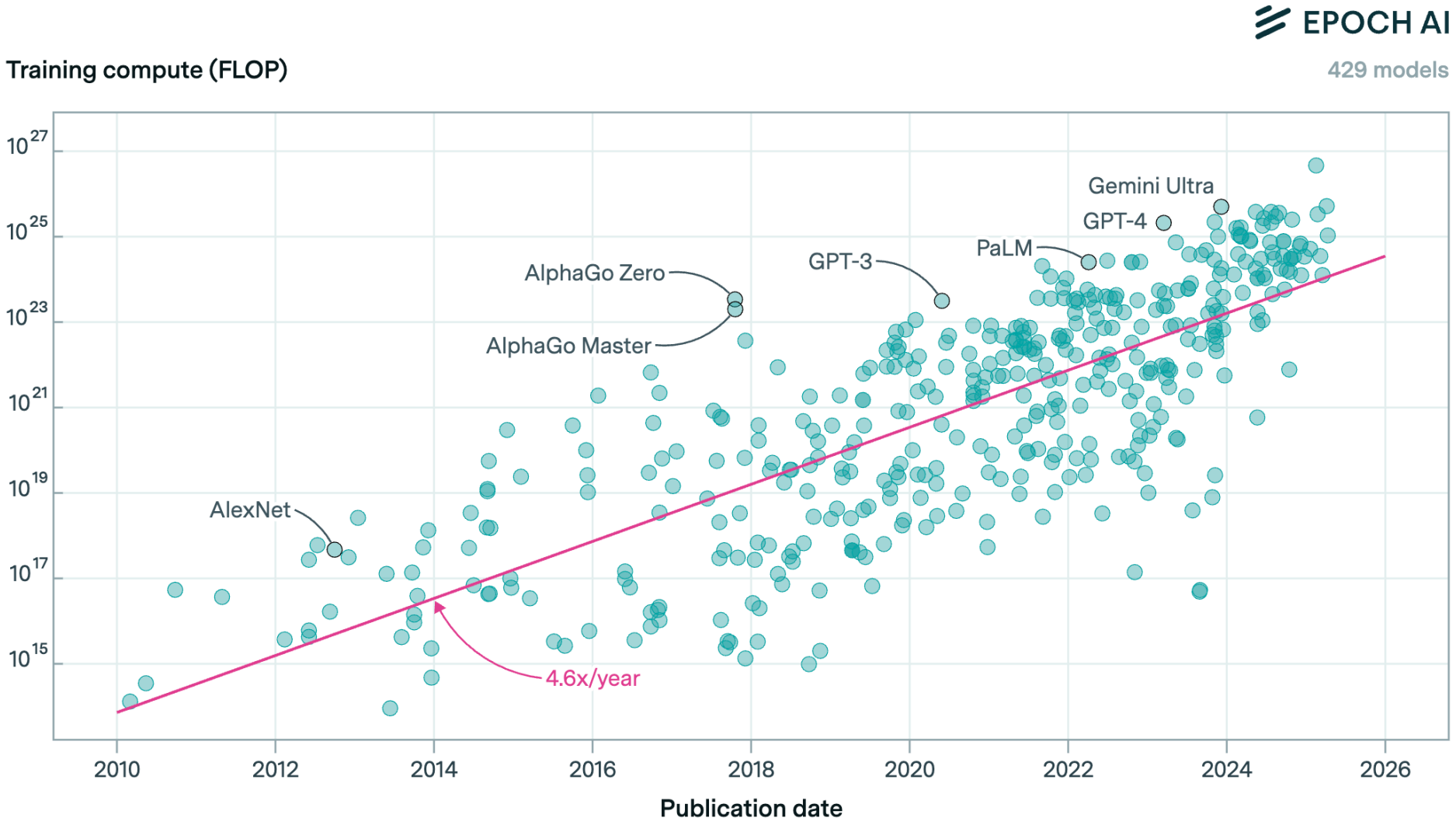
Adaptive Accelerators for Sparse LLMs

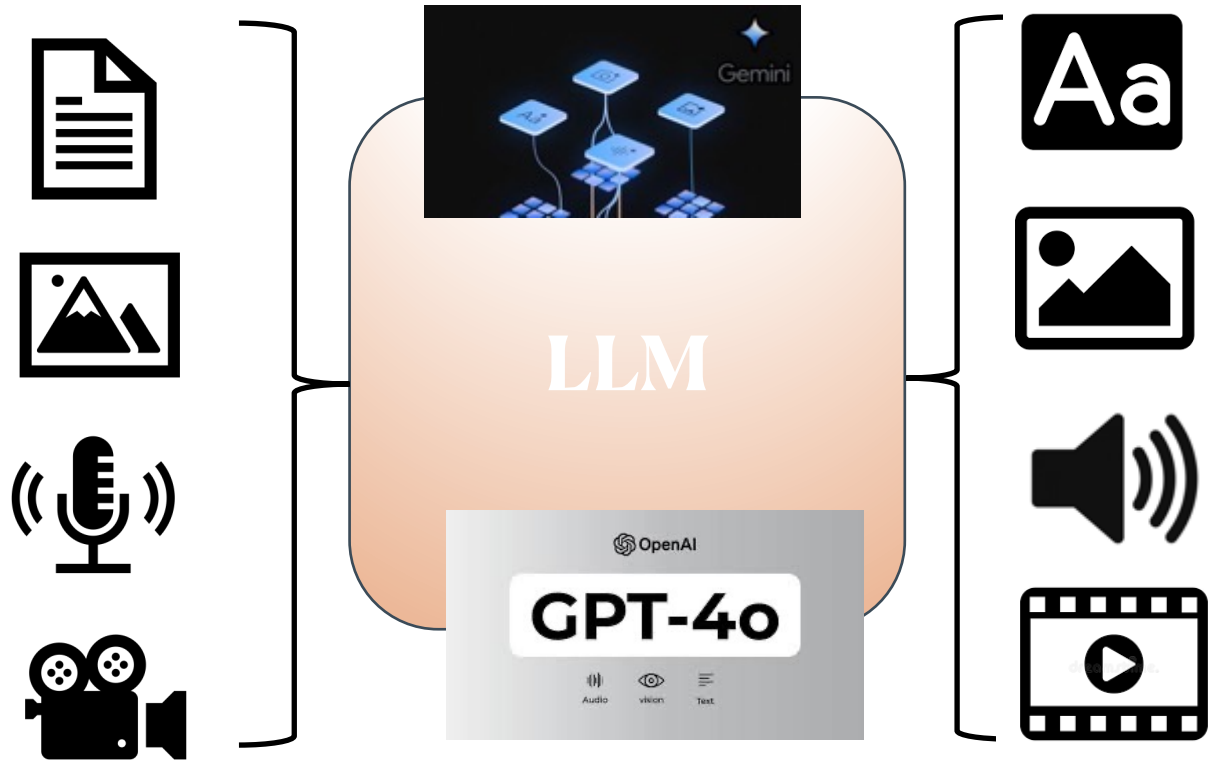
Tulika Mitra
National University of Singapore

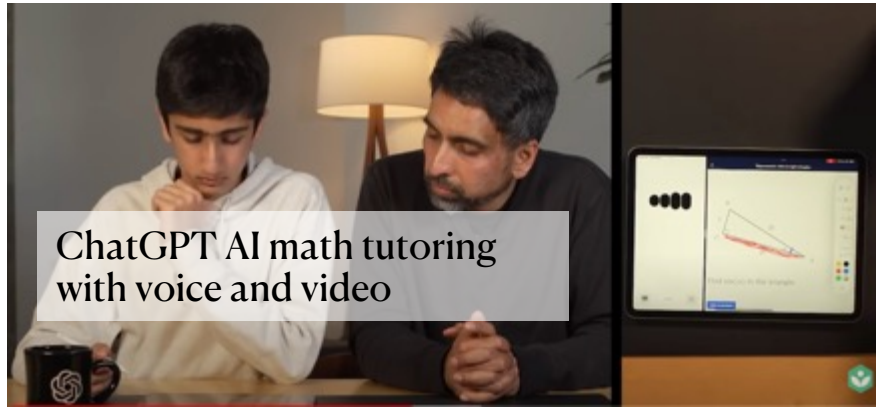
DAC 2025: Keynote at 2nd International Workshop on Deep Learning-Hardware Co-Design for Generative AI Acceleration

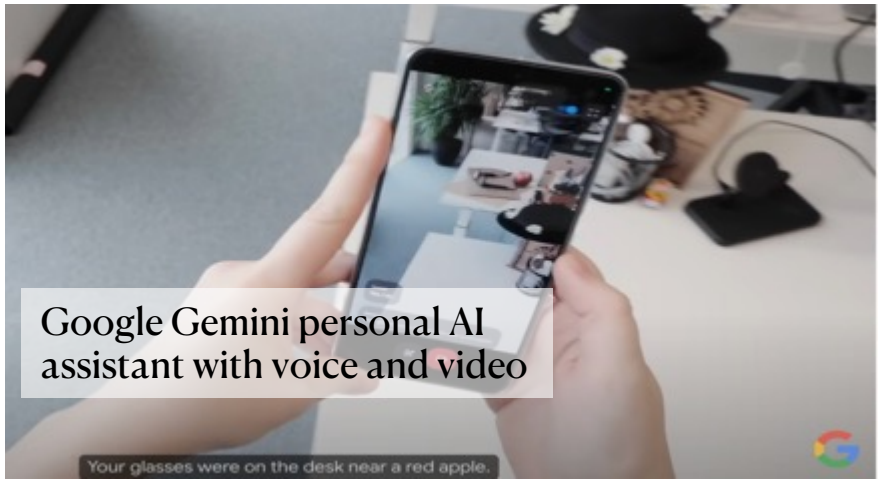
AI model size is growing ~10x/year



Emergence of Native Multimodal LLMs
unifying AI across modalities



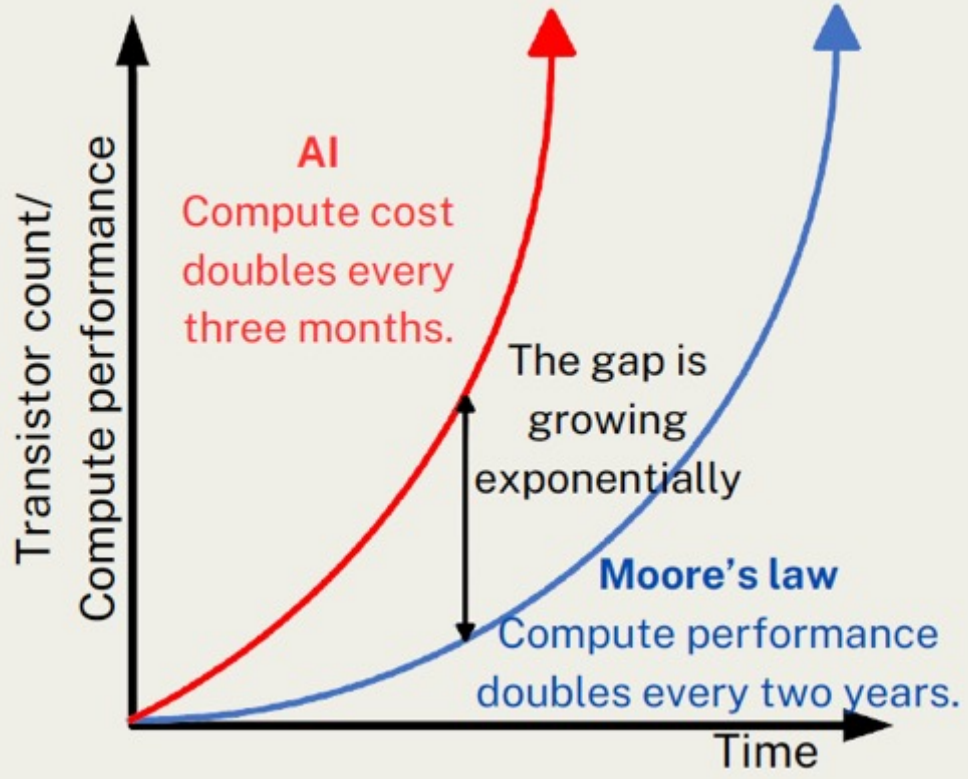




- Seamless interaction with physical world via integration of multimodal data
- Process and generate text, images, audio, and video within a single model
- Perfect foundation for embodied intelligence

AI Compute Challenge

AI compute gap



Short battery life

Power-hungry applications drain battery life very quickly.

