20.2 A 28nm 74.34TFLOPS/W BF16 Heterogenous CIM-Based Accelerator Exploiting Denoising-Similarity for Diffusion Models

Ruiqi Guo[1], Lei Wang[1], Xiaofeng Chen[1], Hao Sun[1], Zhiheng Yue[1], Yubin Qin[1], Huiming Han[1], Yang Wang[1], Fengbin Tu[2], Shaojun Wei[1], Yang Hu[1], Shouyi Yin[1,3]

[1]Tsinghua University, Beijing, China
[2]Hong Kong University of Science and Technology, Hong Kong, China
[3]Shanghai AI Lab, Shanghai, China

2024 IEEE International Solid-State Circuits Conference (ISSCC) | 979-8-3503-0620-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/ISSCC49657.2024.10454308

Diffusion models (DMs) have emerged as a powerful category of generative models with record-breaking performance in image synthesis [1]. A noisy image created from pure Gaussian random variables needs to be denoised by iterative DMs to ensure generative quality. For DMs, quantizing activations to integers (INT) degrades image quality due to changes in activation distributions and the accumulation of quantization errors across iterations. A GPU (Nvidia A100) requires 2560ms and 250W to generate a 256×256 image through 50 iterations of a floating-point (FP) DM. Two adjacent denoised images bring similar visual effects, where the difference between pixels at the same position is very small. As a result, for two adjacent DMs, most input differences within the same layer (ΔIN) are consistently clustered within a narrow range, indicating that most ΔIN can be quantized as INT-data. The remaining ΔIN have relatively large values, whose distributions vary across iterative DMs. To ensure generative quality, a complete ΔIN tensor is divided into a dense INT tensor (INT-ΔIN) and a sparse FP tensor (FP-ΔIN). Compute-in-memory (CIM) has shown high throughput and energy efficiency on INT multiply-and-accumulate (MAC), demonstrating its potential to process ΔIN efficiently. However, prior CIM chips face three challenges in speeding up on-device image generation to seconds with low power consumption [2-7]. First, conventional CIM chips perform MACs with bit-serial inputs, leading to significant runtime. A recent CIM chip incorporates an additional adder-tree to handle one more input bit, albeit at the cost of 85.4% more power and 82.5% more area [3]. Second, CIM chips cannot process FP data at high speed like INT data. They either requires repeated reads/writes to handle high-precision mantissas [4], or face lengthy alignment-cycle latencies [5]. Third, previous FP CIMs do not support identifying and utilizing stored sparse data, leading to redundant computations.

To solve these challenges, this paper presents a heterogenous CIM chip to process dense INT-ΔIN and sparse FP-ΔIN for accelerating DMs with three features: 1) A Sign-Magnitude radix-8 Booth (SMB) CIM macro is designed to store sign-magnitude (SM) weights and process the 4b octal groups of radix-8 Booth (Booth8) encoded INT-ΔIN. Compared to bit-serial CIM with two's complement (2C) data, SMB-CIM reduces cycle count and bitwise multiplication by 67% and reduces power consumption by 32%. 2) A four-operand exponent CIM (4Op-ECIM) macro is designed to activate four bitcells in each column for two addition operations. The 4Op-ECIM is assisted with mantissa processing engines (ManPEs) to efficiently process FP data without the need for exponent alignment and repeated mantissa access. 3) An in-memory redundancy-search-elimination technique is proposed in the 4Op-ECIM. The 4Op-ECIM is designed to be reconfigurable as a binary content-addressable memory (CAM) for identifying the stored sparse FP-ΔIN. The generated sparse index is used to prevent redundant bitline precharging and sensing in 4Op-ECIM.

