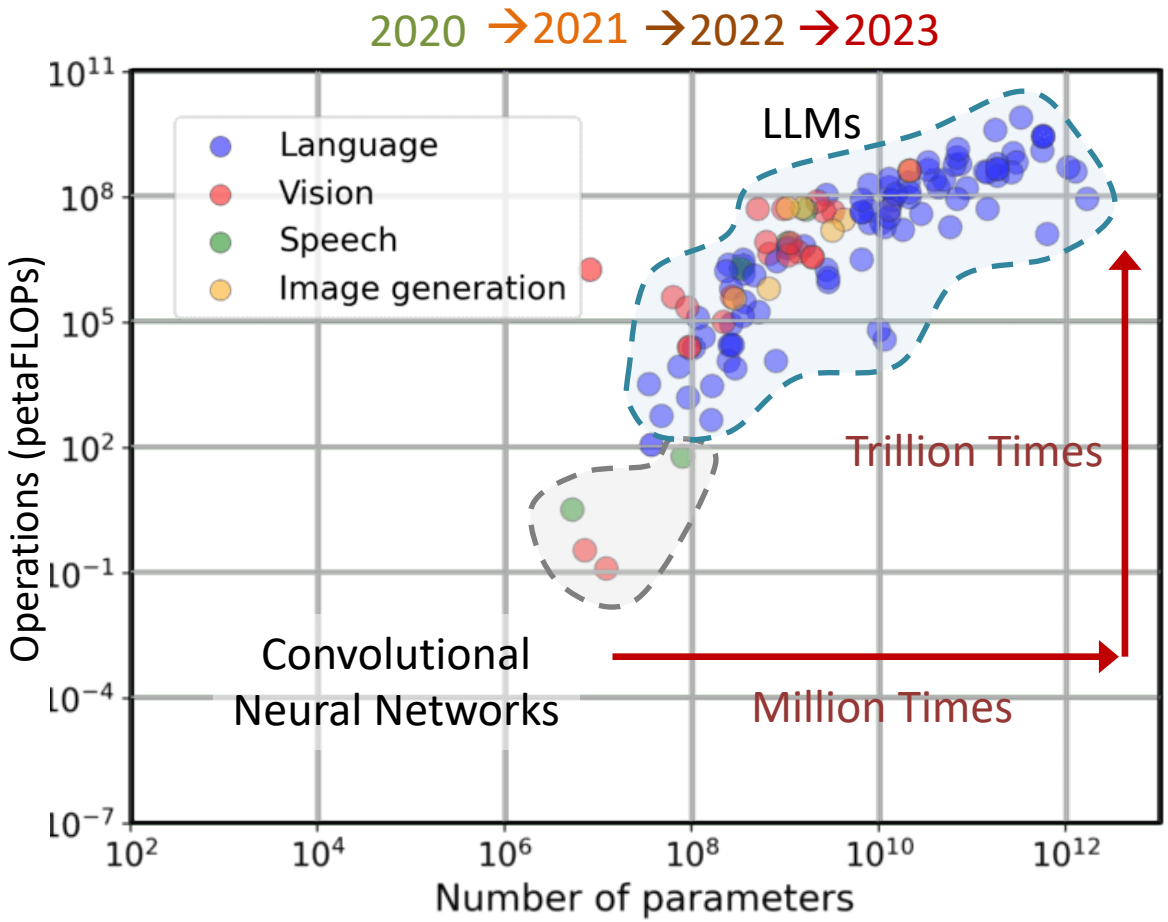
MEADOW: MEMORY-EFFICIENT DATAFLOW AND DATA PACKING FOR LOW POWER EDGE LLMS

Abhishek Moitra[1], Arkapravo Ghosh[1], Shrey Agarwal[2], Aporva Amarnath[3], Karthik Swaminathan[3], Priyadarshini Panda[1]

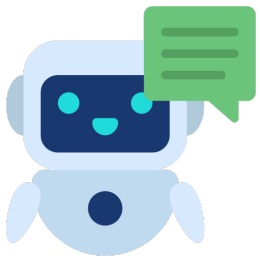
[1]Yale University, [2]IIT Roorkee, [3]IBM Research

Yale University

ScaleUp of Large Language Models



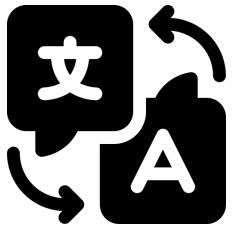
- Exponential scaleup of parameters and computation cost in Large Language Models