AI power gap

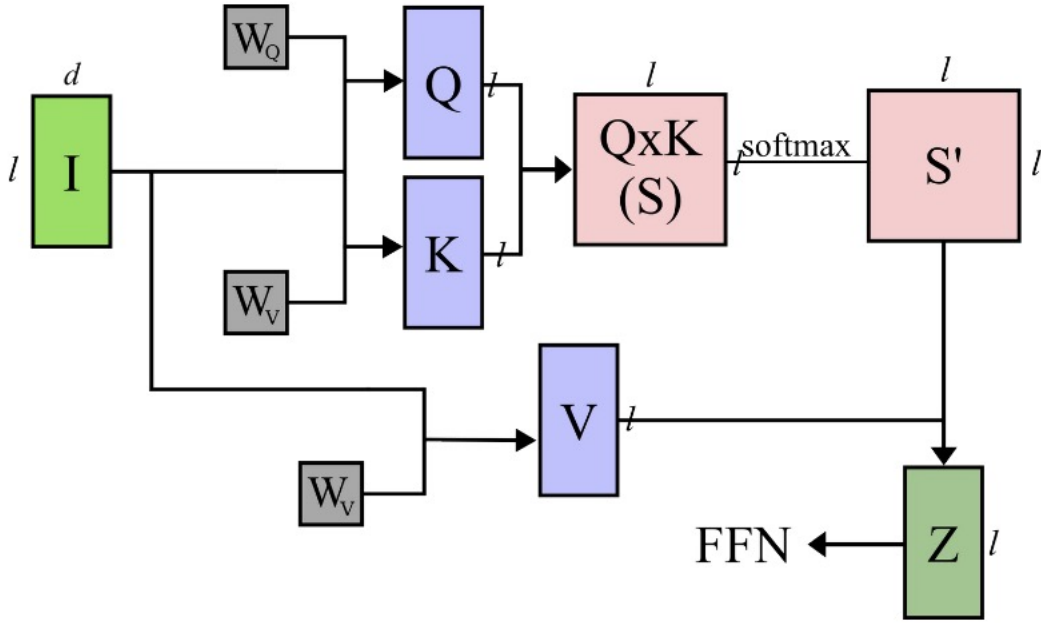
GPU
700W

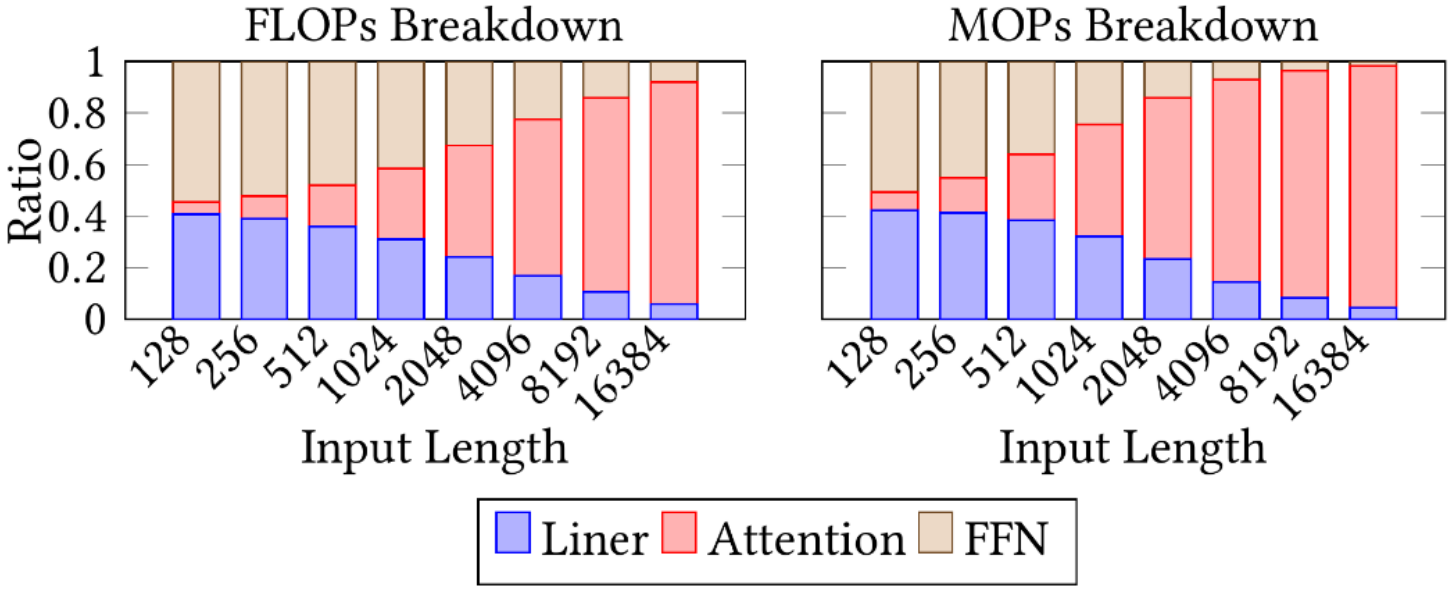
Brain
20W

Watch
1W

Self-Attention: Poor Scalability with Context Length

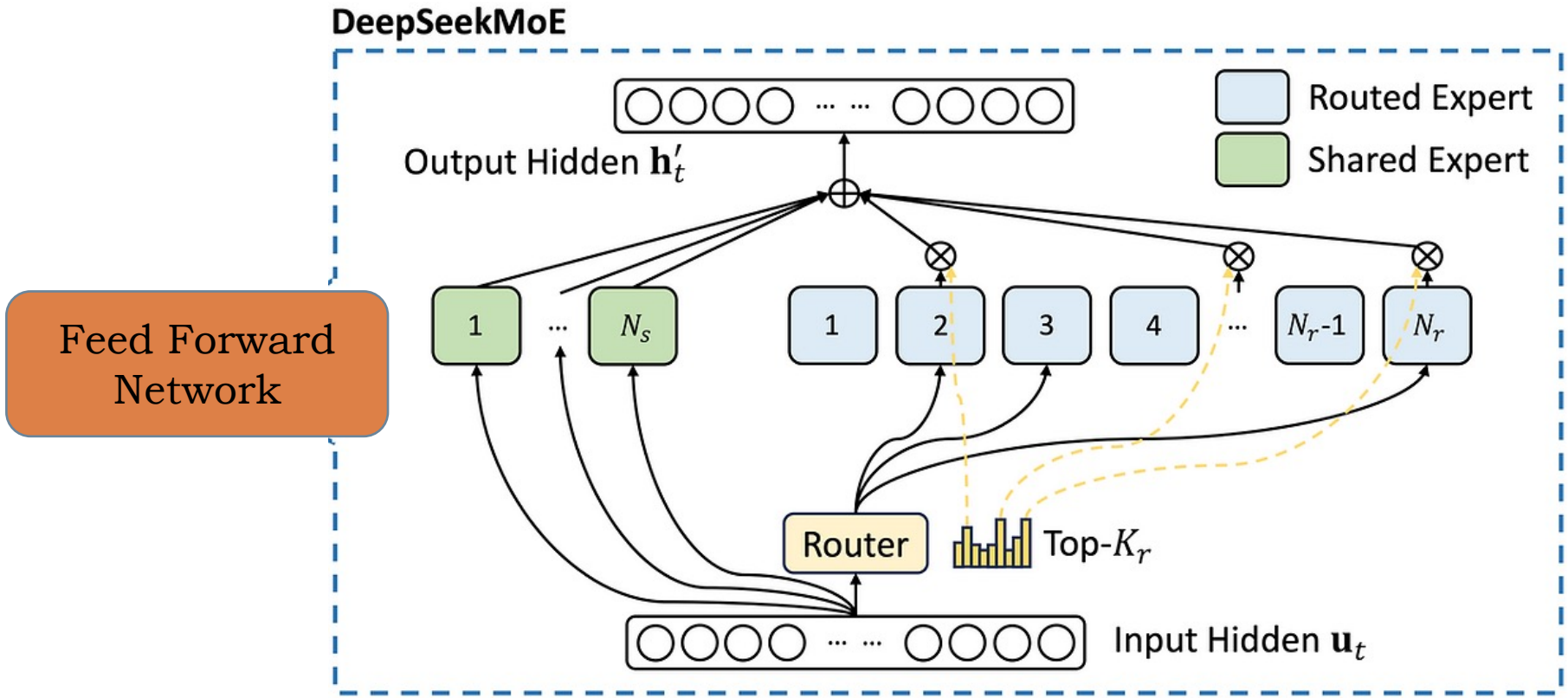
Attention scales poorly with context length, requiring quadratic operations and memory for n tokens





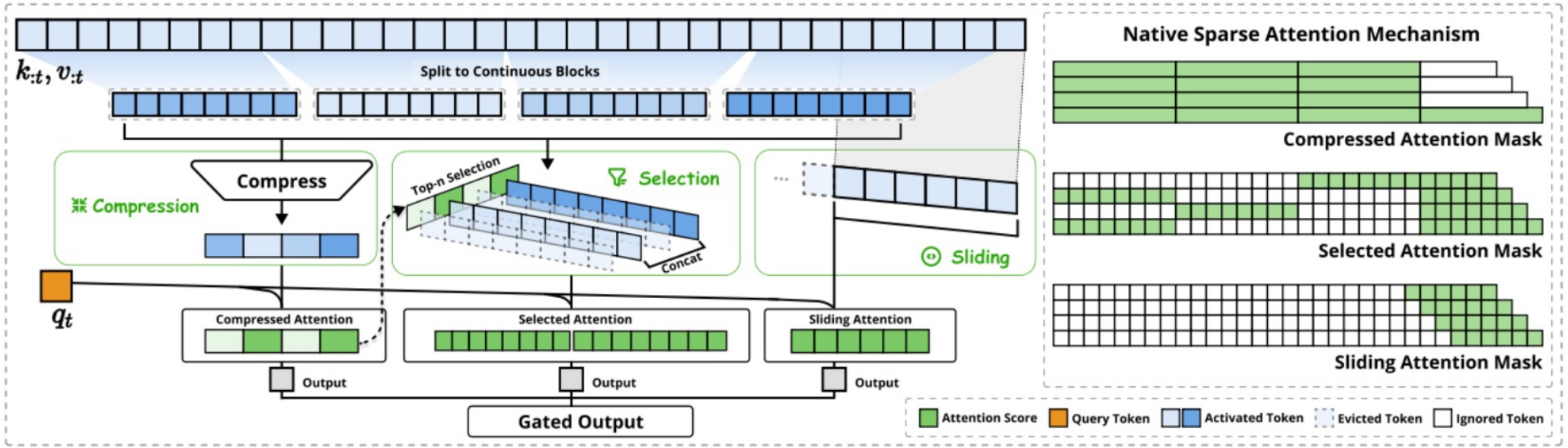
Sparsity in LLM Workloads

- **Mixture of Experts (MoE):** Only a subset of expert modules are activated per token: ~5.5% active parameters in Deepseek-v3 671B



Sparsity in LLM Workloads

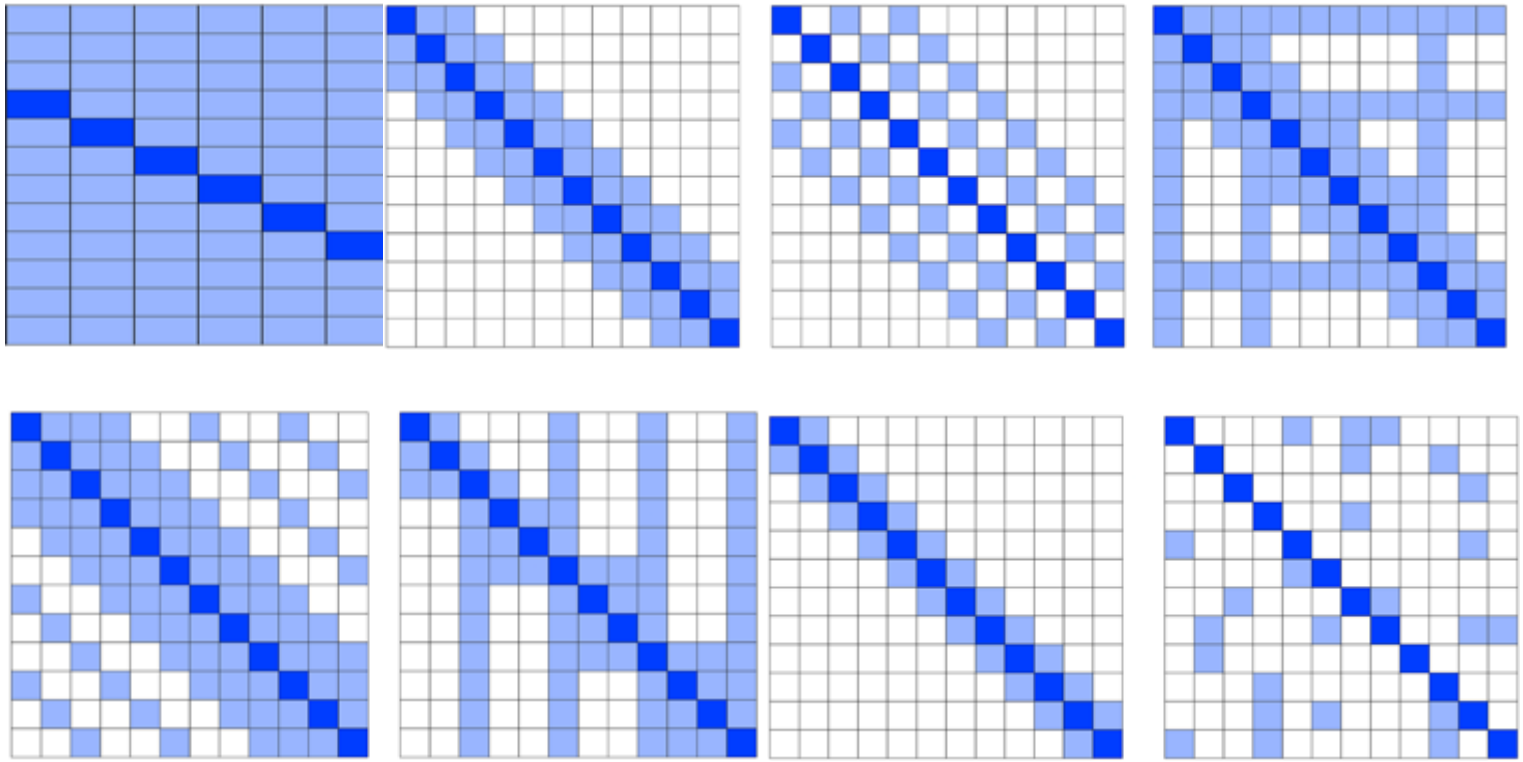
- **Mixture of Experts (MoE):** Only a subset of expert modules are activated per token: ~5.5% active parameters in Deepseek-v3 671B
- **Structured sparsity in attention:** Sparse attention patterns (e.g., Mistral's sliding window, DeepSeek native sparse attention)