Figure 20.2.2 shows the overall architecture of the chip. It consists of an SMB-CIM core, two 4Op-ECIM macros, 16 ManPEs, a sparsity-aware compute-allocator (SACA), a SIMD core, 200KB SRAM and a top controller. The SMB-CIM core processes INT-ΔIN, while 4Op-ECIMs and ManPEs process FP-ΔIN. Similar to video inter-frame processing [8], the chip has to recover diffusion results based on the last diffusion results and convert back to the ΔIN of next layer. The SMB-CIM core includes six SMB-CIM macros and six data encoders. Each macro stores {1, weight mantissa} that pre-aligned by exponent, and receives Booth8 encoded INT-ΔIN. Each 4Op-ECIM macro has two operate-modes. In compute mode, 4Op-ECIM computes two products' exponents (E_P) by summing the exponent of input (E_{IN}) and weight (E_W), and simultaneously computes the E_P difference (ΔE_P). In search mode, 4Op-ECIM is reconfigured as a CAM to search zero-valued E_{IN}. The SACA compares the sparsity of E_{IN} in adjacent rows of 4Op-ECIM and allocates computations involving sparser E_{IN} to the energy-efficient datapath of 4Op-ECIM. ManPEs perform mantissa multiplications of FP-ΔIN and weights, and shift products by ΔE_P for FP32 accumulation. The results are sent to the SIMD core for complex operations within DMs like non-linear activation and differential equation solution.

Figure 20.2.3 shows the SMB-CIM macro that includes 48 MAC arrays and 16 accumulators. Each MAC array comprises 64 input decoders, 64 weight banks (each bank has four rows and four columns), 64 shifters, 64 demultiplexers (Demuxs), two 64-input unsigned adder trees (one for positive and the other for negative additions only), and a 12b subtractor. By leveraging the data representations, the SMB-CIM handles more input bits with lower signal toggles in the dual unsigned adder trees (DUAT). The SMB-CIM stores SM weights. The generated products are accumulated in the DUAT based on their respective signs. Then, two unsigned results are combined by the subtractor. The SM-based MAC operation significantly reduces the toggle activity since weights typically have normal distributions with high occurrence of small values. Furthermore, the DUAT provides opportunities to process Booth8 encoded inputs in a bit-parallel scheme. A Booth8 input is represented by a series 4b octal groups. The set of products of octal groups is {0, ±W, ±2W, ±3W, ±4W}. Each product can be obtained by a subtraction between two left-shifted W. For example, –3W=(W<<0)–(W<<2). Thus, the SMB-CIM can perform the MAC of 4b octal groups in a cycle. Each decoder receives CIM inputs and weights' signs, decoding them as shift-amount signals for shifters and data-forwarding signals for Demuxs. A row of data in a SMB-CIM bank are appropriately shifted and forwarded into DUAT. Compared to conventional bit-serial CIMs with 2C data, the SMB-CIM achieves 2.8-to-3.3× speedup and 1.04-1.47× power savings under different data distributions.

Figure 20.2.4 details the 4Op-ECIM that accelerates exponent processing. In ECIM, 4 bitcells in a column are activated to compute two E_P and their difference (ΔE_P), achieving the balanced latency with SMB-CIMs. 16 ManPEs are integrated to assist a 4Op-ECIM macro without pipeline bubbles. Each 4Op-ECIM macro includes 8×128 sub-arrays and 128 near-memory exponent aligners (NMEAs). A sub-array includes a column of 32 8T-cells. A column of the ECIM is connected to a read bitline (RBL) and a pair of bitlines (BL/BLB). A row of the ECIM is controlled by three separate word-lines (WLL, WLR, RWL). In each 4Op-ECIM, the WLL/WLR of two rows and RWL of two rows are activated simultaneously. Three read ports (BL, BLB, RBL) are fully utilized to perform Boolean logic on four operands (A, B, C, D). BL represents the OR/NOR of A and B with SA0. BLB represents the AND/NAND of A and B with SA1. By setting two appropriate reference voltages, RBL represents the OR/NOR of C and D with SA2, and the AND/NAND of C and D with SA3. Every two operands act as one bit of E_{IN} and one bit of E_W, respectively. All Boolean results are sent to the NMEA to compute: 1) two E_P; 2) the ΔE_P; 3) the comparison with local maximum E_P. The NMEA consists of two 20T full-swing full-adders (FSFAs), a full-adder (FA), a comparator and registers. Due to four Boolean results already obtained for 1b E_{IN} and 1b E_W, E_P is computed bit-serially in the 20T-FSFA instead of a 28T-FSFA. Opposite sums of two 20T-FSFAs are used to compute ΔE_P bit-serially in a simple FA. The 4Op-ECIM achieves 2.0-to-2.3× speedup and 24.5% energy saving compared to [6].