Chatbots

Autonomous Vehicles

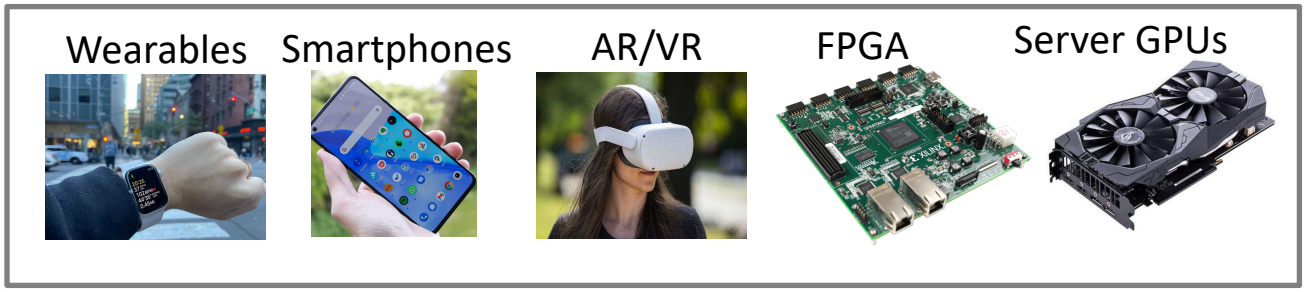
Personal Assistants



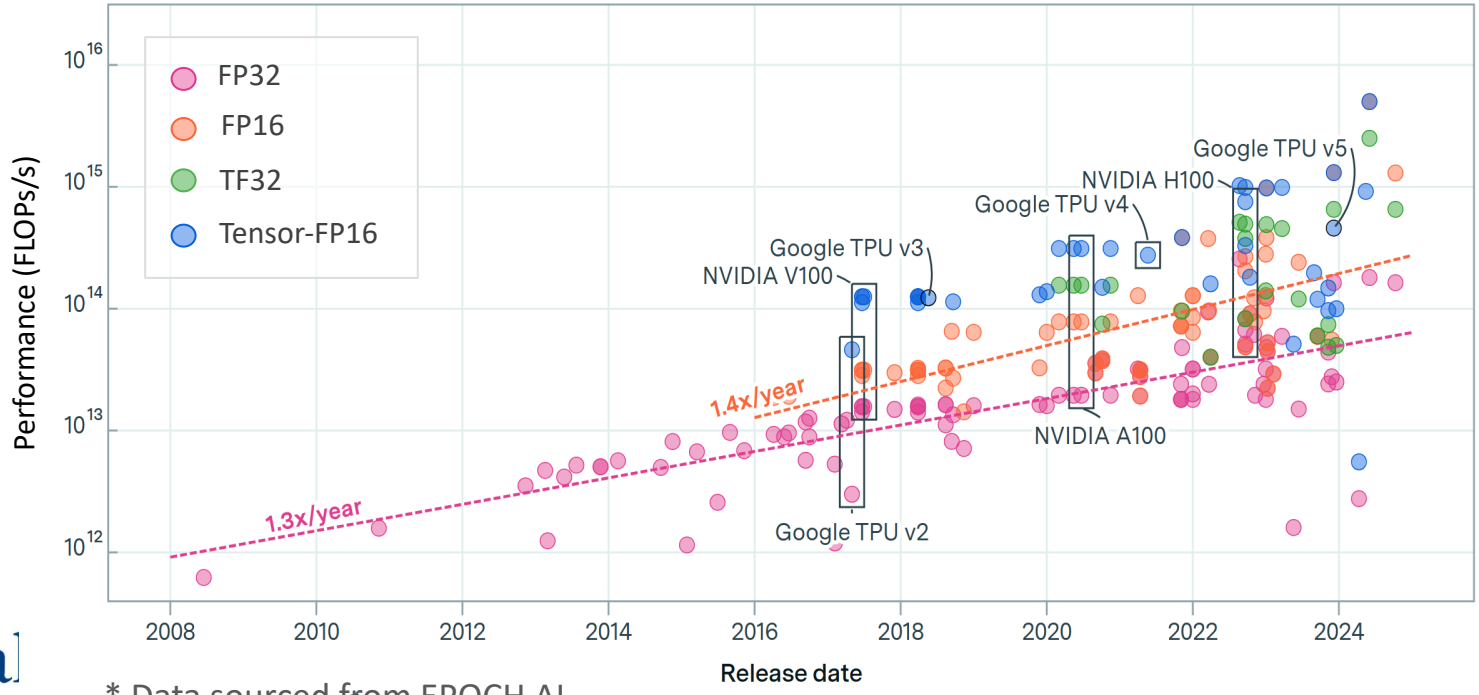
Real-time Translation

Yale University

Hardware Deployment Platforms Today



Widescale AI Deployment Platforms

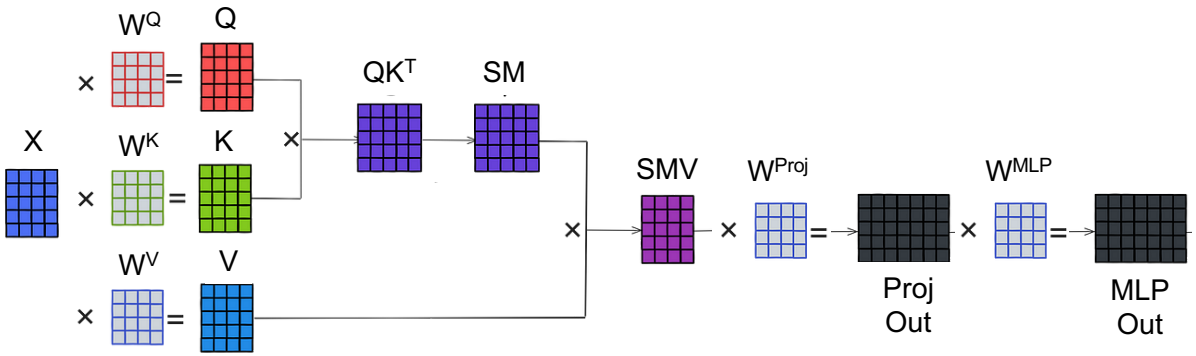


* Data sourced from EPOCH AI

What Happens when an LLM gets a Query?

