23.1 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

Seunghyun Moon[1], Mao Li[1], Gregory K. Chen[2], Phil C. Knag[2], Ram Kumar Krishnamurthy[2], Mingoo Seok[1]

[1]Columbia University, New York, NY
[2]Intel, Hillsboro, OR

Transformer, a recent mainstream model in deep learning, has revolutionized a wide range of AI applications, which motivates a surge in research to develop energy-efficient hardware accelerators. Most prior efforts have concentrated on enhancing on-chip computational energy efficiency through several strategies such as encoder-only models [1-7], quantization/sparsity [8-18], and layer pruning [19]. However, recent works [20,21] show that external memory access (EMA) dominates total energy consumption. Our analysis based on [22,23] also indicates that EMA accounts for up to 81% of the total energy usage (Fig. 23.1.1). Additionally, we recognize that the prior works exhibit low hardware utilization, as low as 9% in [4], which negatively impacts latency performance.

In light of this, we present a novel transformer accelerator named T-REX to address the challenges of EMA and hardware utilization. To reduce EMA, based on [24], we developed a factorizing training model that decomposes each weight matrix into a dense matrix shared across all layers (W_S) and a highly sparse matrix distinct to each layer (W_D). During runtime, T-REX needs to preload W_S only once, significantly reducing EMA. To further scale down EMA, we compress W_S and W_D using several advanced compression techniques. Next, we propose a dynamic batching technique, where T-REX monitors input lengths and, if the input is 2× (4×) smaller than the maximum input length of T-REX, it processes 2 (4) inputs simultaneously by reconfiguring its dataflow. This approach reduces EMA by minimizing the number of parameter loads and also enhances hardware utilization. Finally, we developed two-direction accessible register files (TRFs) within the computing cores to load and store a matrix in both row-by-row (R-R) and column-by-column (C-C) fashions. They eliminate the latency overhead caused by accessing SRAMs multiple times, additionally enhancing hardware utilization. Combining the proposed techniques, we prototyped the T-REX test chip in 16nm FinFET. Measurement results show that T-REX can reduce EMA by 31-65.9× and improve hardware utilization by 1.2-3.4× across four well-known transformer workloads [25-28]. It achieves 68-567μs/token and 0.41-3.95μJ/token, including EMA.

Figure 23.1.2 shows the microarchitecture of T-REX, designed for energy-efficient and low-latency inference with factorized and compressed transformer models. It consists of an IO interface, a RISC-V core-based top controller, a DMA, a global buffer (GB), four dense matrix-multiplication (DMM) cores, four sparse matrix-multiplication (SMM) cores, and two auxiliary function units (AFUs). The GB stores compressed W_S, compressed W_D for one layer, and intermediate data. Each DMM core includes a lookup table (LUT)-based non-uniform dequantizer, input and output buffers, an accumulator, and 4×4 processing elements (PEs). Each PE contains 4×4 multiply-and-accumulate (MAC) units, each of which has a 4b multiplier and a 32b accumulator and performs a 16b (8b, 4b) MAC operation over 16 (4, 1) cycles. Each PE is implemented to perform a 4×4 outer product, allowing DMM cores with 4×4 PEs to generate 16×16 output elements simultaneously and compute tiled matrix multiplication (MM) with a tile size of 16×16. On the other hand, each SMM core consists of a uniform dequantizer, input and output buffers, a sparse line buffer, a bias buffer, an accumulator, and 8×8 MAC units. The MAC units are identical to those in the DMM cores. The SMM cores can be configured to perform row (column) products depending on which input matrix is sparse. Non-zero elements (NZs) in the sparse matrix are loaded into the line buffer, while the corresponding rows (columns) are loaded into the input buffer. This sparsity-aware switching feature enables more efficient computation. Finally, the AFU includes input and output buffers, two LUTs for the exponential and GELU functions, 64 integer arithmetic units (IAUs), 16 floating-point arithmetic units (FAUs), BF16↔INT32 converters. The AFUs perform softmax, layer normalization, GELU, and residual connection. For example, in the softmax, the AFU utilizes the LUT for the exponential function and then uses the IAUs to evaluate the remaining computations. Depending on the transformer model, the converters and the FPUs can be used for higher accuracy requirements.