Figure 20.2.5 shows the in-memory redundancy-search-elimination technique. The FP-ΔIN with zero-valued E_{IN} (sparse E_{IN}) is approximated to zero, incurring a zero product. Thus, the computation involving sparse E_{IN} is redundant. 4Op-ECIM can find this redundancy by working in search mode, using only WLR and RWL. To search sparse E_{IN}, the activated WLR and RWL are set to 0 and 1, respectively. Taking 2b E_{IN} as an example, the WLR0/WLR1=0/0, and conversely, the RWL0/RWL1=1/1. The RBL and BLB stay at the precharged value with a sparse E_{IN}. Conversely, the RBL/BL discharges, and the AND of SAs is 0, thus indicating a sparse E_{IN}. The ECIM compares 128 E_{IN} with zero in a cycle, achieving 8× speedup and 4.64× energy reduction. The generated sparse index is stored to optimize the ECIM working in compute mode. When a sparse E_{IN} is computed through BL/BLB, the connected SA0 and SA1 are powered off by the 1b sparsity index. When a sparse E_{IN} is computed through RBL, the precharge of RBL is bypassed, and the two connected SAs are powered off. Therefore, mapping computations involving more zero-valued E_{IN} onto RBL saves more power consumption. The SACA compares the E_{IN} sparsity in two adjacent rows of 4Op-ECIM, allocating the sparser row of E_{IN} to be computed through RBL. By eliminating redundant bit-line precharging and SA sensing, 4Op-ECIM achieves 1.56× power reduction.

Figure 20.2.6 shows the measurement results of the fabricated 28nm CIM processor. The chip works at 50-540MHz with 0.6-to-1.0V supply. Experiments are conducted on 50 iterative DMs. By leveraging the denoising similarity, the original full FP computations are handled by dense INT-ΔIN and sparse FP-ΔIN, which reduces 3.12× energy consumption and 1.82× execution time. In addition, the chip obtains 3.23× speedup on the entire DM since the SMB-CIM and 4Op-ECIM achieves 3.3× and 2.1× speedup for INT-ΔIN and FP-ΔIN, respectively. Moreover, the chip achieves 4.0× energy savings as a consequence of reducing signal toggles by the SMB-CIM and utilizing sparsity by the 4Op-ECIM. The peak system energy efficiency is 67.89TFLOPS/W for FP16 and 74.34TFLOPS/W for BF16 at 0.65V, 120MHz, which is 4.02× and 2.55× higher than state-of-the-art FP CIM chips [5, 7]. Figure 20.2.7 shows the summary and die photo.

Acknowledgement:
This work was supported in part by NSFC Grant 62125403, NSFC Grant 92164301, NSFC Grant U19B2041, Beijing Municipal Science and Technology Project Grant Z221100007722023, Special funding for industrial infrastructure reengineering and high-quality development of manufacturing industry 2022 (CEIEC-2022-ZM02-0245), the BNRist, and the Beijing Advanced Innovation Center for Integrated Circuits. The corresponding author of this paper is Shouyi Yin (yinsy@tsinghua.edu.cn).

References:

[1] J. Ho et al., "Denoising Diffusion Probabilistic Models," *NeurIPS*, 2020.
[2] X. Si et al., "A 28nm 64-kb 31.6-TFLOPS/W Digital-Domain Floating-Point-Computing-Unit and Double-Bit 6T-SRAM Computing-in-Memory Macro for Floating-Point CNNs," *ISSCC*, pp. 128-129, 2023.

