yin et al. Cramming more weight data onto compute-in-memory macros for high task-level energy efficiency using custom rom with 3984-kb/mm 2 density in 65-nm cmos. *JSSC*, 2023.

[4] B. Liu et al. A 52.03 tops/w dcim-based accelerator with flashattention and sparsity-aware alignment for llms. In *CICC*. IEEE, 2025.

[5] W. Jiang et al. A 16nm 128kb high-density fully digital in memory compute macro with reverse sram pre-charge achieving 0.36 tops/mm 2, 256kb/mm 2 and 23. 8tops/w. In *ESSCIRC*. IEEE, 2023.

[6] X. Wang et al. Rescim: Variation-resilient high weight-loading bandwidth in-memory computation based on fine-grained hybrid integration of multi-level reram and sram cells. In *ICCAD*, 2024.

[7] T.-H. Wen et al. 34.8 a 22nm 16mb floating-point reram compute-in-memory macro with 31.2 tflops/w for ai edge devices. In *ISSCC*, 2024.

[8] T. Tambe et al. 22.9 a 12nm 18.1 tflops/w sparse transformer processor with entropy-based early exit, mixed-precision predication and fine-grained power management. In *ISSCC*. IEEE, 2023.

[9] R. Guo et al. A 28-nm 28.8-tops/w attention-based nn processor with correlative cim ring architecture and dataflow-reshaped digital-assisted cim array. *JSSC*, 2025.