Figure 23.1.3 shows our training model, which replaces a weight matrix W with the product of two submatrices: $W=W_S \cdot W_D$. During runtime, W_S is loaded only once, which substantially reduces EMA. Additionally, W_D is trained to be sparse by adding a regularization term to the loss function, ensuring that each column contains a fixed number of NZs. As W_D becomes highly sparse, we store only the indices and values of the NZs. This compressed format, although similar to compressed sparse column format, does not require storing the column pointer, enabling additional EMA reduction. The proposed training model reduces EMA by 8.5-10.7× across four transformer workloads. The main operation of T-REX is sequential MM, $(X \cdot W_S) \cdot W_D$, where X is the input matrix. We choose this computing order over $X \cdot (W_S \cdot W_D)$ because the hidden size of W_S, is much smaller than that of $W_S \cdot W_D$, reducing the total number of MACs. Furthermore, even compared to $X \cdot W$, the chosen computation requires 1-2.14× fewer MAC operations across the tested models.

To further reduce EMA, we apply 16b-to-4b non-uniform quantization to W_S, reducing the size of W_S by 4× with negligible accuracy loss. We also apply 8b-to-5b delta encoding (*i.e.*, storing the difference of two consecutive values) to indices of W_D. Smaller delta values allow us to use narrower bitwidth, improving the compression ratio. To minimize the delta values without changing $W_S \cdot W_D$, we rearranged the columns of W_S and the corresponding rows of W_D. We also apply 16b-to-6b uniform quantization to values of W_D. To improve the compression ratio, we normalize each value of W_D with a layer-specific scale (M-m) and offset (m), making the distribution symmetric around zero and maximizing the available range and precision of the uniform quantization. The proposed compression techniques enable an additional EMA reduction of 2.1-2.9× across the target models.

Figure 23.1.3 bottom illustrates the hardware support for the main computations in T-REX. The DMM cores handle the first part of the main computation, *i.e.*, $X \cdot W_S$. The input data and W_S are loaded, and the LUT-based non-uniform dequantizer decompresses the 4b non-uniformly quantized W_S to 16b integers, followed by MM within the PEs. For the encoder and decoder layers, as well as for the attention and feed-forward layers, we define separate W_S and maintain independent quantized values. The LUT is reconfigured to accommodate these different quantization settings. Next, the SMM cores perform the second MM, *i.e.*, $(X \cdot W_S) \cdot W_D$. To load the input, delta-encoded indices are used for addressing. Instead of explicit decoding, we use relative addressing to load the corresponding columns of the input matrix. For values of W_D, the uniform dequantizer restores the 6b values of W_D back to 16b using the stored scale and offset. The MAC units then perform the MM, considering only NZs.

We developed a dynamic batching technique to further reduce EMA and improve hardware utilization. T-REX supports the maximum input length of 128. If the input length is between 128 and 65, we configure the dataflow to take one input and produce one output (Fig. 23.1.4 top left). On the other hand, if the input length is between 64 and 33 (32 or less), as shown in Fig. 23.1.4 top right (bottom left) we reconfigure the dataflow to process two (four) inputs simultaneously by specifying which submatrices the DMM/SMM cores use, and which blocks are utilized inside the AFUs. Note that data movement between computing blocks occurs via memory operations, rather than through dedicated buses. Therefore, it incurs <0.1% area overhead to support the dataflow reconfiguration. The proposed dynamic batching technique is particularly effective when the model processes many inputs with short lengths, such as in BERT-Large. It reduces EMA by allowing T-REX to reuse parameters across multiple inputs and improves hardware utilization by up to 3.31×, leading to reduced latency.