Authorized licensed use limited to: University of Texas at Austin. Downloaded on September 18,2025 at 19:35:11 UTC from IEEE Xplore. Restrictions apply.

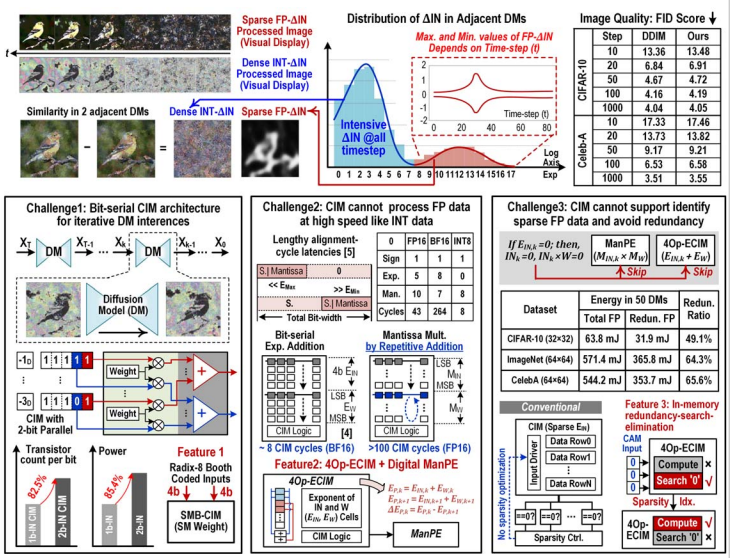


Figure 20.2.1: Challenges of designing a CIM processor to leverage the similarity of iterative diffusion models.

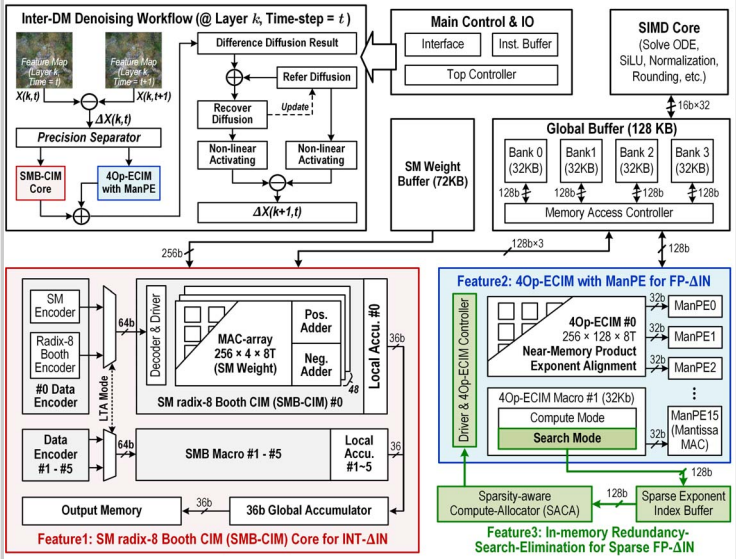


Figure 20.2.2: The overall architecture of the proposed CIM processor for diffusion acceleration.

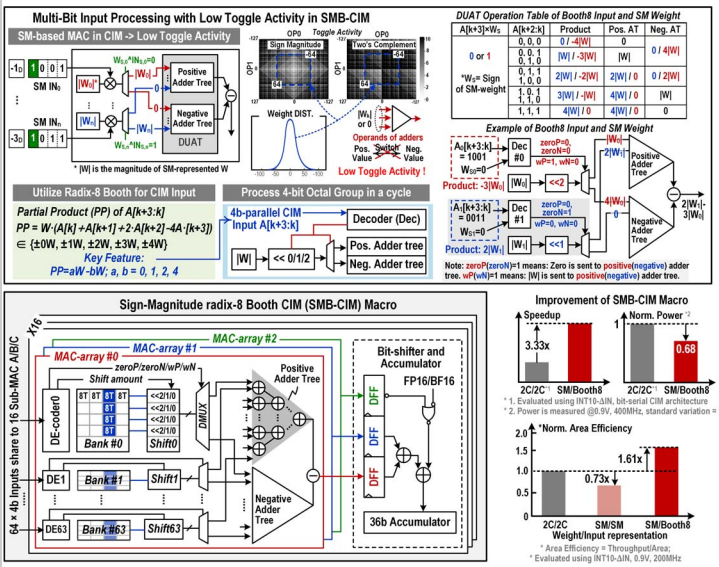


Figure 20.2.3: A Sign-Magnitude radix-8 Booth (SMB) CIM macro that performs low-toggle-activity MAC operations with 4b input data.