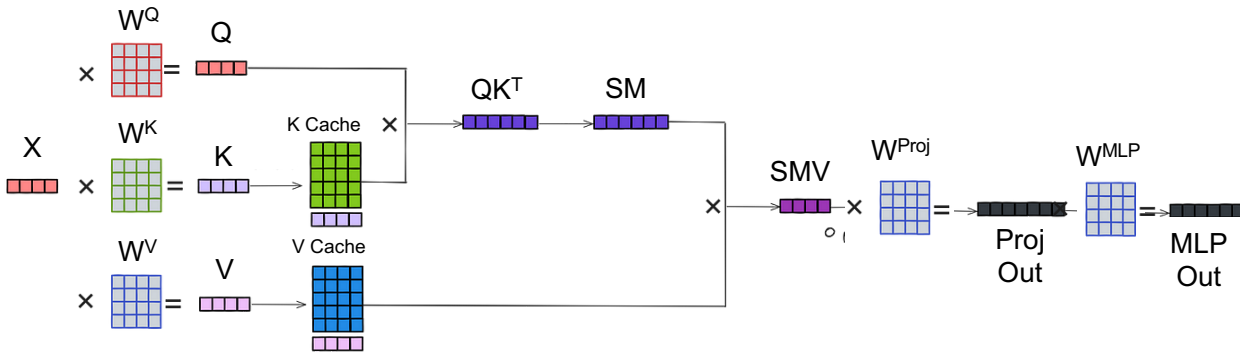
Yale University

What Happens when an LLM gets a Query?



Prefill Stage

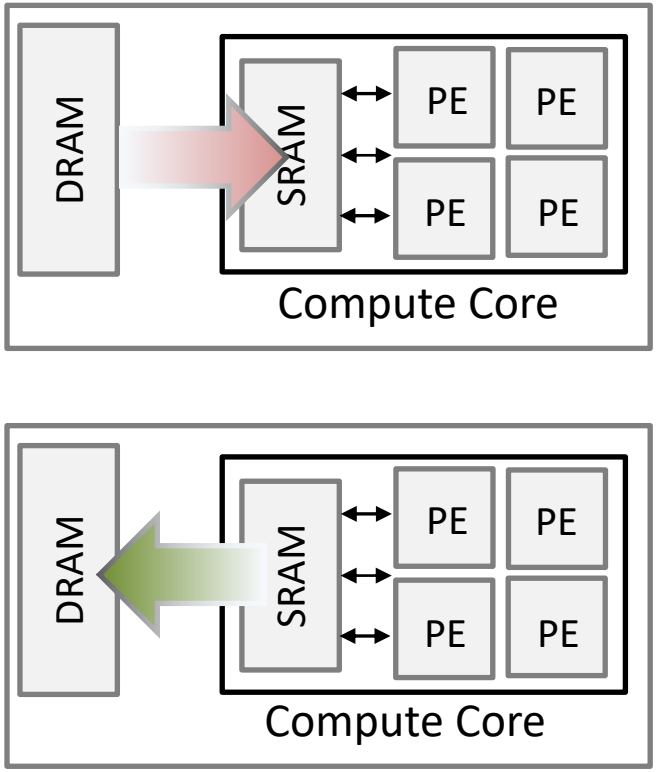
- Large number of input tokens
- Contextual Understanding