Figure 23.1.5 shows a complexity associated with MMs where matrices need to be accessed in different directions. In DMMs using an outer product, X (W_S) needs to be loaded C-C (R-R), and the result Y needs to be stored C-C for the subsequent column product in SMMs. The SMM output Z also needs to be stored in the appropriate direction depending on the next operation; here, it is assumed to be stored R-R. However, if all buffers allow only R-R access as in the conventional memory architecture, it results in wasted clock cycles due to the significant number of SRAM accesses. To address this, we implemented TRFs as the input and output buffers, which contain square-shaped submatrices and allow data access in both row and column directions. These TRF-based buffers eliminate the waste of SRAM access that would otherwise cause all PEs to be idle, thereby improving hardware utilization by 12-20%.

We prototyped the T-REX test chip in 16nm FinFET with a total area of 10.15mm^2 (Fig. 23.1.7). The measurement results show that T-REX operates at 60-450MHz across 0.45 to 0.85V, consuming 7.12 to 152.5mW. Figure 23.1.6 shows the four transformer models that we trained. The proposed training and compression techniques reduce the parameter size by 15.9-25.5× with minimal accuracy loss. When performing inference with these models, T-REX requires 31 to 65.9× less EMA and exhibits 1.2 to 3.4× higher hardware utilization. We compared T-REX with the previous accelerators. For those works that do not consider EMA, we estimated the energy cost at 3.7pJ/b and the latency cost at 6.4GB/s, both based on the LPDDR3 SDRAM [22,23]. T-REX achieves 68 to 567μs/token and 0.41 to 3.95μJ/token, marking significant improvements across several workloads over prior works.

Acknowledgement:
This work was supported in part by an SRC AIHW program (Task 3160.002) and by COGNISENSE, one of seven centers in JUMP 2.0, an SRC program sponsored by DARPA.

2025 IEEE International Solid-State Circuits Conference (ISSCC) | 979-8-3315-4101-9/25/$31.00 ©2025 IEEE | DOI: 10.1109/ISSCC49661.2025.10904793

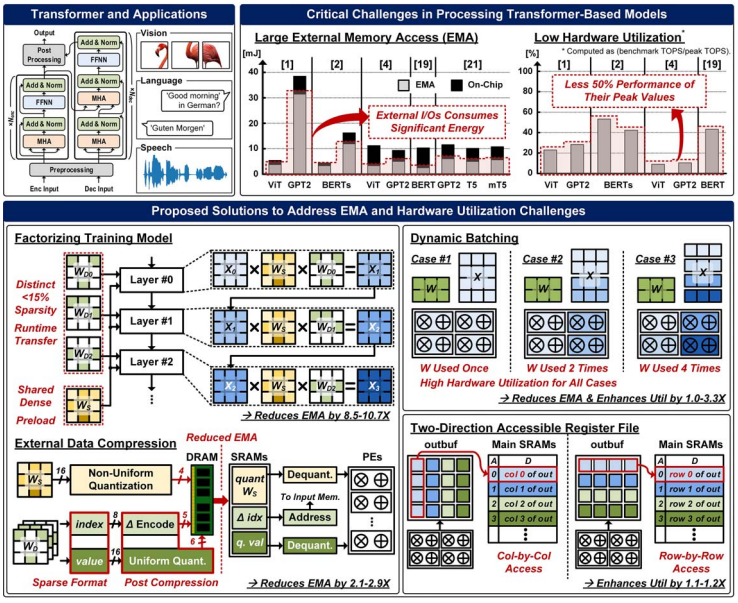


Figure 23.1.1: Challenges in transformer processing and proposed solutions.