Deepseek Native Sparsity

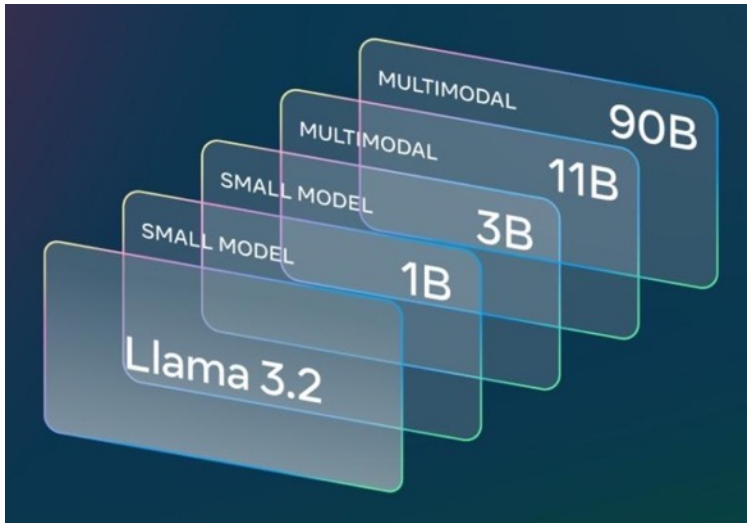
Structured Sparsity in Attentions

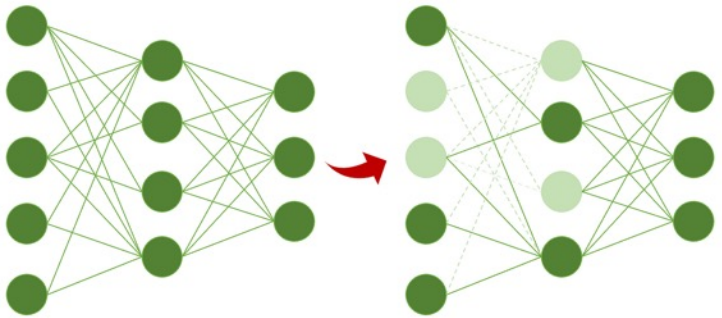
- Advanced attention mechanisms leverage diverse **sparse matrix computations** with specialized patterns for enhanced efficiency
- Each type of attention mechanism requires **unique dataflow**

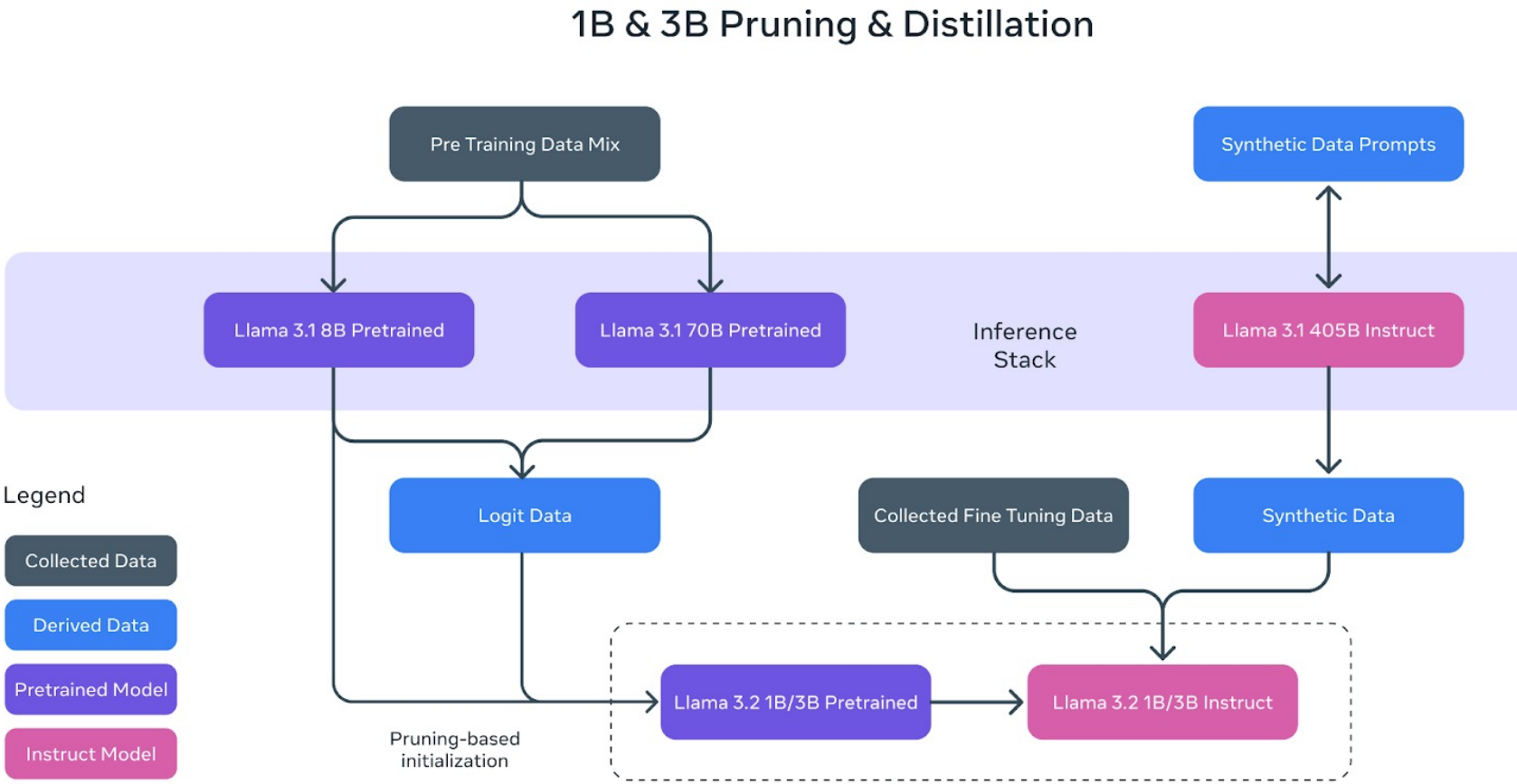


NUS National University of Singapore

Pruning + Knowledge Distillation + Quantization





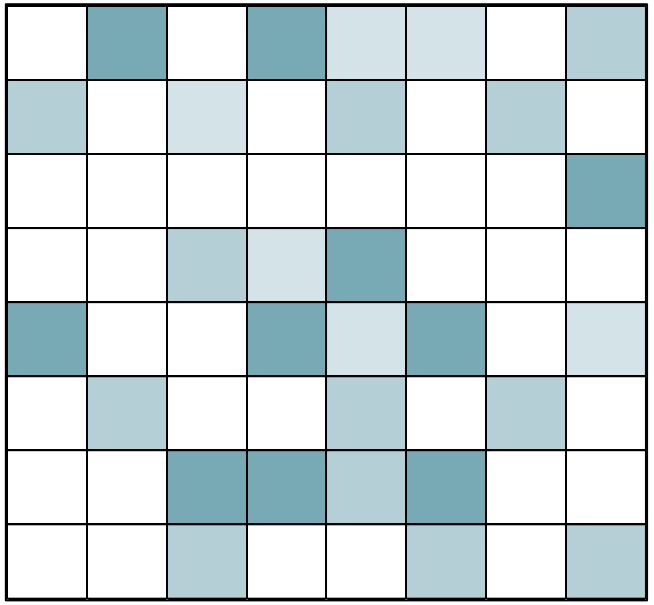


Model Size	BF16/FP16	FP8	INT4
3B	6.5 GB	3.2 GB	1.75 GB
1B	2.5 GB	1.25 GB	0.75 GB

Source: Meta

Sparsity in LLM Workloads

- **Mixture of Experts (MoE):** Only a subset of expert modules are activated per token: ~5.5% active parameters in Deepseek-v3 671B
- **Structured sparsity in attention:** Sparse attention patterns (e.g., Mistral's sliding window, DeepSeek native sparse attention)
- **Unstructured Sparsity:** Individual parameters becoming zero via pruning, activation, or training-time sparsity constraints



We need sparsity for efficiency…
But is the hardware ready?

The Hardware Lottery

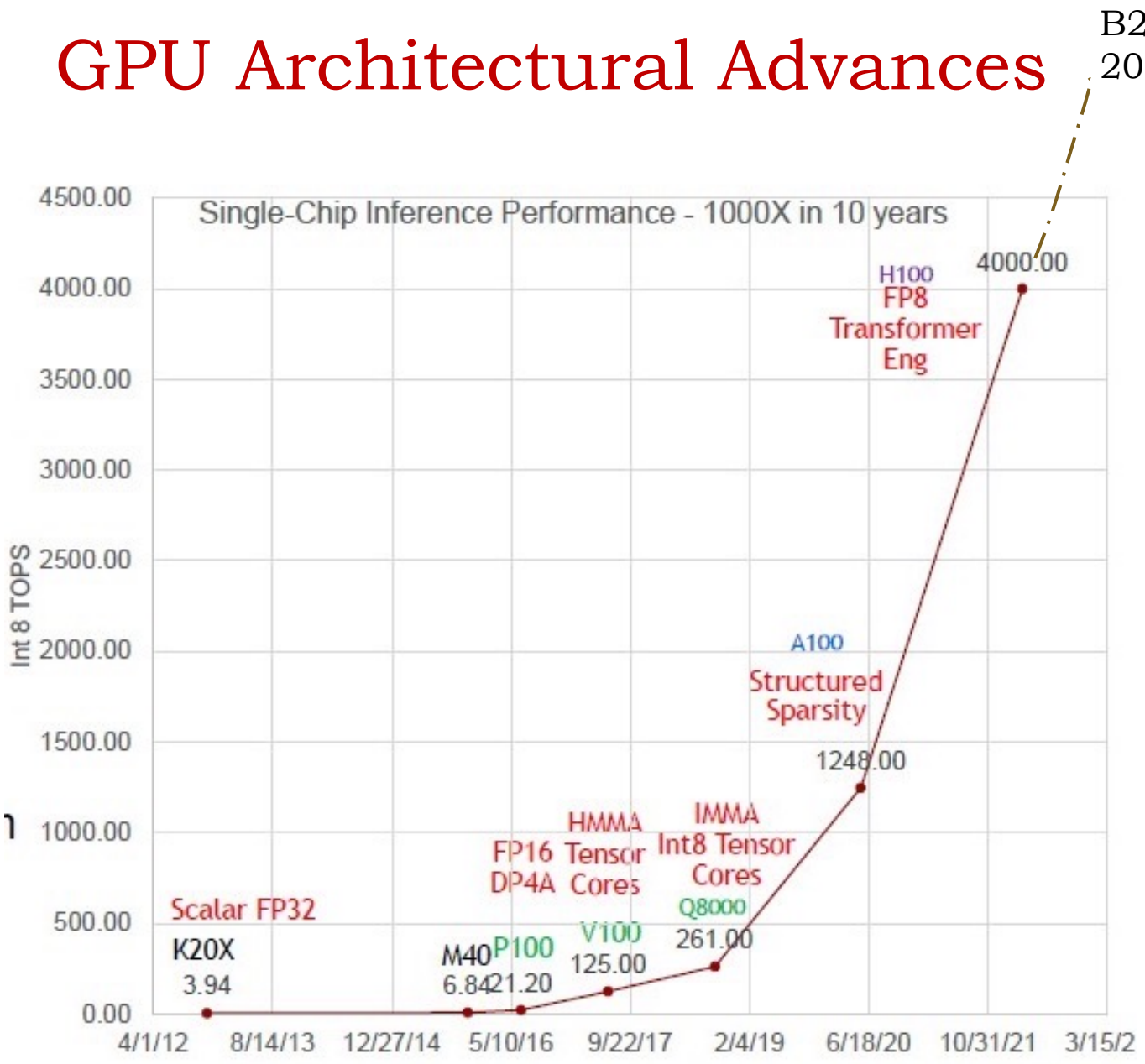
Sara Hooker

Google Brain
shooker@google.com

Transformer Architecture is ideally matched with GPUs

Hardware, systems and algorithms research communities have historically had different incentive structures and fluctuating motivation to engage with each other explicitly. This historical treatment is odd given that hardware and software have frequently determined which research ideas succeed (and fail). This essay introduces the term hardware lottery to describe when a research idea wins because it is suited to the available software and hardware and *not* because the idea is superior to alternative research directions. Examples from early computer science history illustrate how hardware lotteries can delay research progress by casting successful ideas as failures. These lessons are particularly salient given the advent of domain specialized hardware which makes it increasingly costly to stray off of the beaten path of research ideas. This essay posits that the gains from progress in computing are likely to become even more uneven, with certain research directions moving into the fast-lane while progress on others is further obstructed.