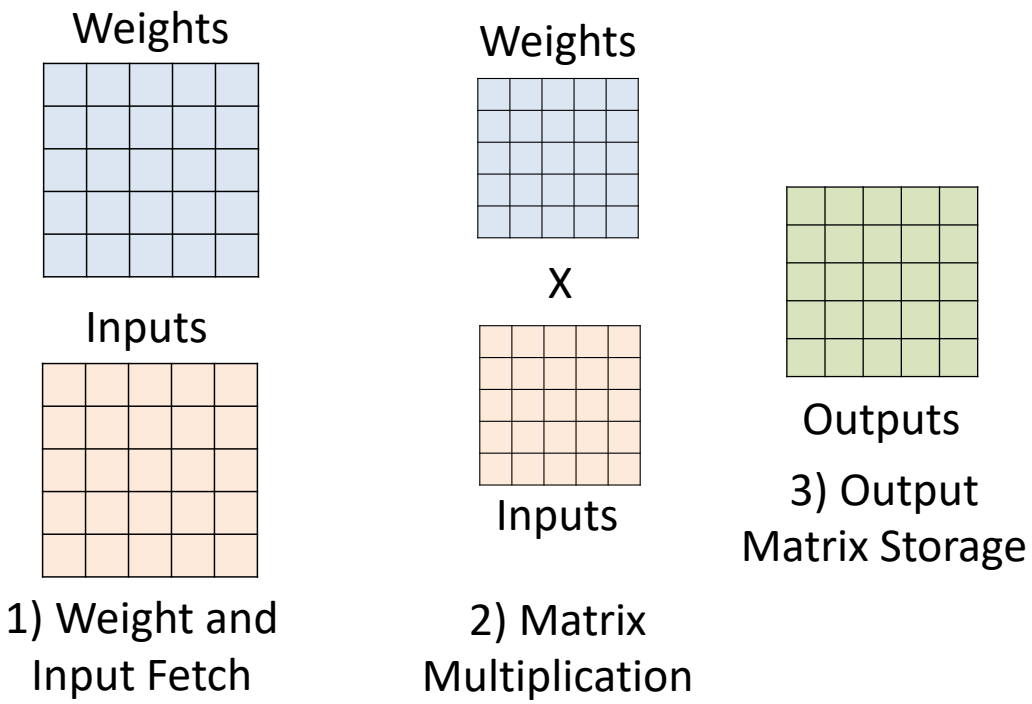


Decode Stage

- Tokens generated one-by-one

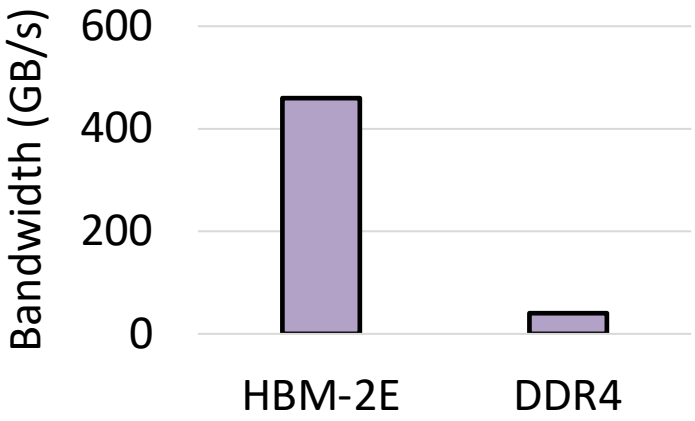
GEMM Dataflow and HBM-based GPUs

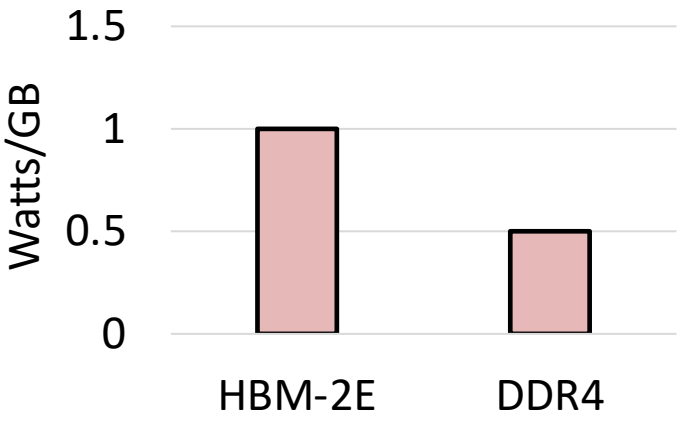




Outputs are fetched and stored on and off DRAM chip during attention computation

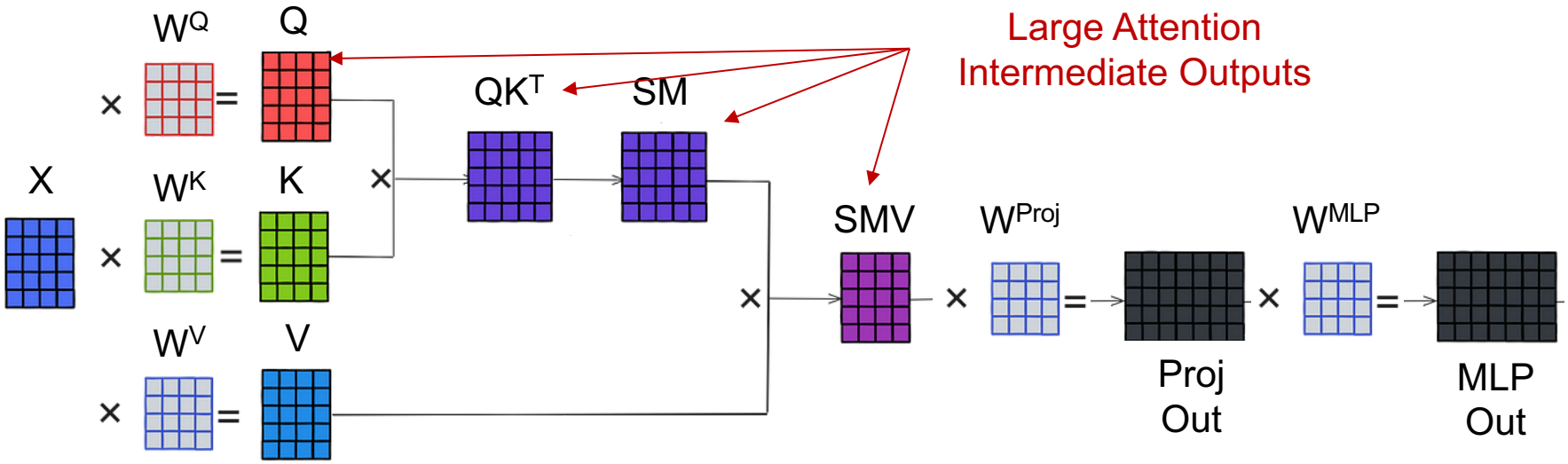
High Bandwidth Memory to Overcome Bottleneck

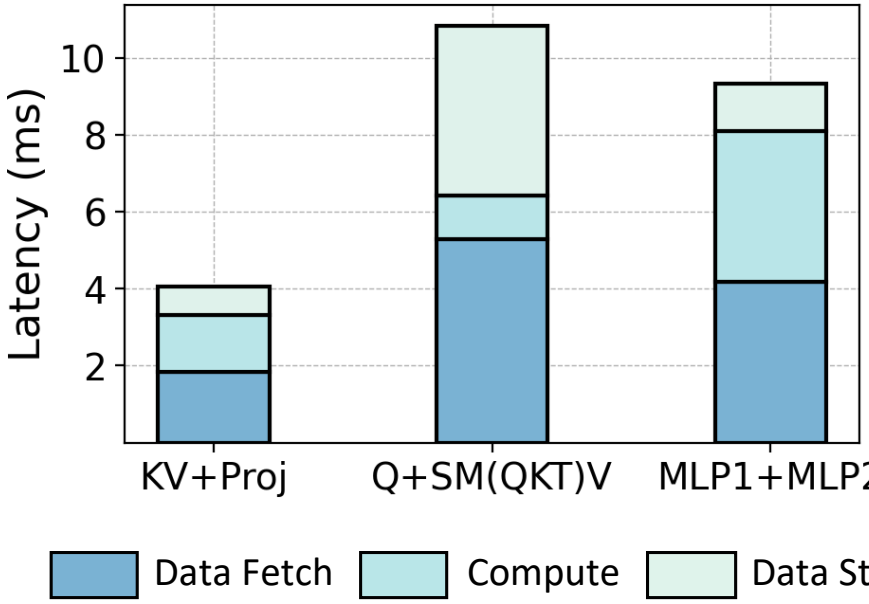




- HBM overcomes memory bottleneck through high bandwidth
- Consumes 2x power/GB of storage compared to standard DDR4 DRAMs
- Low power embedded devices still have a strict power constraint < 10 W