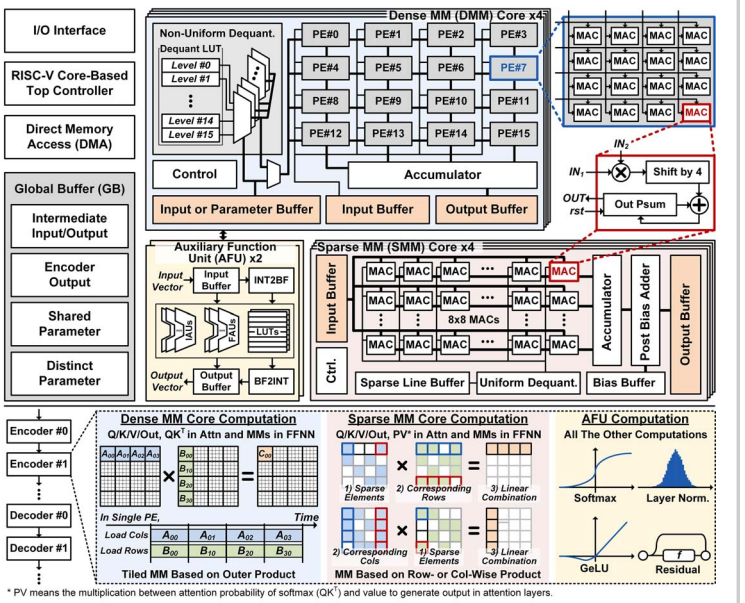


Figure 23.1.2: Overall architecture of T-REX.

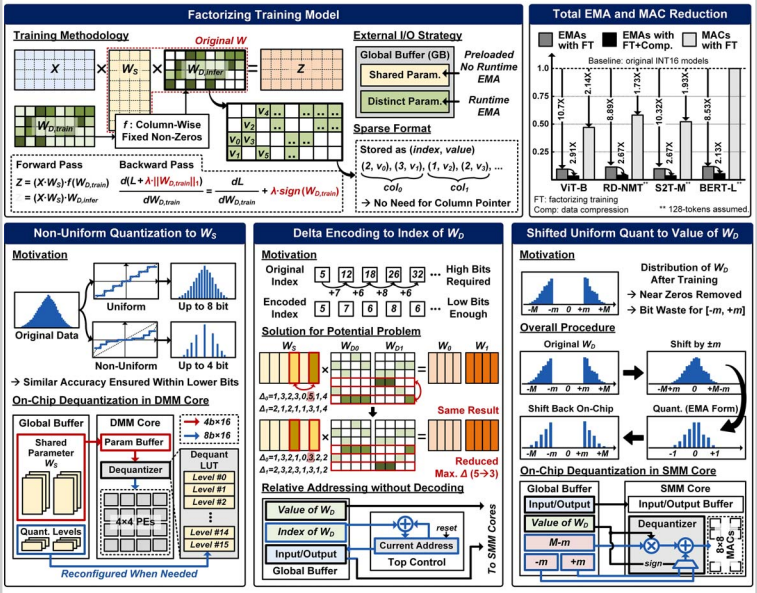


Figure 23.1.3: Factorizing training and compressions with hardware support.

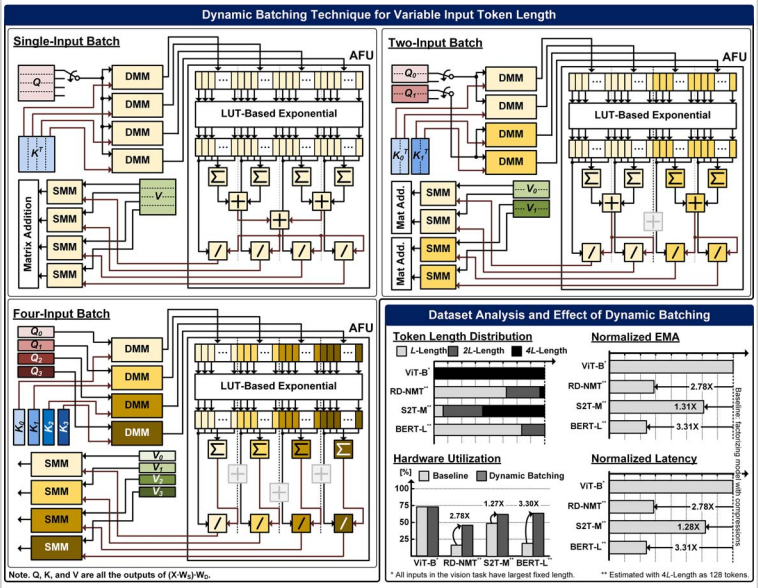


Figure 23.1.4: Dynamic batching technique for variable input token length.