GPU Architectural Advances



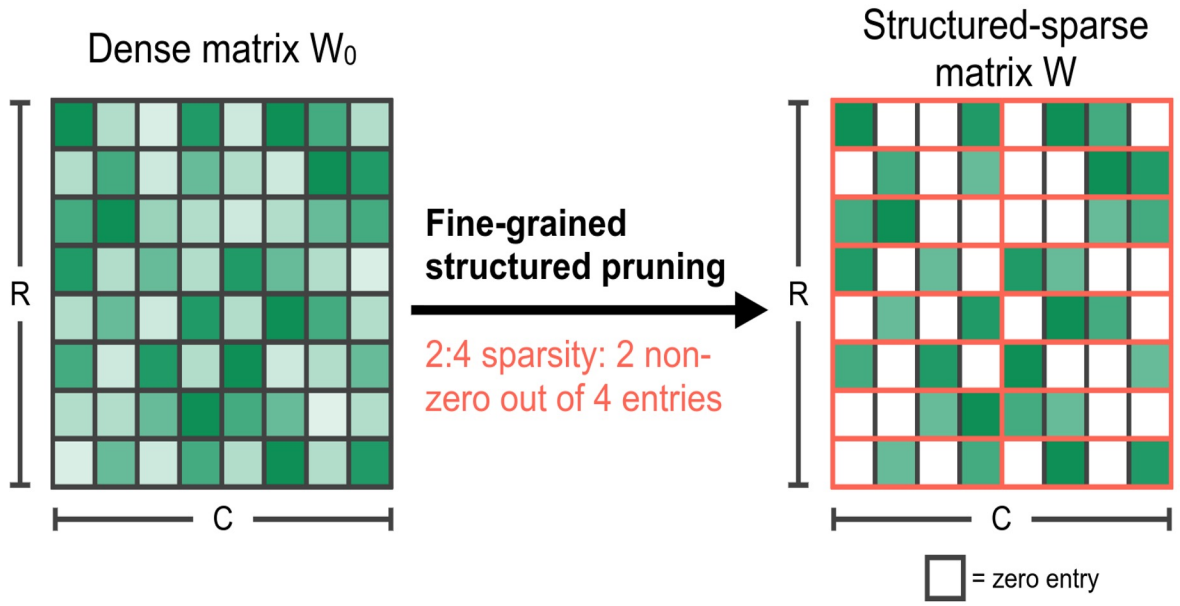
Blackwell gains w.r.t. K20X

- Number Representation (~32x)
  FP32, FP16, Int8, FP4
- Complex Instructions (~12.5×)
  DP4, HMMA, IMMA
- Technology node (~3x)
  28m, 16nm, 7nm, 5nm, 4nm
- Die Size 2x

NUS National University of Singapore

Source: Nvidia; Bill Dally

GPUs Adopted Restricted Structured Sparsity



GPUs cannot deal with unstructured sparsity

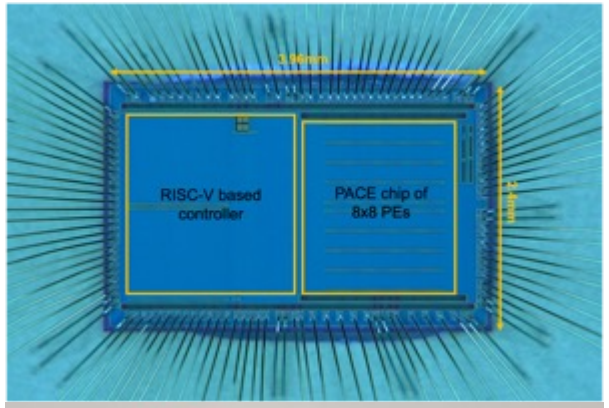
Blackwell gains w.r.t. K20X

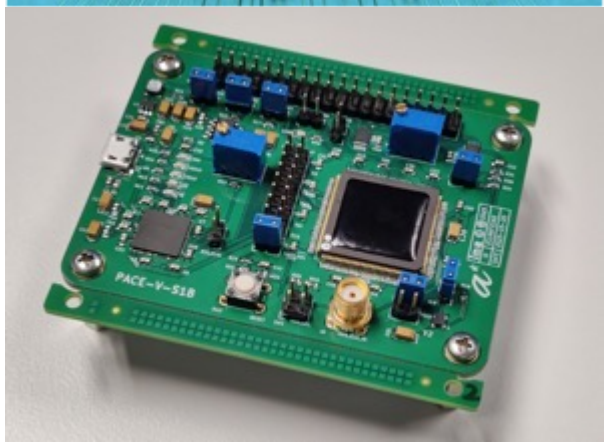
- Number Representation (~32x)
  FP32, FP16, Int8, FP4
- Complex Instructions (~12.5×)
  DP4, HMMA, IMMA
- Technology node (~3x)
  28m, 16nm, 7nm, 5nm, 4nm
- Structured sparsity (~2x)
- Die Size 2x
- Total gain: ~4,800x in 11 years

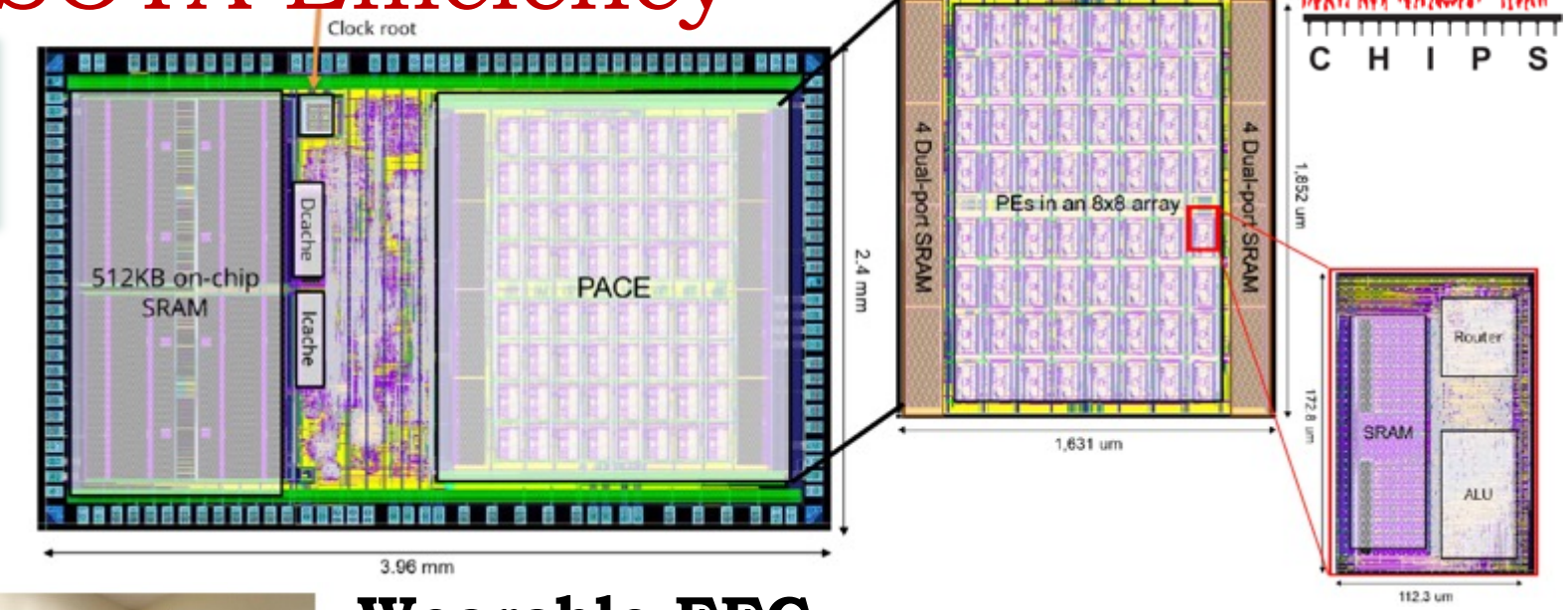
Source: Nvidia; Bill Dally

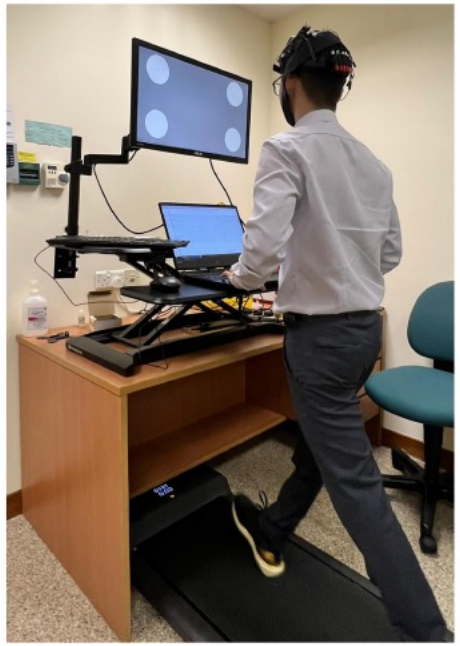
NUS PACE CGRA: SOTA Efficiency

8x8 CGRA: 1.1 mW at 10MHz
582 GOPS/W at 0.45V, 40nm ULP
Estimated 1 TOPS/W at 22nm