Memory Bottleneck During the Prefill Stage



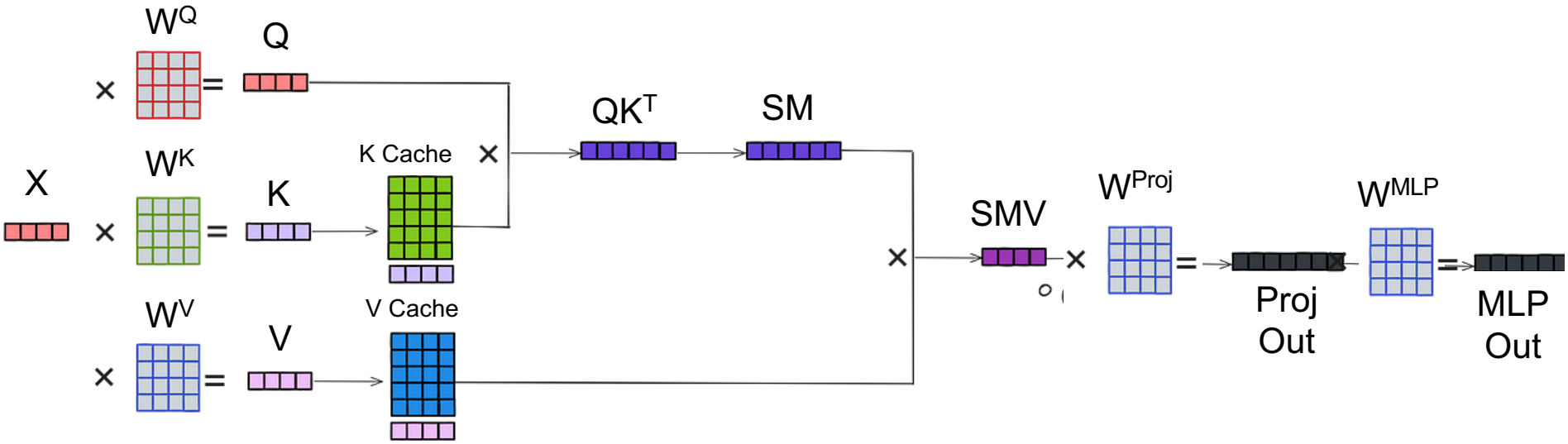


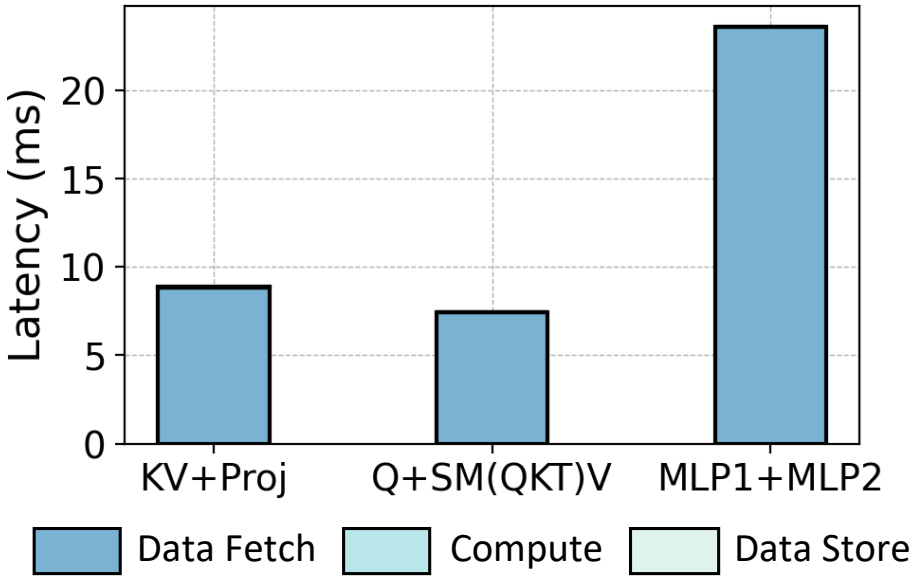
Dominant Components:

- Fetch and Storage of Intermediate Outputs
- Weight Fetches
- Compute

Moitra, Abhishek, et al. "MEADOW: Memory-efficient Dataflow and Data Packing for Low Power Edge LLMs." *arXiv preprint arXiv:2503.11663* (2025).

Memory Bottleneck During the Decode Stage



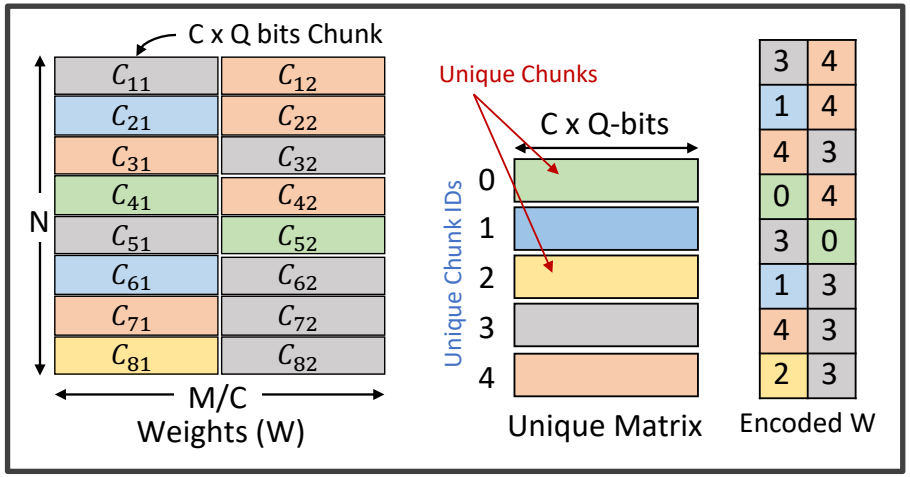


Dominant Components:

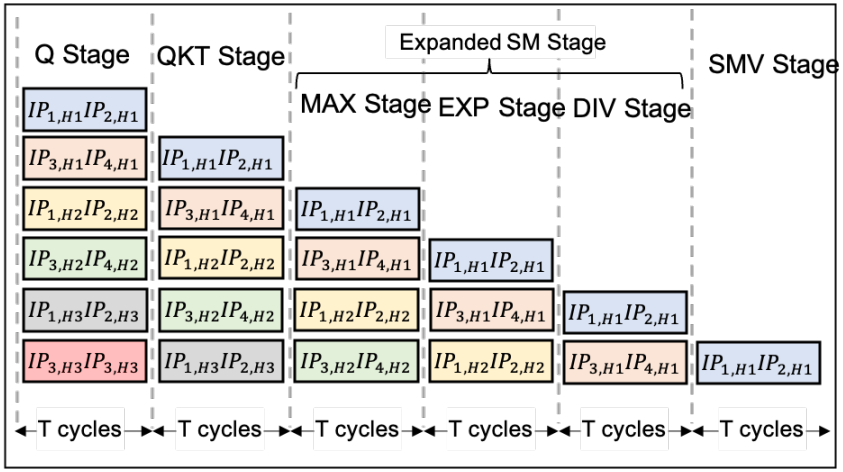
- Weight Fetches
- KV Cache Fetch

Moitra, Abhishek, et al. "MEADOW: Memory-efficient Dataflow and Data Packing for Low Power Edge LLMs." *arXiv preprint arXiv:2503.11663* (2025).

MEADOW: Dataflow and Compression Strategy for Memory-bound LLM



Weight Packing



Token-parallel Head Sequential Dataflow

- Decomposes large weight matrices to smaller unique chunks
- Loss-less conversion
- Reduces weight fetch latency during prefill and decode stages

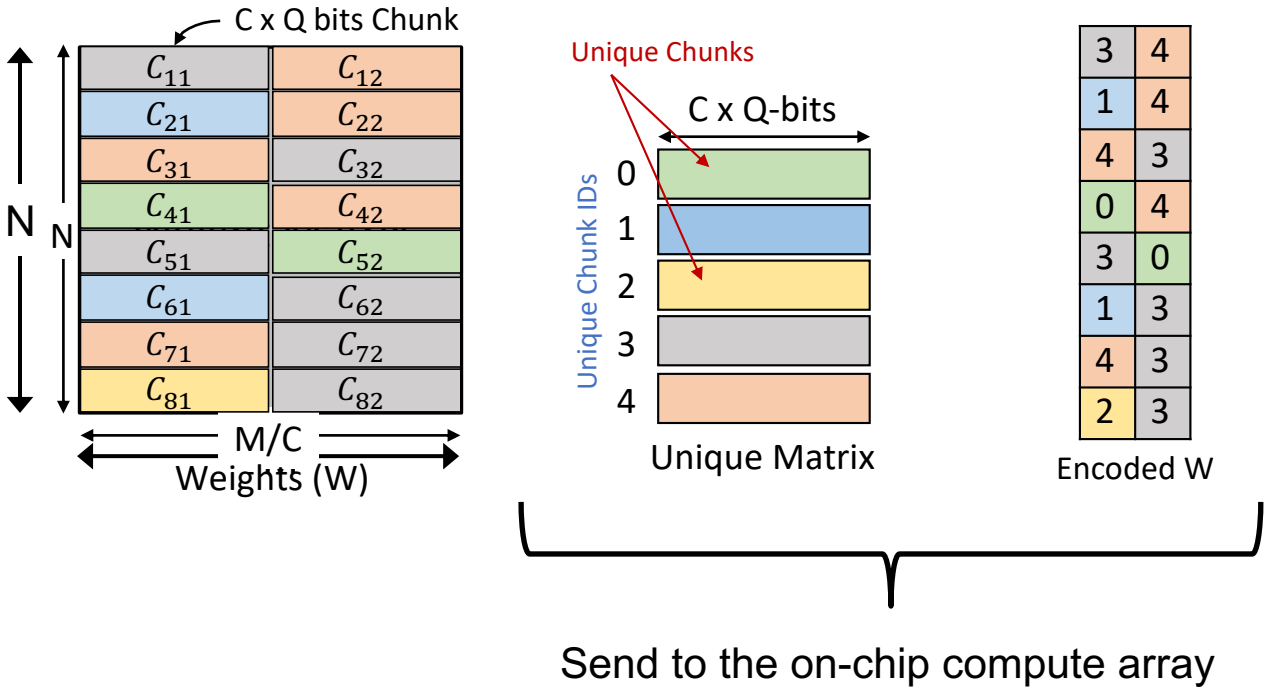
- Pipelines the attention layers
- Prevents data fetch and store latency during Prefill

*In Collaboration with IBM Research

Yale University

Moitra, Abhishek, et al. "MEADOW: Memory-efficient Dataflow and Data Packing for Low Power Edge LLMs." *arXiv preprint arXiv:2503.11663* (2025).