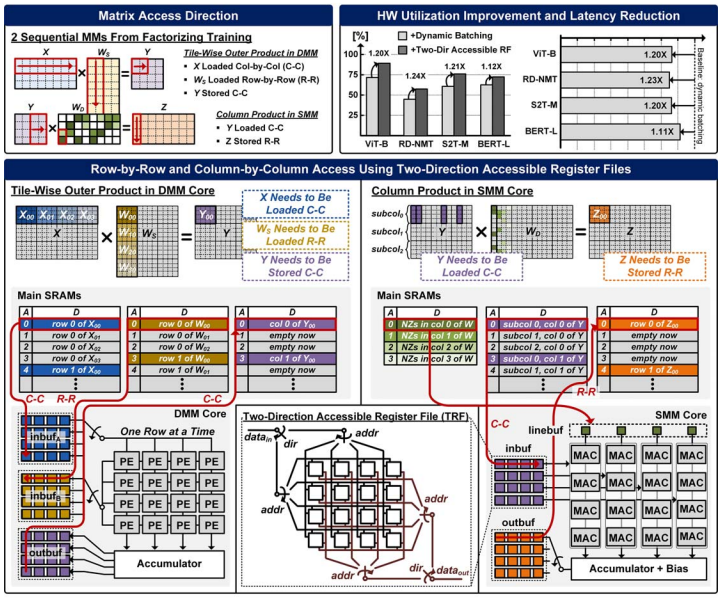
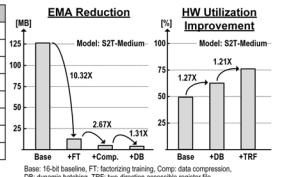


Figure 23.1.5: Input and output buffers based on two-direction accessible register file.

Model	ViT-Base [25]	RD-NMT [26]	S2T-Medium [27]	BERT-Large [28]
Task	Image Classification	Translation	Speech-to-Text	Text Classification
Dataset	ImageNet	IWSLT	LibriSpeech	SST2
Accuracy [1]	Top-1 76.7%	BLEU 29.7	WER 4.1%	GLUE 91.0%
Accuracy Loss	-1.3%p (Ref 78%)	-1.02 (Ref 30.72)	+0.6%p (Ref 3.5%)	-2.1%p (Ref 93.1%)
MM Precision	INT8/16	INT8/16	INT8/16	INT8/16
Parameters [MB] [2]	170 → 6.65	63 → 3.04	126 → 5.16	604 → 37.9
EMA Reduction [2]	31.03×	65.95×	36.18×	60.17×
HW Utilization [%]	73.9 → 89.0	16.7 → 57.4	49.1 → 75.8	19.5 → 72.0

1) Lower is better for WER (word error rate) and higher is better for the others.
2) INT16 assumed as original model precision.