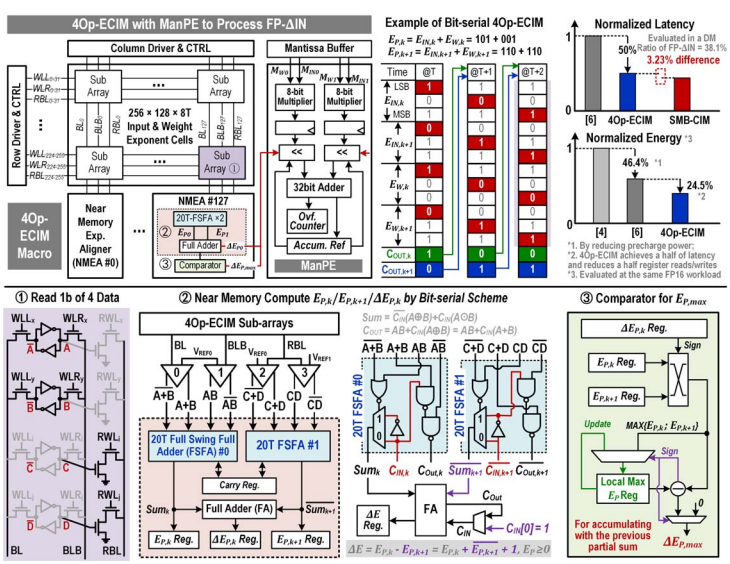


Figure 20.2.4: A four-operand exponent CIM (4Op-ECIM) macro with an auxiliary mantissa processing engine (ManPE) to speed up FP computation.

Figure 20.2.5: An in-memory redundancy-search-elimination technique that optimizes the processing of sparse FP-ΔIN.

Figure 20.2.6: Measurement results and performance comparison table.

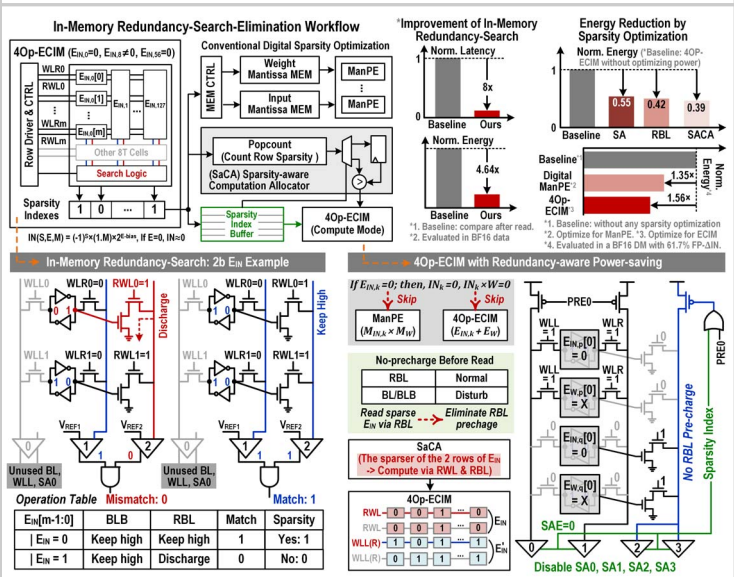
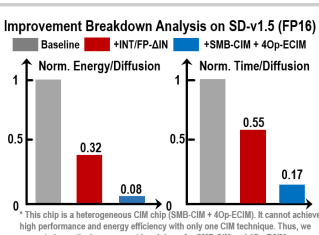


Figure 20.2.7: Measurement results and comparison table.