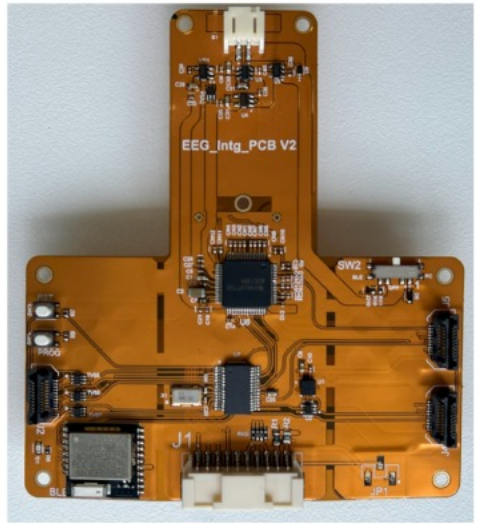






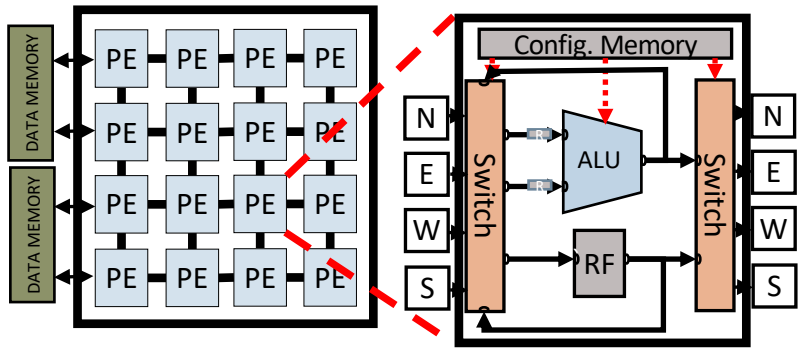


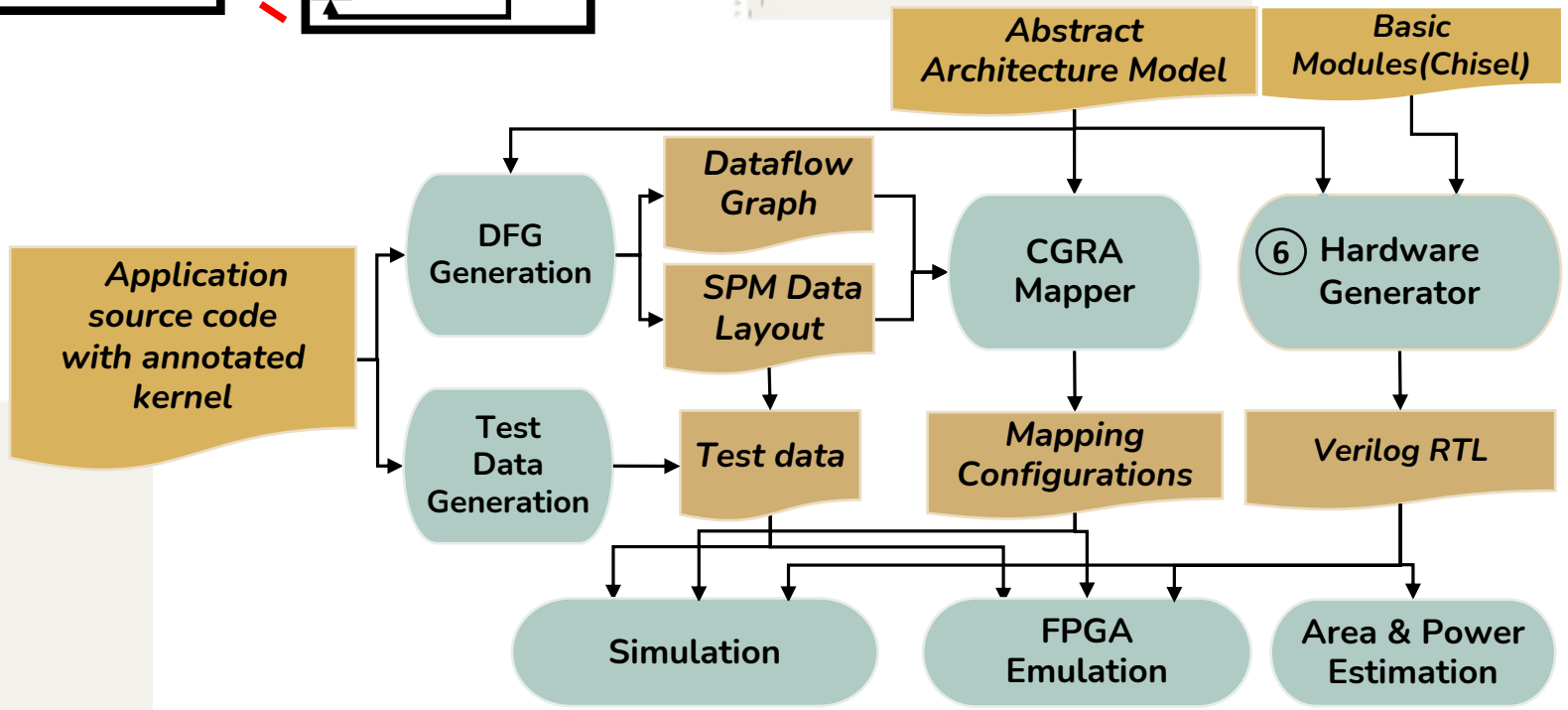
Wearable EEG

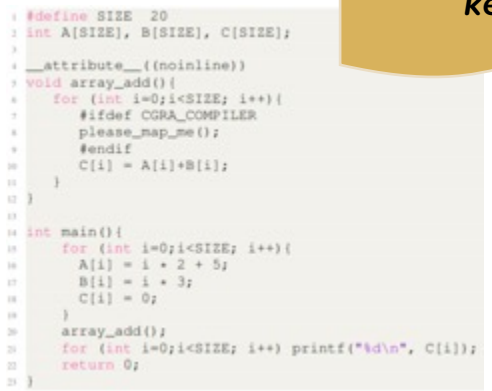


NUS Morpher Open-Source CGRA Toolchain

SCAN ME



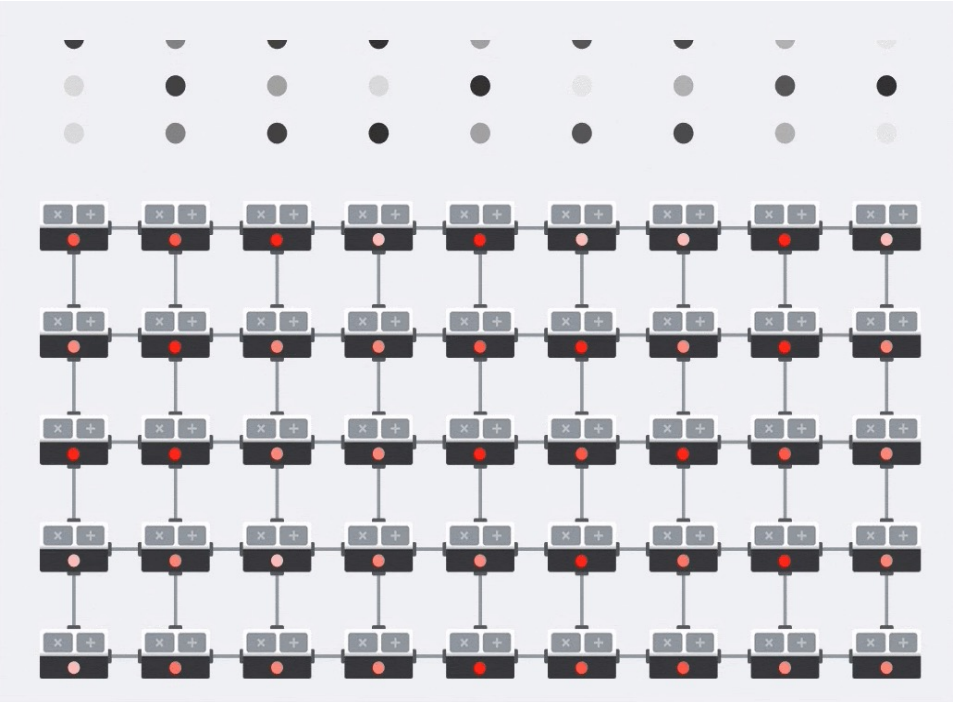




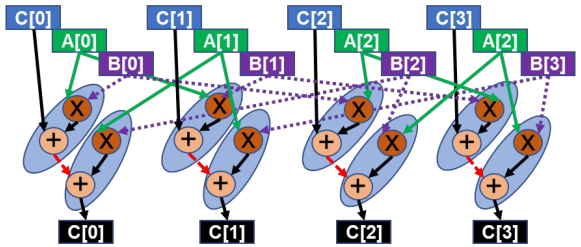
https://github.com/ecolab-nus/morpher

Regular Dataflow on TPU and CGRA

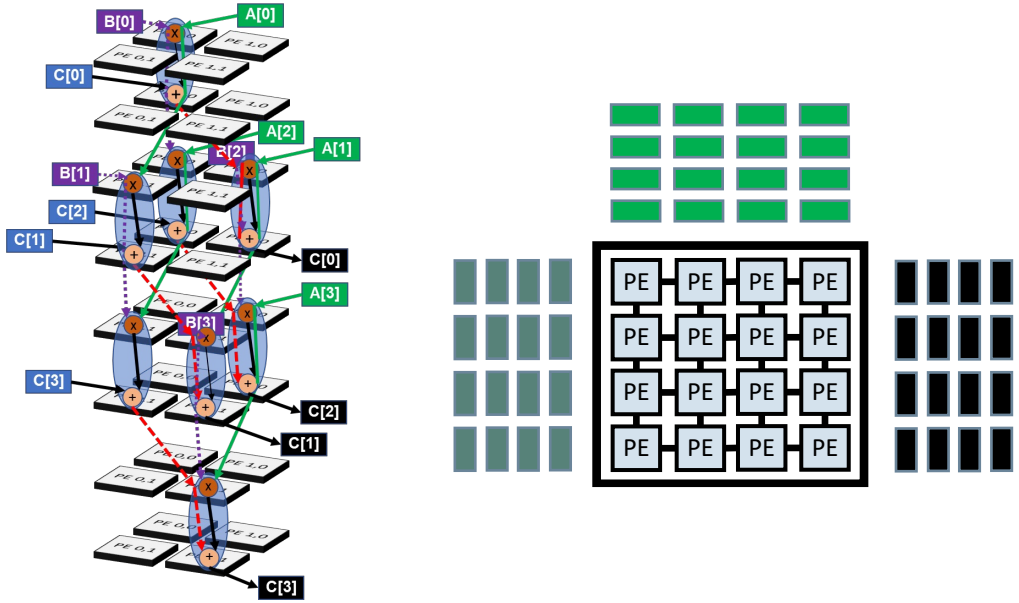
Spatial Accelerators cannot deal with unstructured sparsity easily



Google TPU Matrix Multiply Unit (MXU) Dataflow

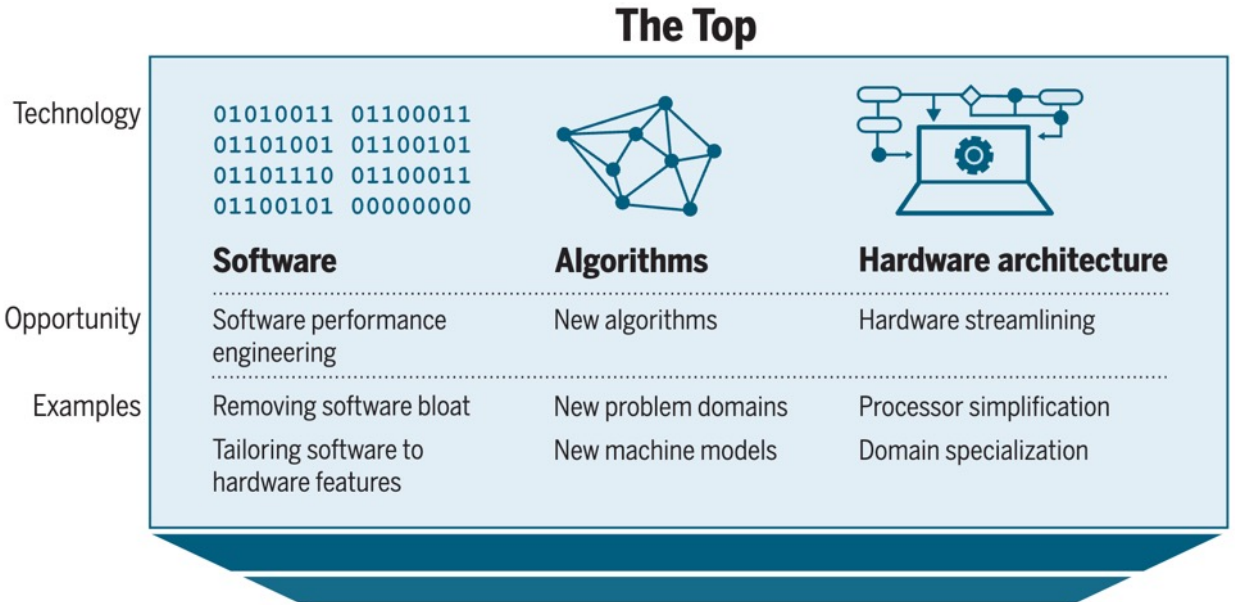


GEMM DFG



HiMap GEMM Schedule & Dataflow on CGRA

Winning the Hardware Lottery: Algorithm-Architecture Codesign for Sparse LLMs



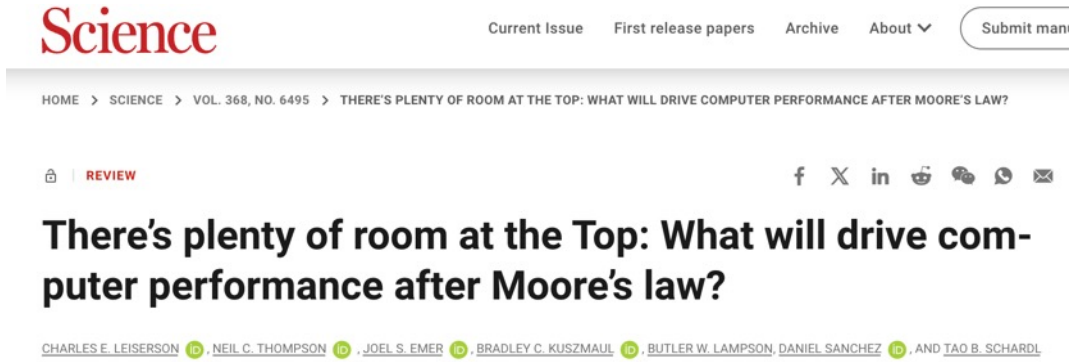
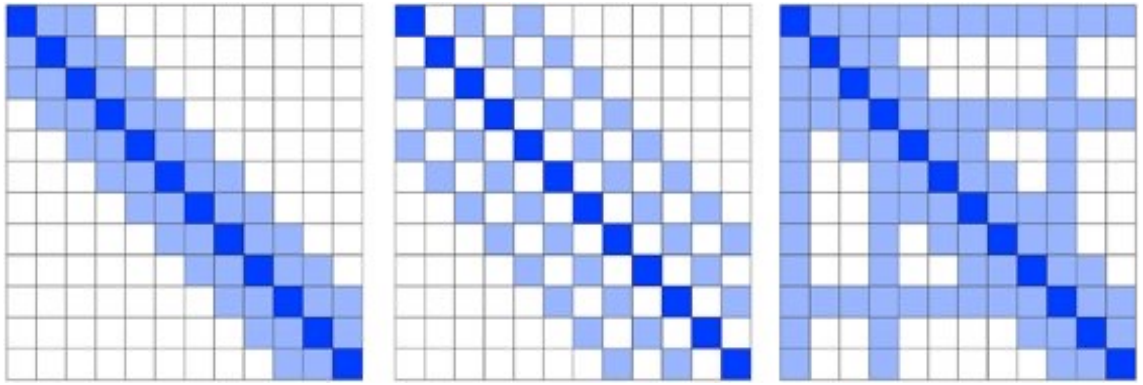


Table 1. Speedups from performance engineering a program that multiplies two 4096-by-4096 matrices. Each version represents a successive refinement of the original Python code. "Running time" is the running time of the version. "GFLOPS" is the billions of 64-bit floating-point operations per second that the version executes. "Absolute speedup" is time relative to Python, and "relative speedup," which we show with an additional digit of precision, is time relative to the preceding line. "Fraction of peak" is GFLOPS relative to the computer's peak 835 GFLOPS. See Methods for more details.

Version	Implementation	Running time (s)	GFLOPS	Absolute speedup	Relative speedup	Fraction of peak (%)
1	Python	25,552.48	0.005	1	—	0.00
2	Java	2,372.68	0.058	11	10.8	0.01
3	C	542.67	0.253	47	4.4	0.03
4	Parallel loops	69.80	1.969	366	7.8	0.24
5	Parallel divide and conquer	3.80	36.180	6,727	18.4	4.33
6	plus vectorization	1.10	124.914	23,224	3.5	14.96
7	plus AVX intrinsics	0.41	337.812	62,806	2.7	40.45

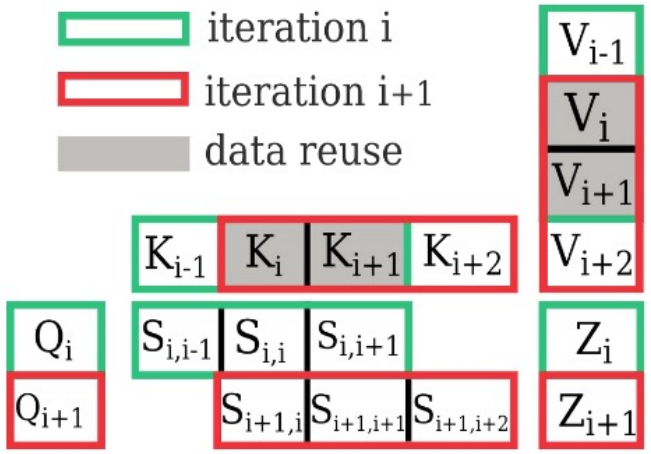
NUS National University of Singapore

Sliding Window Attention

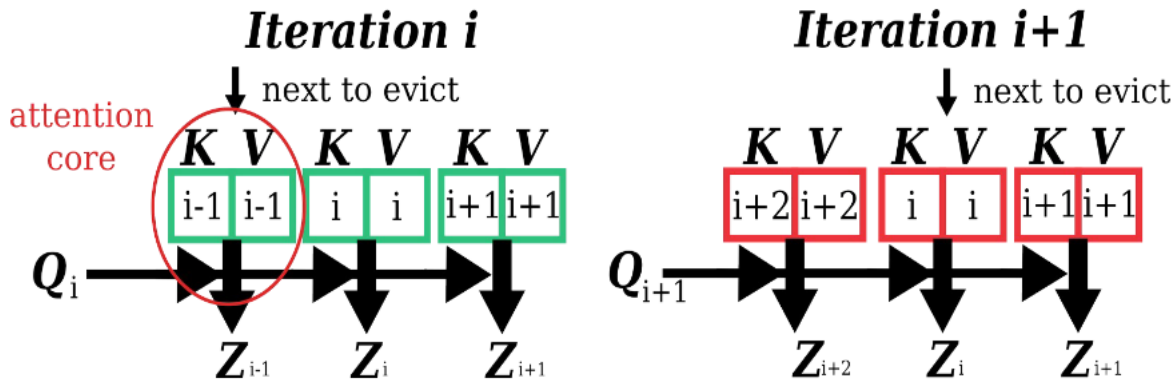


- Each token attend to fixed number of predecessors/successor tokens
  - Reduces the complexity from quadratic to linear
- Often mixed with blocked, row-wise, column-wise attention
- Used in many models: Mistral-7B, DeepSeek MLA, Swin (Vision)
- But….efficient implementation on hardware is hard
- Naïve implementation is worse than dense due to masking overhead
- GPU does not provide the fine granularity control