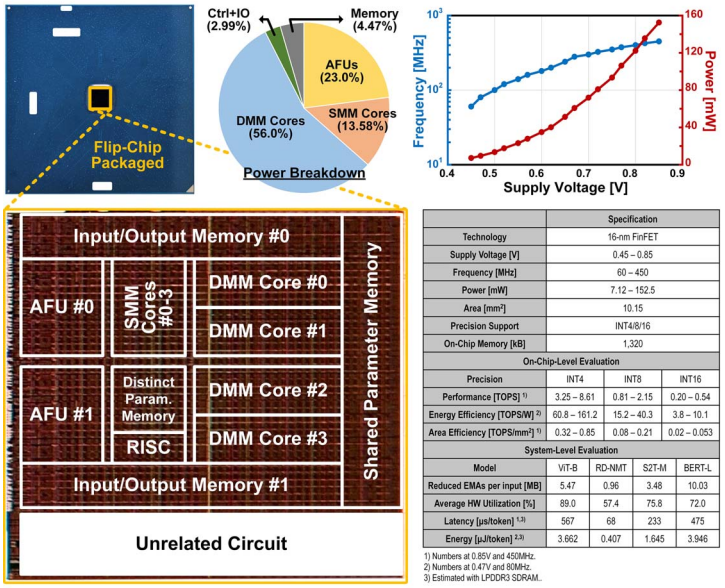


Base: 16-bit baseline, FT: factorizing training, Comp: data compression, DB: dynamic batching, TRF: two-direction accessible register file

	ISSCC'22 [1]	ISSCC'22 [2]	VLSI'23 [4]	ISSCC'23 [19]	ISSCC'24 [21]	This Work
Technology	28-nm CMOS	28-nm CMOS	28-nm CMOS	12-nm FinFET	28-nm CMOS	16-nm FinFET
Supply Voltage [V]	0.56 – 1.1	0.6 – 1.0	0.68 – 1.0	0.62 – 1.0	0.7 – 1.1	0.45 – 0.85
Frequency [MHz]	50 – 510	80 – 240	200 – 580	77 – 717	50 – 200	60 – 450
Power [mW]	12.06 – 272.8	27.04 – 118.21	107 – 391	9 – 122	47.5 – 469.2	7.12 – 152.5
Area [mm²]	6.82	6.83	6.4	4.6	20.25	10.15
Precision Support	INT12	INT8/16	INT8/16	FP4/8	INT8	INT4/8/16
On-Chip Memory [kB]	336	216	480	647	500	1,320
EMA-Excluded On-Chip-Level Comparison						
Performance [1] [TOPS or TFLOPS]	0.52 – 4.07	< 1.48 (INT8) < 0.37 (INT16)	1.18 – 14.82 (INT8) 0.59 – 6.67 (INT16)	< 0.734 (FP4) < 0.367 (FP8)	< 3.41 (INT8)	0.81 – 2.15 (INT8) 0.20 – 0.54 (INT16)
Energy Efficiency [2] [TOPS/W or TFLOPS/W]	4.25 – 27.56	12.5 – 20.5 (INT8) 3.1 – 5.1 (INT16)	3.03 – 77.35 (INT8) 1.51 – 33.7 (INT16)	6.61 – 18.1 (FP4) 3.0 – 8.24 (FP8)	22.9 – 47.8 (INT8)	15.2 – 40.3 (INT8) 3.8 – 10.1 (INT16)
Area Efficiency [1] [TOPS/mm² or TFLOPS/mm²]	0.076 – 0.596	< 0.217 (INT8) < 0.054 (INT16)	0.18 – 2.31 (INT8) 0.092 – 0.36 (INT16)	< 0.16 (FP4) < 0.08 (FP8)	< 0.17 (INT8)	0.08 – 0.21 (INT8) 0.0197 – 0.053 (INT16)
EMA-Included System-Level Comparison						
Benchmark Latency [1] [μs/token]	584 (ViT-Base) [3] 3,707 (GPT2-Large) [3]	411 (BERT-Base) 1,623 (BERT-Large)	384 (ViT-Base) [3]	667 (BERT-Base)	466 (GPT2-Large)	567 (ViT-Base) [3] 68 (RD-NMT) [3] 233 (S2T-Medium) [3] 475 (BERT-Large) [3]
Benchmark Energy [2] [μJ/token]	27.6 (ViT-Base) [3] 92.19 (GPT2-Large) [3]	16.41 (BERT-Base) [3] 58.01 (BERT-Large) [3]	57.02 (ViT-Base) [3]	75.19 (BERT-Base)	18.1 (GPT2-Large)	3.662 (ViT-Base) [3] 0.407 (RD-NMT) [3] 1.645 (S2T-Medium) [3] 3.946 (BERT-Large) [3]

1) Numbers at the fastest voltage and frequency.
2) Numbers at the most energy-efficient voltage and frequency.
3) Estimated with LPDDR3 SDRAM.