Loss-less Weight Packing

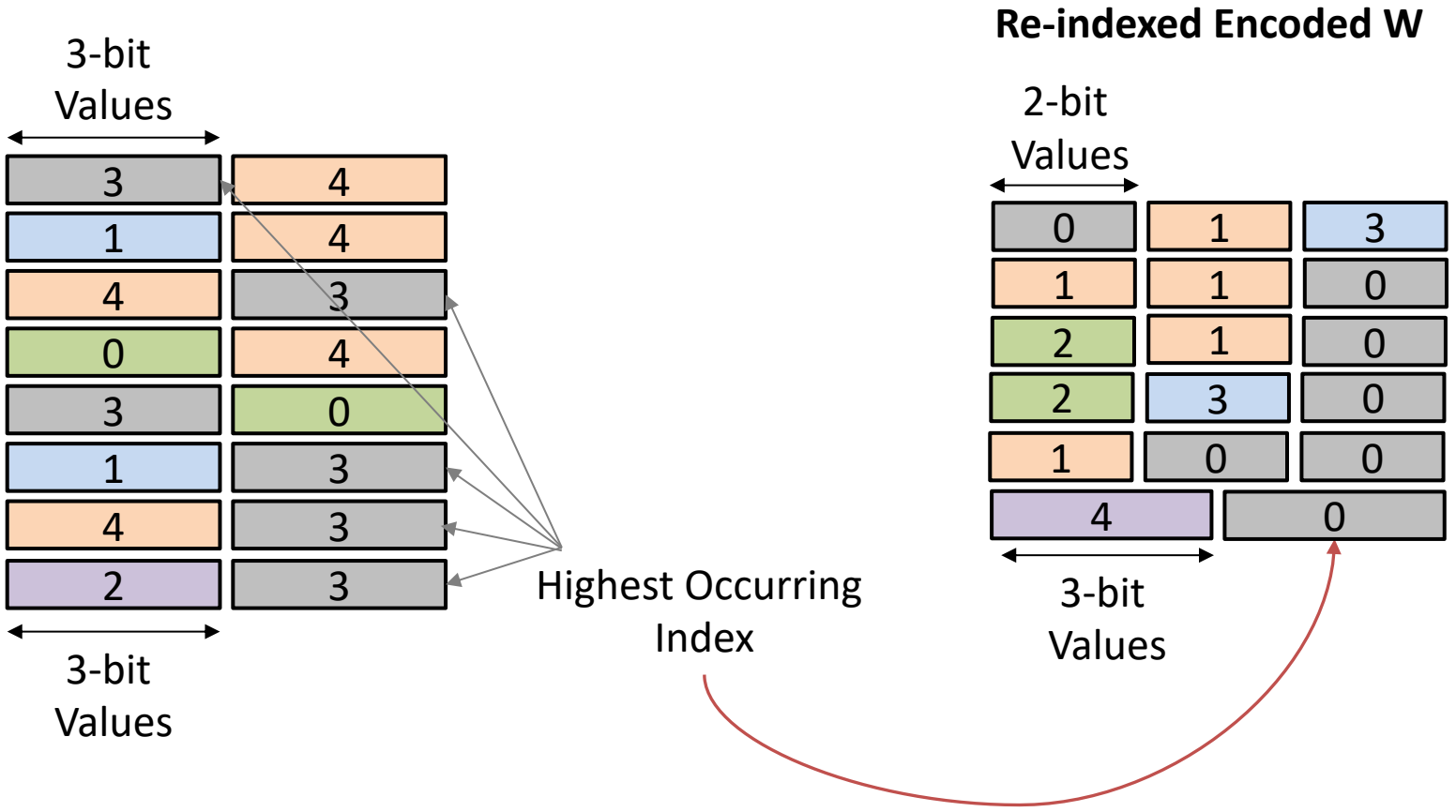


- Divides a weight matrix into chunks
- Encoded W → Weight Matrix encoded in terms of the unique chunk IDs
- Unique Matrix + Encoded W transferred from DRAM to On-chip Compute

Yale University

Moitra, Abhishek, et al. "MEADOW: Memory-efficient Dataflow and Data Packing for Low Power Edge LLMs." *arXiv preprint arXiv:2503.11663* (2025).

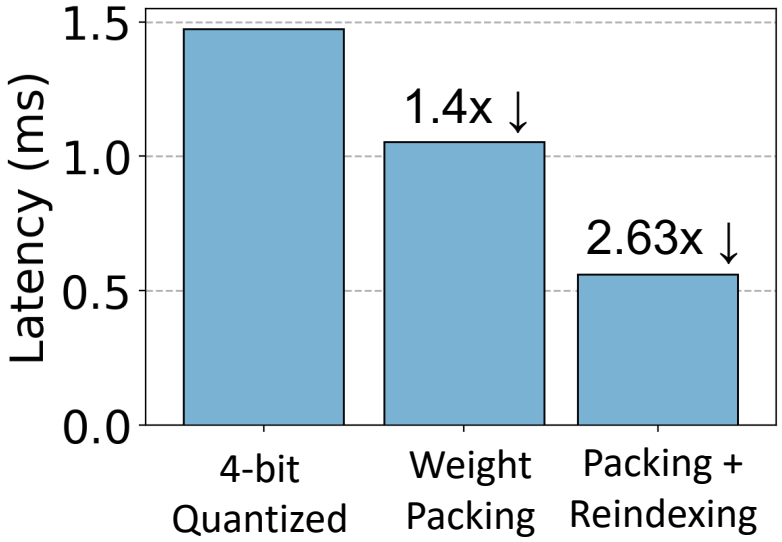
Frequency-aware Reindexing to Further Increase Packing Efficiency