SWAT: Maximized data reuse



For each Q, the used elements in K has the same index as the elements in V because they are projected from the same input (self-attention)



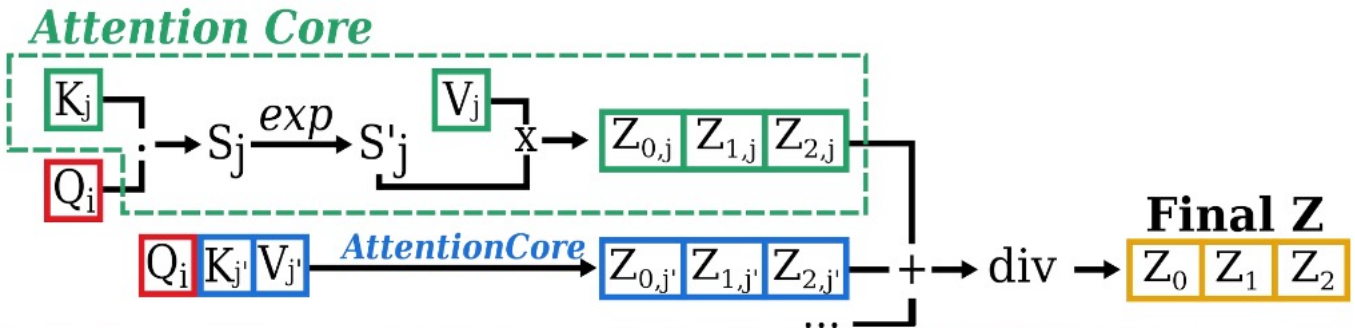
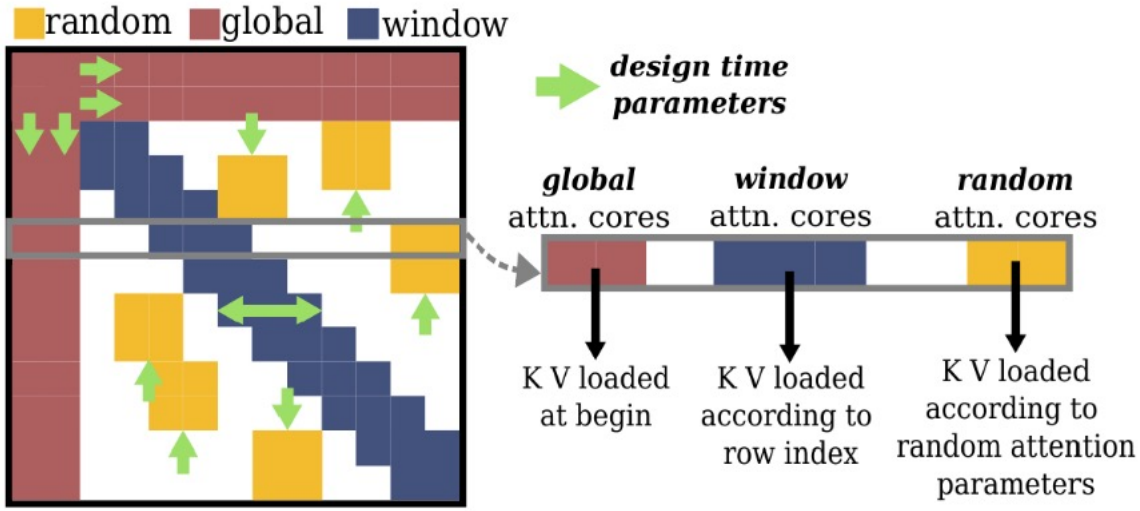


Table 1: The timing (in cycles) of the pipeline stages

LOAD	QK	SV	ZRED1	ZRED2	DIV&OUT
66	201	197	195	66	179
			ROWSUM1	ROWSUM2	
			195	27	

SWAT: Parameterized Design



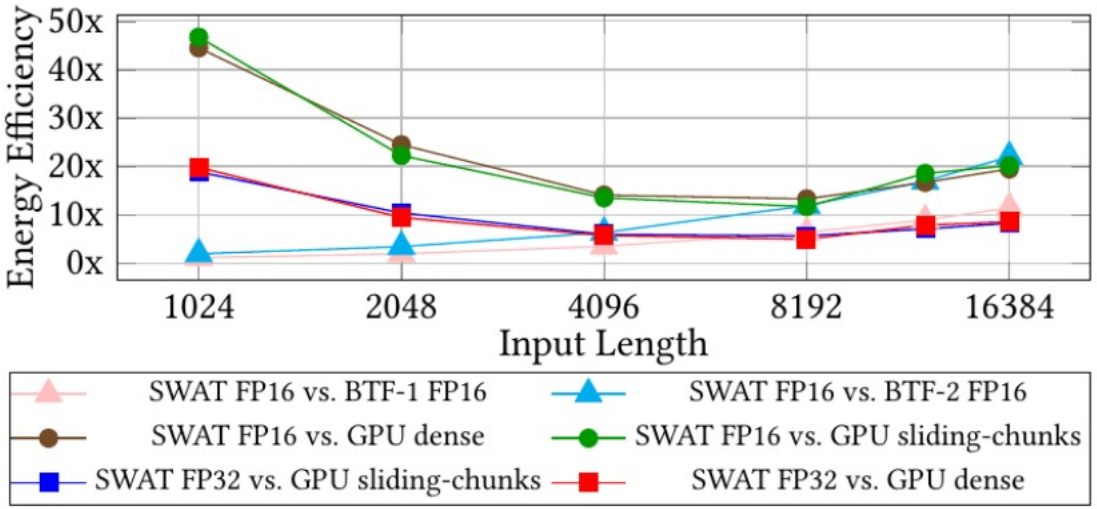
Implementation on Xilinx U55C/U280

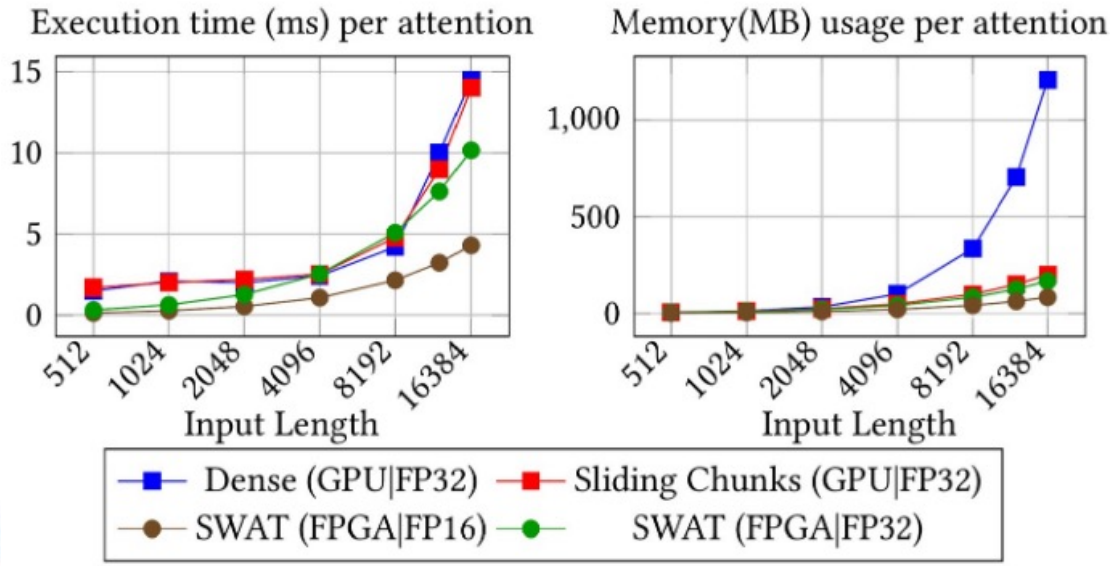
Design	DSP	LUT	FF	BRAM
FP16 (512 attn)	19%	38 %	11%	25%
FP16 (BigBird 512 attn)	19%	33 %	11	25%
FP16 (BigBird 2 x 512 attn)	38%	66 %	22	50%
FP32 (512 attn)	49%	67%	23%	25%
Butterfly (FP16, 120-BE)	32%	79%	63%	49%

SWAT supports a combination of structured patterns: row-wise, column-wise, block-wise, for LongFormer, BigBird, and beyond. The pattern is determined at compile-time and fixed in the bitstream

NUS National University of Singapore

SWAT vs. GPU implementations



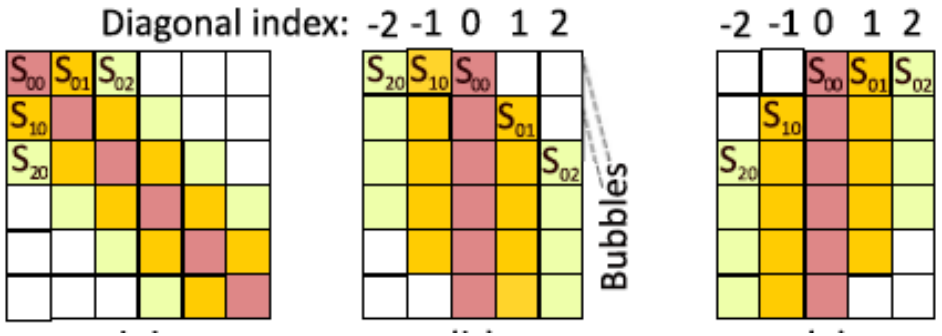


SWAT FPGA Sliding-Window Attention achieves 15x energy-efficiency over GPU

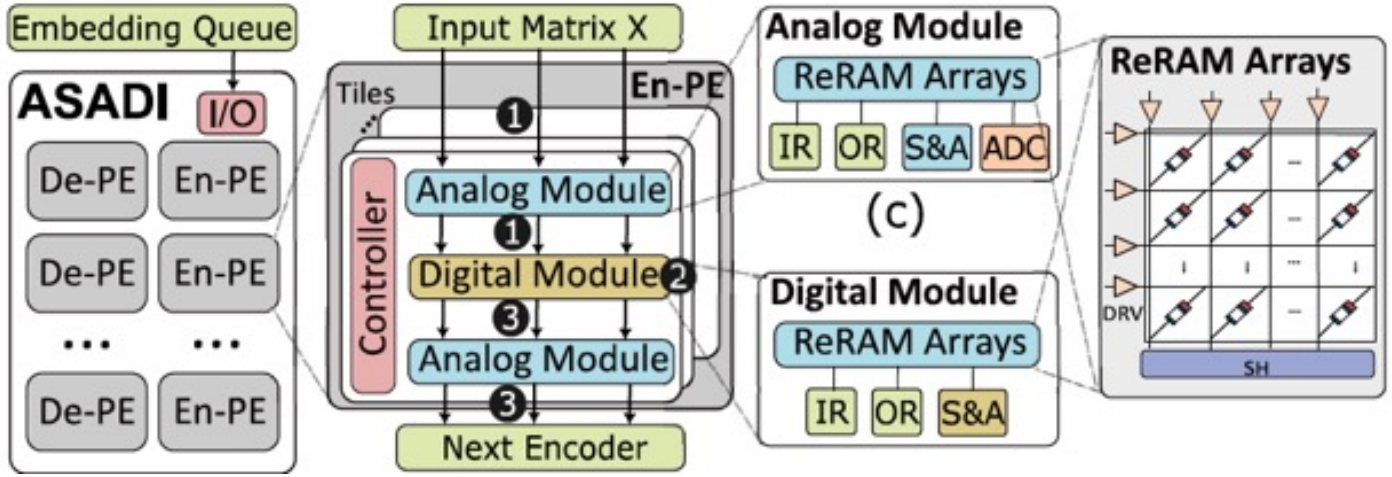
Single-batch:
Xilinx U55C vs. Nvidia A100

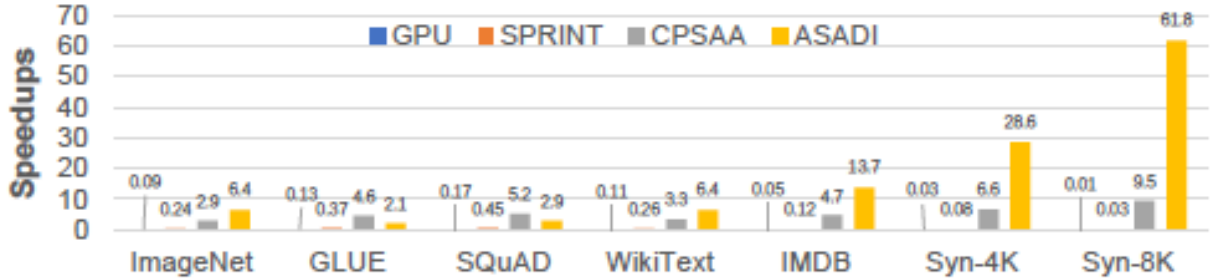
- Improved execution time on worse technology node and much reduced power
- High energy efficiency gain:
  < 4096: A100 is under utilized
  > 4096: SWAT architectural benefits