Measurement Results on Stable Diffusion v1.5 (SD-v1.5)[*5]

Model Precision	FP16	FP16	FP16
Dataset	ImageNet (64×64)	CIFAR-10 (32×32)	CelebA (64×64)
Baseline FID (50 Steps)	20.73	7.02	7.33
Chip FID (50 Steps)	20.78	7.04	7.39
Execution Time (s)[*3]	18.380	4.656	18.378
Overall Macro Energy[*4] Efficiency (TFLOPS/W)	71.98 (SMB-CIM)	67.14 (SMB-CIM)	73.60 (SMB-CIM)
System Energy[*4] Efficiency (TFLOPS/W)	65.73	58.01	67.89

1. Off-chip memory access is excluded; 2. FP16/BF16 Model with INT10-ΔIN, FP16/BF16-ΔIN; 3. @540MHz, 1.0V; 4. @120MHz, 0.65V. 5. Measured the Unet module of SD-v1.5.



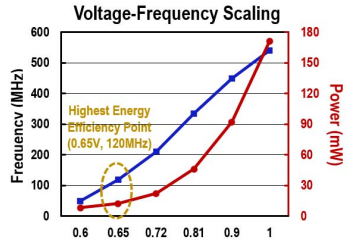
Comparison with State-of-the-art FP CIM Processor/Macro

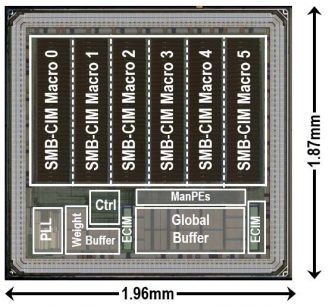
	ISSCC 23 [5]	ISSCC 22 [7]	VLSI 21 [6]	ISSCC 23 [1]	This work
Technology (nm)	28	28	28	28	**28**
Task	DNN Training	Cloud DNN Inference	DNN Training	DNN Inference	**Diffusion Model**
Data Precision	INT4/8, FP16/BF16	INT8/16, BF16, FP32	BF16	BF16	**INT10/16, FP16/BF16**
Chip Area (mm²)	4.54	6.69	5.8	0.146 (Macro)	**3.67**
Supply Voltage (V)	0.469 – 0.9	0.6 – 1.0	0.76 – 1.1	0.6 – 0.9	**0.6 – 1.0**
Frequency (MHz)	10 – 400	50 – 220	~250	81 – 182	**50 – 540**
CIM Size	16kb	96kb	1280kb	64kb	**288kb SMB-CIM + 64kb 4Op-ECIM**
CIM Function	FP-MAC	FP-MAC	Element-wise FP-Multiply/Add	FP-MAC	**SMB-CIM: INT-MAC; ECIM: Element-wise FP-Add**
CIM Area	0.269	No Reported	No Reported	0.146	**SMB-CIM: 0.225; ECIM: 0.02625**
CIM Macro Energy Efficiency (TFLOPS/W)	17.2 – 91.3 (FP16) (Block-wise Sparsity)	23.2 (BF16) 3.0 (FP32)	13.7 (BF16)	14.04 – 31.6 (50% Input Sparsity)	**SMB-CIM: 80.3 (FP16)** @0.65V, 120MHz
System Power (mW)	0.87 – 74.9	12.5 – 69.4	1.2 – 156.1	No Reported	**8.268 – 171.0**
System Peak Performance (TFLOPS)	No Reported for FP precision	1.08 (BF16) 0.14 (FP32)	0.119 (BF16)	N/A	**6.636 (BF16)** @1.0V, 540MHz **4.424 (FP16)** @1.0V, 540MHz
System Peak Energy Efficiency (TFLOPS/W)	16.9 (FP16)	29.2 (BF16) 3.7 (FP32)	1.43 (BF16)	N/A	**74.34 (BF16)** @0.65V, 120MHz **67.89 (FP16)** @0.65V, 120MHz
Chip Area Efficiency (TFLOPS/mm²)	No Reported	0.160 (BF16) 0.021 (FP32)	0.021 (BF16)	2.05 (BF16) (Only macro)	**1.808 (BF16)** @1.0V, 540MHz **1.205 (FP16)** @1.0V, 540MHz

Figure 20.2.8: Measurement results and performance comparison table.