Figure 23.1.6: Measurement result and comparison table.



	Specification			
Technology	16-nm FinFET			
Supply Voltage [V]	0.45 – 0.85			
Frequency [MHz]	60 – 450			
Power [mW]	7.12 – 152.5			
Area [mm²]	10.15			
Precision Support	INT4/8/16			
On-Chip Memory [kB]	1,320			
On-Chip-Level Evaluation				
Precision	INT4	INT8	INT16	
Performance [TOPS] 1)	3.25 – 8.61	0.81 – 2.15	0.20 – 0.54	
Energy Efficiency [TOPS/W] 2)	60.8 – 161.2	15.2 – 40.3	3.8 – 10.1	
Area Efficiency [TOPS/mm²] 1)	0.32 – 0.85	0.08 – 0.21	0.02 – 0.053	
System-Level Evaluation				
Model	ViT-B	RD-NMT	S2T-M	BERT-L
Reduced EMAs per input [MB]	5.47	0.96	3.48	10.03
Average HW Utilization [%]	89.0	57.4	75.8	72.0
Latency [μs/token] 1,3)	567	68	233	475
Energy [μJ/token] 2,3)	3.662	0.407	1.645	3.946

1) Numbers at 0.85V and 450MHz.
2) Numbers at 0.47V and 80MHz.
3) Estimated with LPDDR3 SDRAM..

Figure 23.1.7: Chip photograph and performance summary.

References:

[1] Y. Wang et al., "A 28nm 27.5TOPS/W Approximate-Computing-Based Transformer Processor with Asymptotic Sparsity Speculating and Out-of-Order Computing," *ISSCC*, pp. 464-465, 2022.

[2] F. Tu et al., "A 28nm 15.59uJ/Token Full-Digital Bitline-Transpose CIM-Based Sparse Transformer Accelerator with Pipeline/Parallel Reconfigurable Modes," *ISSCC*, pp. 466-467, 2022.

[3] 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," *ISSCC*, pp. 250-251, 2023.

[4] Y. Wang et al., "A 28nm 77.35TOPS/W Similar Vector Traceable Transformer Processor with Principal-Component-Prior Speculating and Dynamic Bit-wise Stationary Computing," *IEEE Symp. VLSI Circuits*, C16-5, 2023.

[5] H. You et al., "ViTCoD: Vision Transformer Acceleration via Dedicated Algorithm and Accelerator Co-Design," *IEEE HPCA*, 2023.

[6] P. Dong et al., "HeatViT: Hardware-Efficient Adaptive Token Pruning for Vision Transformers," *IEEE HPCA*, 2023.

[7] J. Dass et al., "ViTALiTy: Unifying Low-rank and Sparse Approximation for Vision Transformer Acceleration with a Linear Taylor Attention," *IEEE HPCA*, 2023.

[8] B. Keller et al., "A 17-95.6 TOPS/W Deep Learning Inference Accelerator with Per-Vector Scaled 4-bit Quantization for Transformers in 5nm," *IEEE Symp. VLSI Circuits*, C2-1. 2022.

[9] S. Moon et al., "A 127.8TOPS/W Arbitrarily Quantized 1-to-8b Scalable-Precision Accelerator for General-Purpose Deep Learning with Reduction of Storage, Logic and Latency Waste," *ISSCC*, pp. 330-331, 2023.