In-memory computing for sparse attention



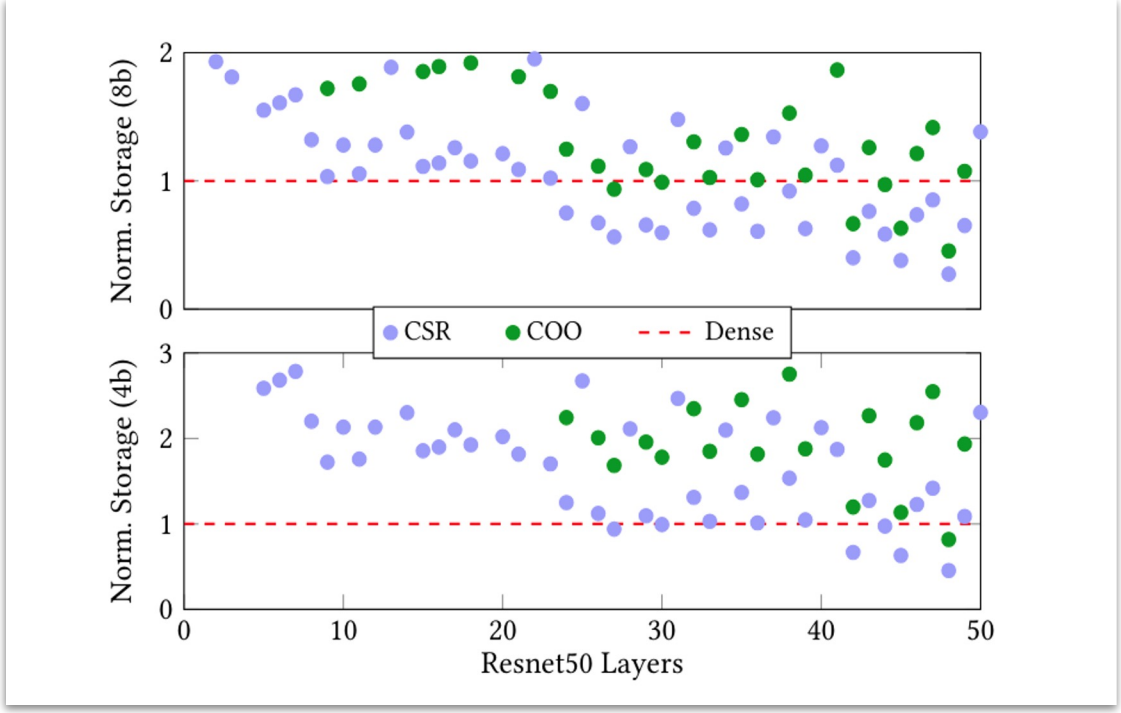
ASADI: ReRAM in-situ computing for sparse attention can achieve at least 13x speedup over GPU

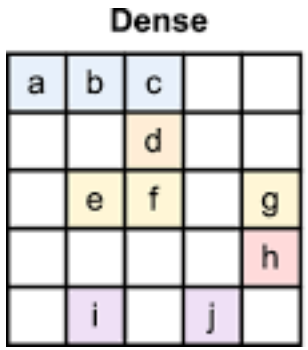


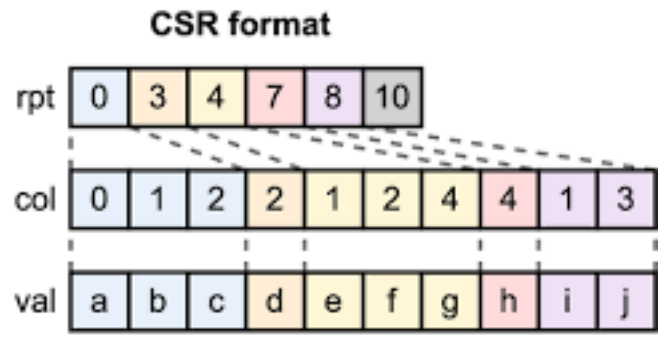


Unstructured Sparsity is hard

At 4-bit or 8-bit quantization, sparse matrix representation formats used by hardware accelerators (CSR, COO) require more storage than Dense format due to meta-data overhead!!

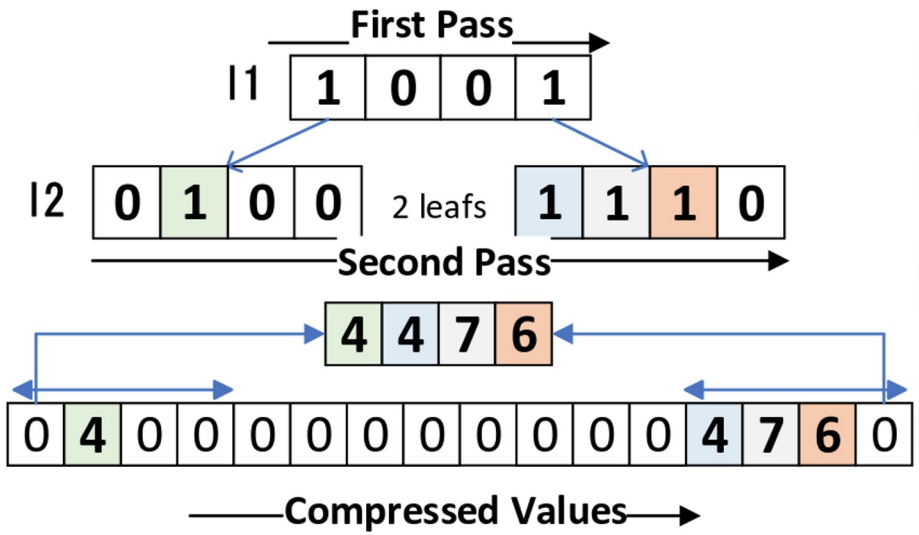


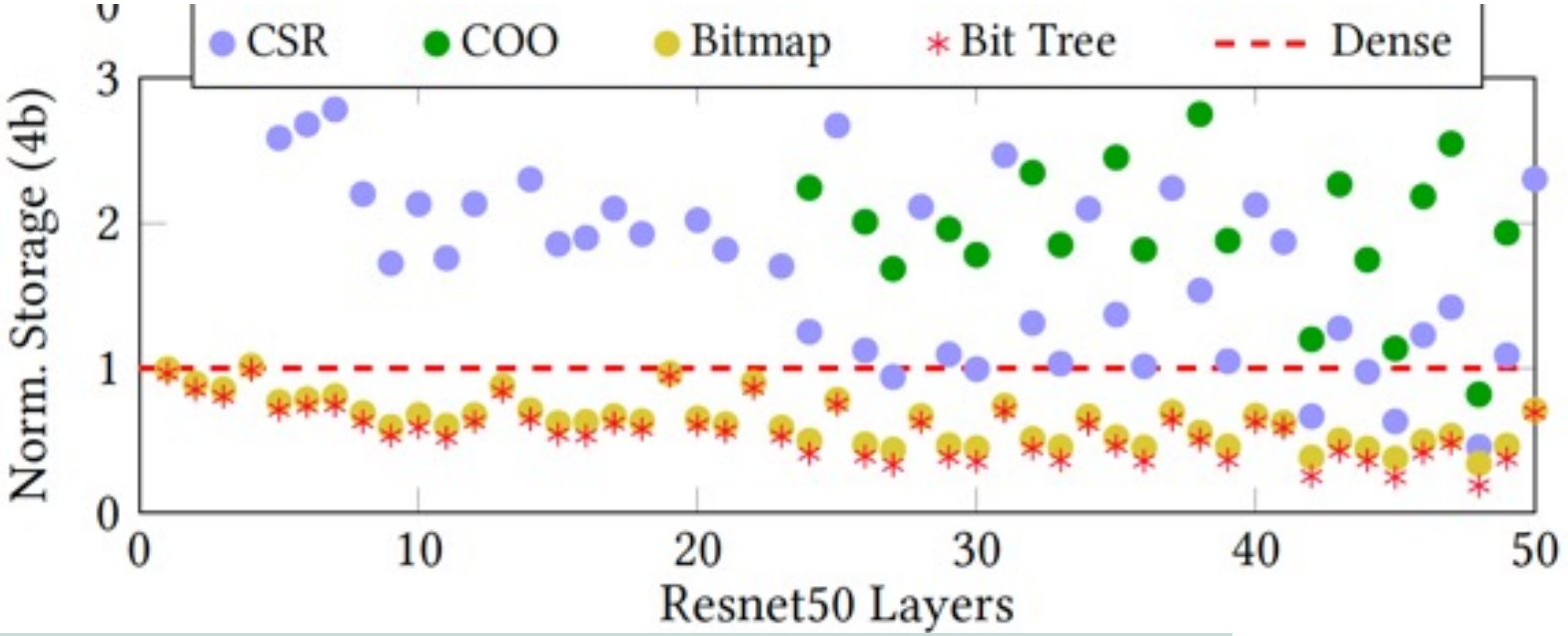




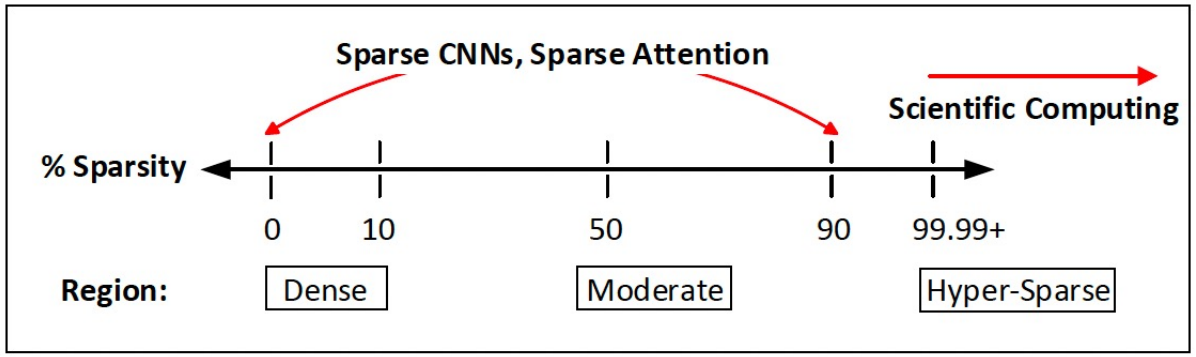
Sparse storage formats lead to inflation not compression!
Bitmap decoding is costly due to long runlength of zeros

Taming Unstructured Sparsity: Bit-Tree Format

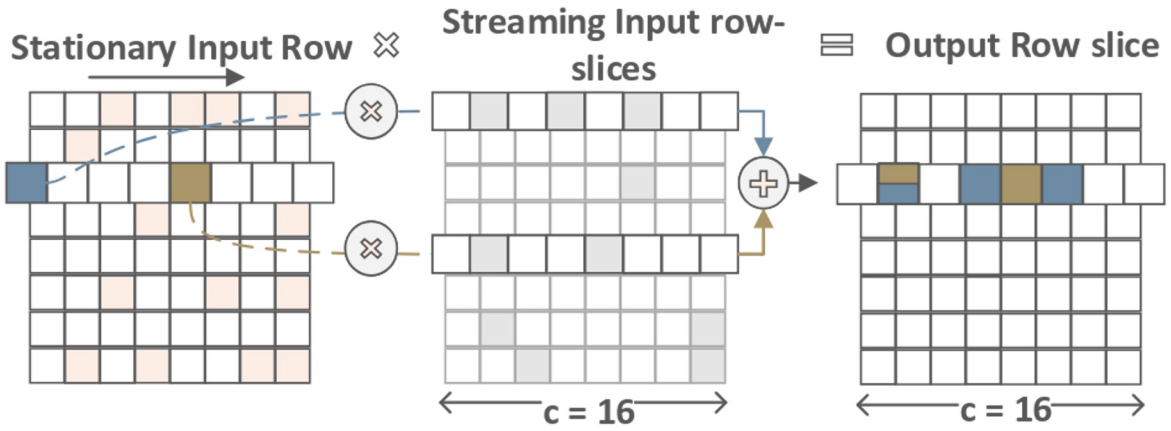




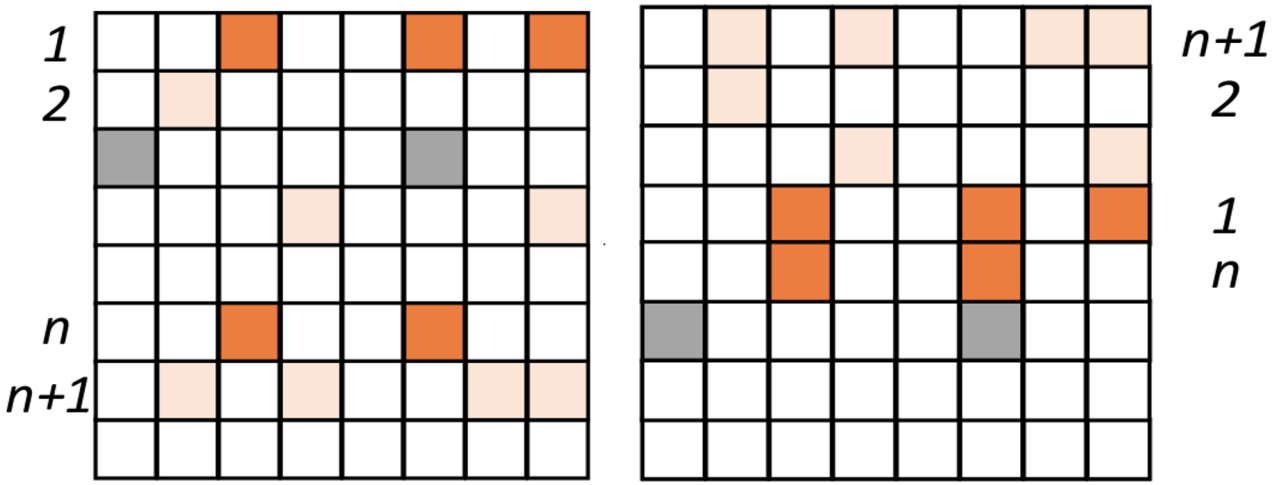
Storage: Same as ideal bitmap
Fast Decoding: N-passes over N-level Bit-Tree
Handles all sparsity level