Authorized licensed use limited to: University of Texas at Austin. Downloaded on September 18,2025 at 19:35:11 UTC from IEEE Xplore. Restrictions apply.

	Specifications
Technology	28nm CMOS
Chip Area	1.87mm×1.96mm
CIM Macro Area	SMB-CIM: 0.90mm×0.25mm ×6 4Op-ECIM: 0.36mm×0.075mm ×2
Precision	FP16/BF16 Weight, Input Hybrid INT10-ΔIN, FP16-ΔIN
Voltage	0.6V – 1.0V
Frequency	50MHz – 540MHz
CIM Size	288kb SMB-CIM + 64kb 4Op-ECIM
SRAM Size	200 KB
Chip Power	8.268 mW @ 0.6V, 50MHz 171.0 mW @ 1.0V, 540MHz
System Peak Performance *1	6.636 TFLOPS (BF16) 4.424 TFLOPS (FP16)
SMB-CIM Macro *2 Energy Efficiency	80.3 TFLOPS/W (FP16)
System Energy Efficiency *2	74.34 TFLOPS/W (BF16) 67.89 TFLOPS/W (FP16)
Area Efficiency *2	1.808 TFLOPS/mm^2 (BF16) 1.205 TFLOPS/mm^2 (FP16)
Energy of *3 50 Iterative DMs	230.88 mJ (BF16) 252.81 mJ (FP16)





1. Measured at the highest performance point, 1.0V, 540MHz. 2: Measured at the highest energy efficiency, 0.65V, 120MHz.
3. Only Include the core module of the diffusion mode, e.g. U-Net model. The computations of auto-encoder and decoder are excluded.

Figure 20.2.7: Chip micrograph and performance summary.

Additional References:

[3] C. Lee et al., "A 12nm 121-TOPS/W 41.6-TOPS/mm2 All Digital Full Precision SRAM-based Compute-in-Memory with Configurable Bit-width For AI Edge Applications," *IEEE Symp. VLSI Circuits*, pp. 24-25, 2022.

[4] J. Wang et al., "A Compute SRAM with Bit-Serial Integer/Floating-Point Operations for Programmable In-Memory Vector Acceleration," *ISSCC*, pp. 224-225, 2019.

[5] J. Yue et al., "A 28nm 16.9-300TOPS/W Computing-in-Memory Processor Supporting Floating-Point NN Inference/Training with Intensive-CIM Sparse-Digital Architecture," *ISSCC*, pp. 252-253, 2023.

[6] J. Lee et al., "A 13.7 TFLOPS/W Floating-point DNN Processor using Heterogeneous Computing Architecture with Exponent-Computing-in-Memory," *IEEE Symp. VLSI Circuits*, 2021.

[7] F. Tu et al., "A 28nm 29.2TFLOPS/W BF16 and 36.5TOPS/W INT8 Reconfigurable Digital CIM Processor with Unified FP/INT Pipeline and Bitwise In-Memory Booth Multiplication for Cloud Deep Learning Acceleration," *ISSCC*, pp. 254-255, 2022.

[8] Z. Yuan et al., "A 65nm 24.7μJ/Frame 12.3mW Activation-Similarity-Aware Convolutional Neural Network Video Processor Using Hybrid Precision, Inter-Frame Data Reuse and Mixed-Bit-Width Difference-Frame Data Codec," *ISSCC*, pp. 232-233, 2020.

Authorized licensed use limited to: University of Texas at Austin. Downloaded on September 18,2025 at 19:35:11 UTC from IEEE Xplore. Restrictions apply.

979-8-3503-0620-0/24/$31.00 ©2024 IEEE