[10] F. Tu et al., "MulTCIM: A 28nm 2.24uJ/Token Attention-Token-Bit Hybrid Sparse Digital CIM-Based Accelerator for Multimodal Transformers," *ISSCC*, pp. 248-249, 2023.

[11] H. Mun et al., "A 28 nm 66.8 TOPS/W Sparsity-Aware Dynamic-Precision Deep-Learning Processor," *IEEE Symp. VLSI Circuits*, C16-1, 2023.

[12] B. Keller et al., "A 95.6-TOPS/W Deep Learning Inference Accelerator With Per-Vector Scaled 4-bit Quantization in 5 nm," *IEEE JSSC*, vol. 58, no. 4, pp. 1129-1141, 2023.

[13] Y. Qin et al., "FACT: FFN-Attention Co-optimized Transformer Architecture with Eager Correlation Prediction," *IEEE/ACM ISCA*, 2023.

[14] C. Tang et al., "A 28nm 4.35TOPS/mm² Transformer Accelerator with Basis-vector Based Ultra Storage Compression, Decomposed Computation and Unified LUT-Assisted Cores," *IEEE Symp. VLSI Circuits*, 2024.

[15] P. Wu et al., "A 99.2TOPS/W Transformer Learning Processor with Approximated Attention Score Gradient Computation and Ternary Vector-based Speculation," *IEEE Symp. VLSI Circuits*, C10-3, 2024.

[16] Y. Wang et al., "A 22nm 54.94TFLOPS/W Transformer Fine-Tuning Processor with Exponent-Stationary Re-computing, Aggressive Linear Fitting, and Logarithmic Domain Multiplicating," *IEEE Symp. VLSI Circuits*, 2024.

[17] S. Moon et al., "Multipurpose Deep-Learning Accelerator for Arbitrary Quantization With Reduction of Storage, Logic, and Latency Waste," *IEEE JSSC*, vol. 59, no. 1, pp. 143-156, 2024.

[18] Y. Qin et al., "Ayaka: A versatile Transformer Accelerator with Low-Rank Estimation and Heterogeneous Dataflow," *IEEE JSSC*, 2024.

[19] T. Tambe et al., "A 12nm 18.1TFLOPs/W Sparse Transformer Processor with Entropy-Based Early Exit, Mixed-Precision Prediction and Fine-Grained Power Management," *ISSCC*, pp. 342-343, 2023.

[20] B. Zhang et al., "A 1-TFLOPS/W, 28-nm Deep Neural Network Accelerator featuring Online Compression and Decompression and BF16 Digital In-Memory-Computing Hardware," *IEEE CICC*, 26-3, 2024.

[21] S. Kim et al., "C-Transformer: A 2.6-18.1uJ/Token Homogeneous DNN-Transformer/Spiking-Transformer Processor with Big-Little Network and Implicit Weight Generation for Large Language Models," *ISSCC*, pp. 368-369, 2024.

[22] Y.-C. Bae, et al. "A 1.2V 30nm 1.6Gb/s/pin 4Gb LPDDR3 SDRAM with input skew calibration and enhanced control scheme," *ISSCC*, pp. 44-46, 2012.

[23] D. Dutoit et al., "A 0.9 pJ/bit, 12.8 GByte/s WideIO Memory Interface in a 3D-IC NoC-based MPSoC," *IEEE Symp. VLSI Circuits*, pp. C22-C23, 2013.

[24] Q. Lou et al., "DictFormer: Tiny Transformer with Shared Dictionary," *ICLR*, 2022.

[25] A. Dosovitskiy et al., "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," arXiv:2010.11929, 2021.

[26] X. Liang et al., "R-Drop: Regularized Dropout for Neural Networks," arXiv:2106.14448, 2021.

[27] C. Wang et al., "fairseq S2T: Fast Speech-to-Text Modeling with fairseq," arXiv:2010.05171, 2022.

[28] J. Devlin et al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," arXiv:1810:04805, 2019.

979-8-3315-4101-9/25/$31.00 ©2025 IEEE