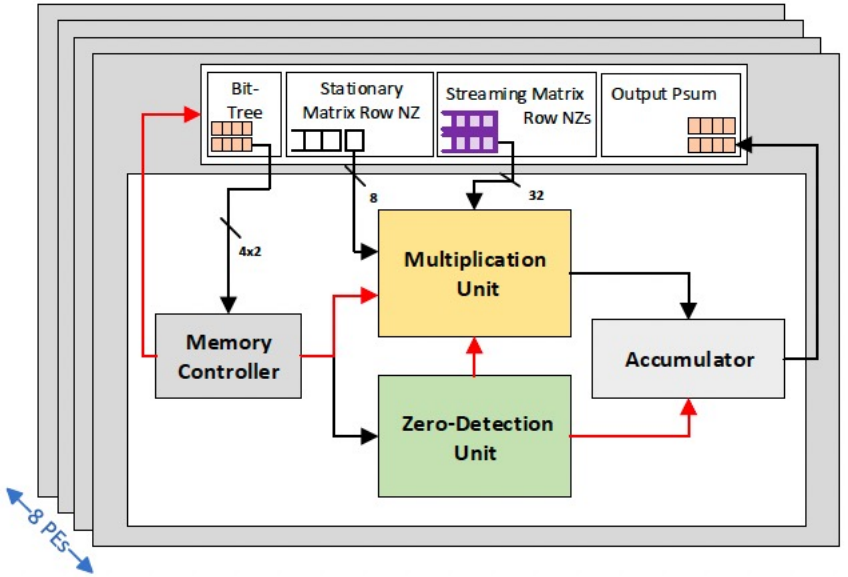
Taming Unstructured Sparsity: Irregular memory access



Gustavson's Dataflow: Row-wise product



ZeD ASIC accelerator achieves 3.2x performance-per-area improvement over SOTA for sparse ML workloads

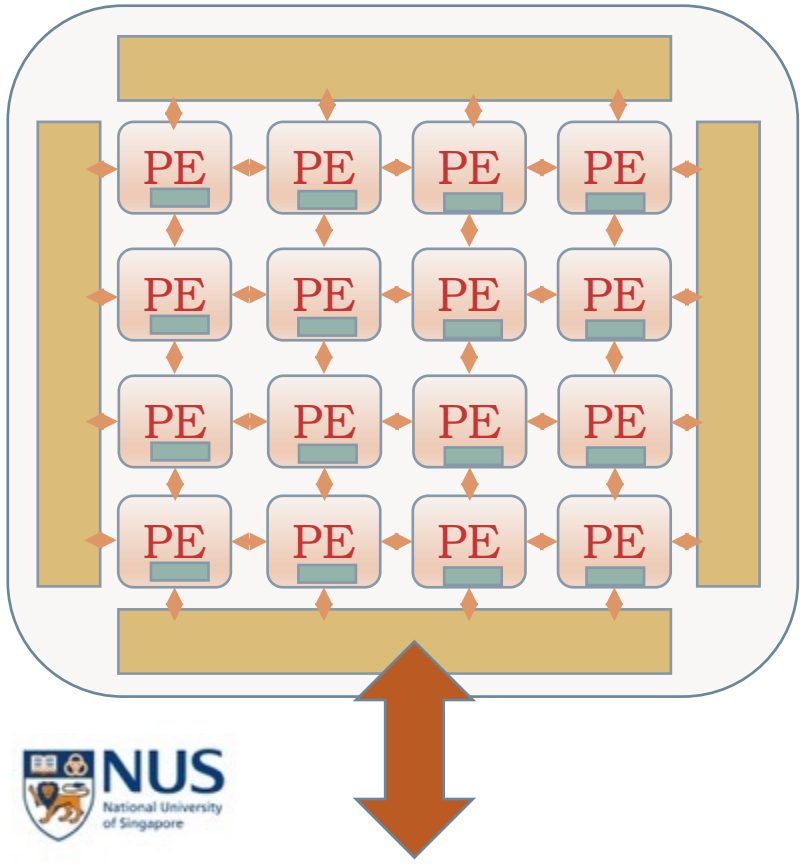


Reorder stationary matrix rows to maximally reuse streaming rows

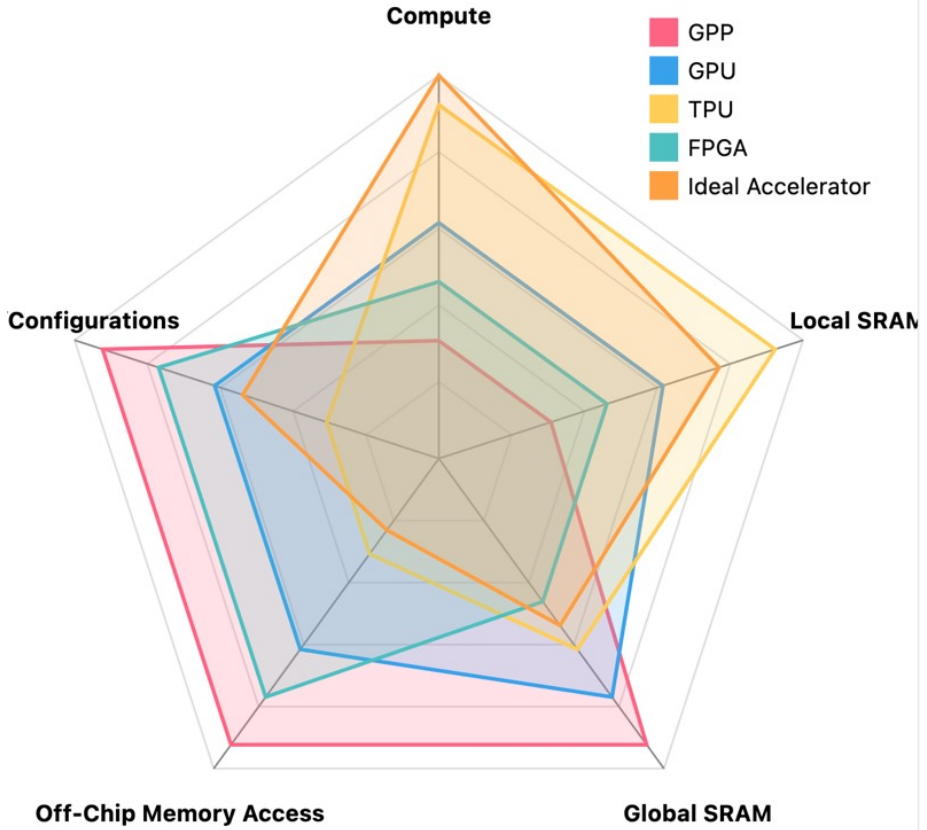
Can one architecture rule them all?

Architecture: Compute-Memory Tradeoff

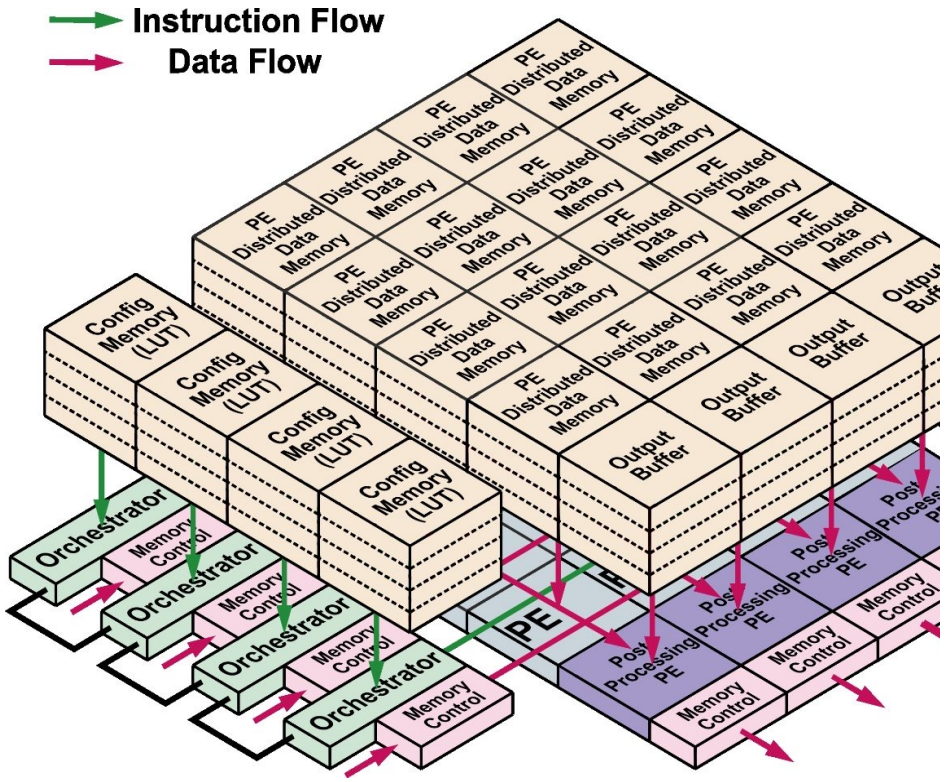
- Power budget allocation (~1W) among compute (MAC), local/global SRAM, off-chip memory access, and configurability
- As data becomes smaller (4 bit), instruction cost is very high!
- Ideal accelerator achieves performance speedup by adapting to emerging workloads with minimal instruction/configuration cost

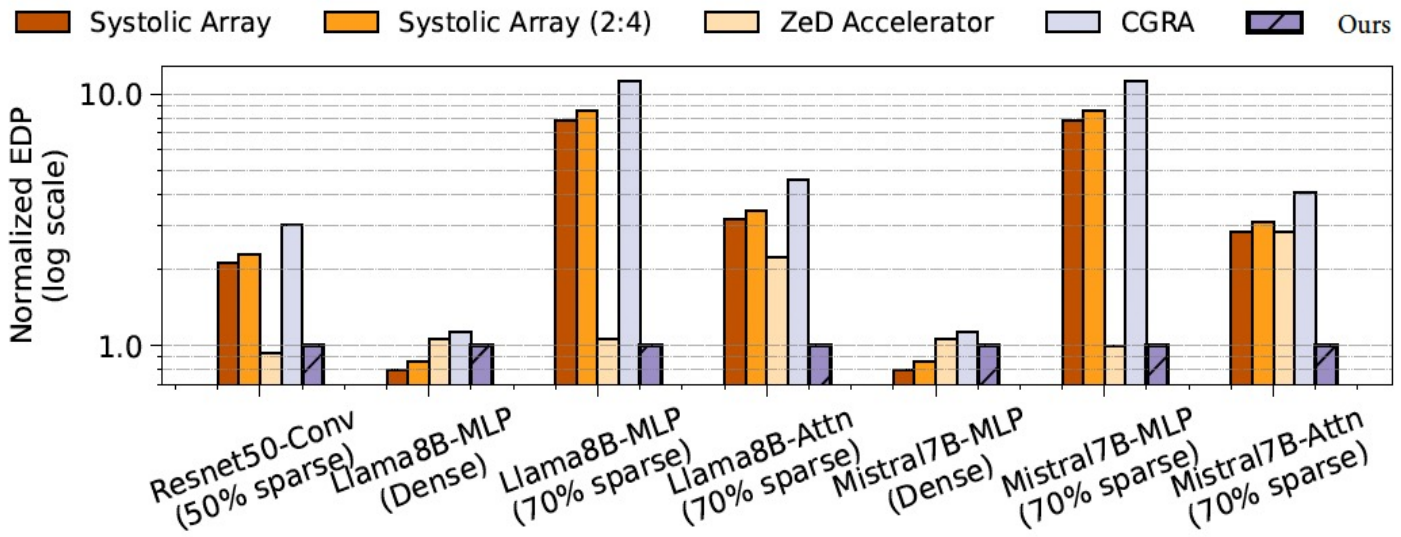


	Energy at 5nm
FP16 MAC	1 pJ
Local SRAM/word	5 pJ
Global SRAM/word	50 pJ
Off-chip memory/word	500 pJ



Canon: Ideal Sparse Transformer Accelerator





- Minimalist reconfigurable processing array that adapts to diverse sparsity with minimal reconfiguration cost
- Smart algorithm that maps arbitrary dataflows on the fabric
- High parallel processing while handling complex dataflow dependencies

Toward Efficient Intelligence: Rethinking AI Accelerators for Sparsity

- LLMs are growing faster than compute can scale. Sparsity is not optional, it's essential.
- Today's hardware is fundamentally mismatched with emerging sparse workloads.
- Specialized architectures show what's possible; but we need a universal, programmable accelerator.
- Winning the hardware lottery requires co-design: architecture must evolve with algorithms, not after them.

NUS National University of Singapore

Acknowledgments

Zhenyu Bai

Pranav Dangi

This research has been supported by the National Research Foundation, Singapore under its Competitive Research Program Award NRF-CRP23-2019-0003

GAIA: Rethinking data centre computation and communication design for Green AI

Achieve sustainable AI with energy-efficient sparse architecture
$9.2M Funding from Ministry of Education Singapore: July 2025-2030

Lead PI **Tulika Mitra**, Provost's Chair Professor, NUS Computing

Team PI **Trevor Carlson**, Assistant Professor, NUS Computing
Bingsheng He, Professor, NUS Computing
Jialin Li, Assistant Professor, NUS Computing
Li-Shiuan Peh, Provost's Chair Professor, NUS Computing

NUS National University of Singapore | Artificial Intelligence Institute

NUS National University of Singapore

Thank You