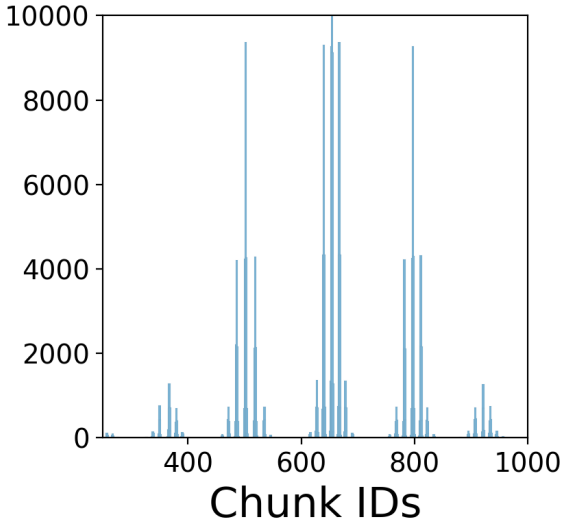
Yale University

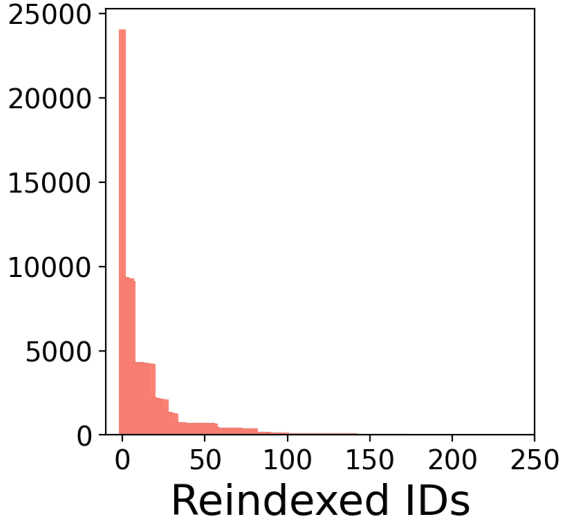
Moitra, Abhishek, et al. "MEADOW: Memory-efficient Dataflow and Data Packing for Low Power Edge LLMs." *arXiv preprint arXiv:2503.11663* (2025).

Compression Benefits for OPT-125M LLM



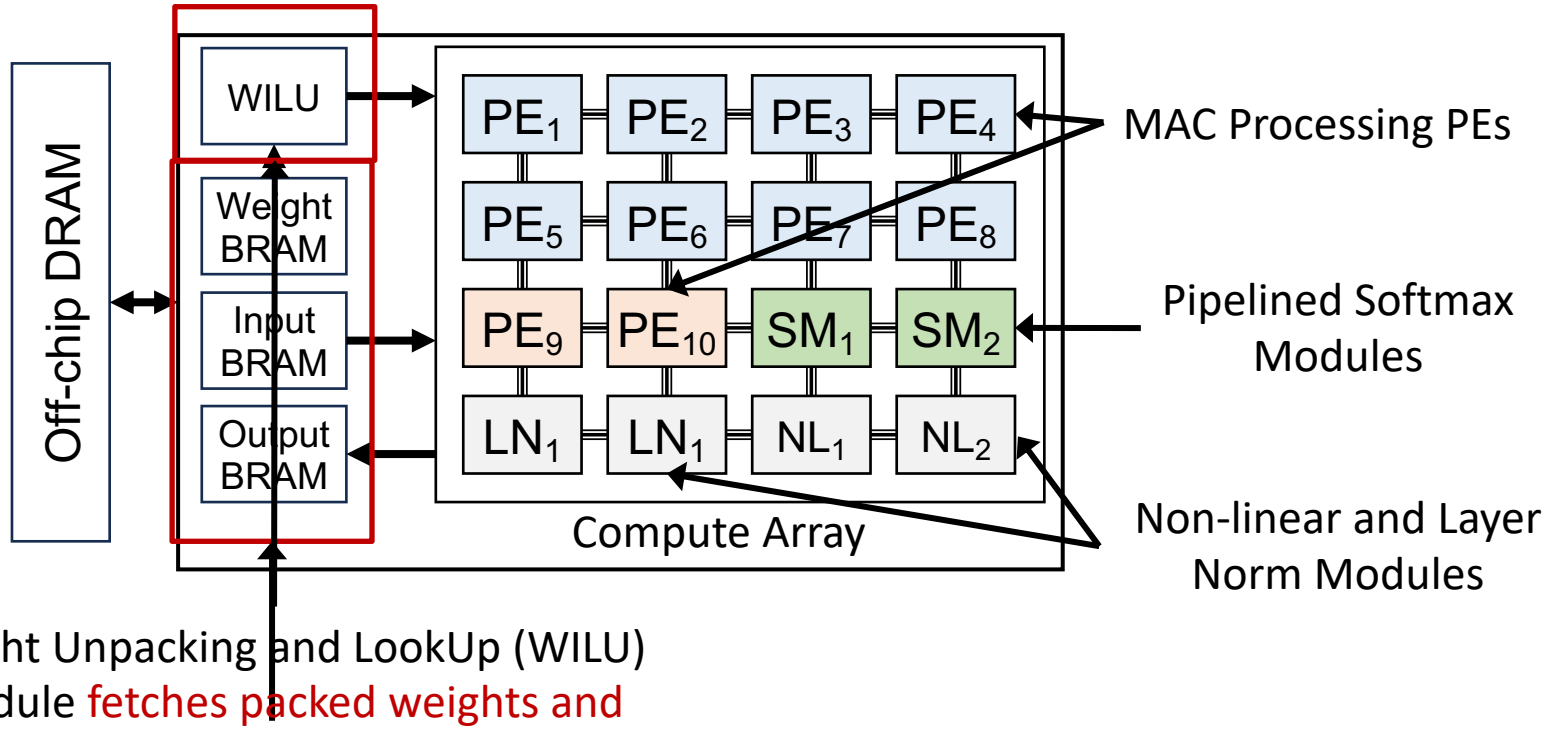
* Latency for 1 decoder





Yale Un

Architecture of MEADOW



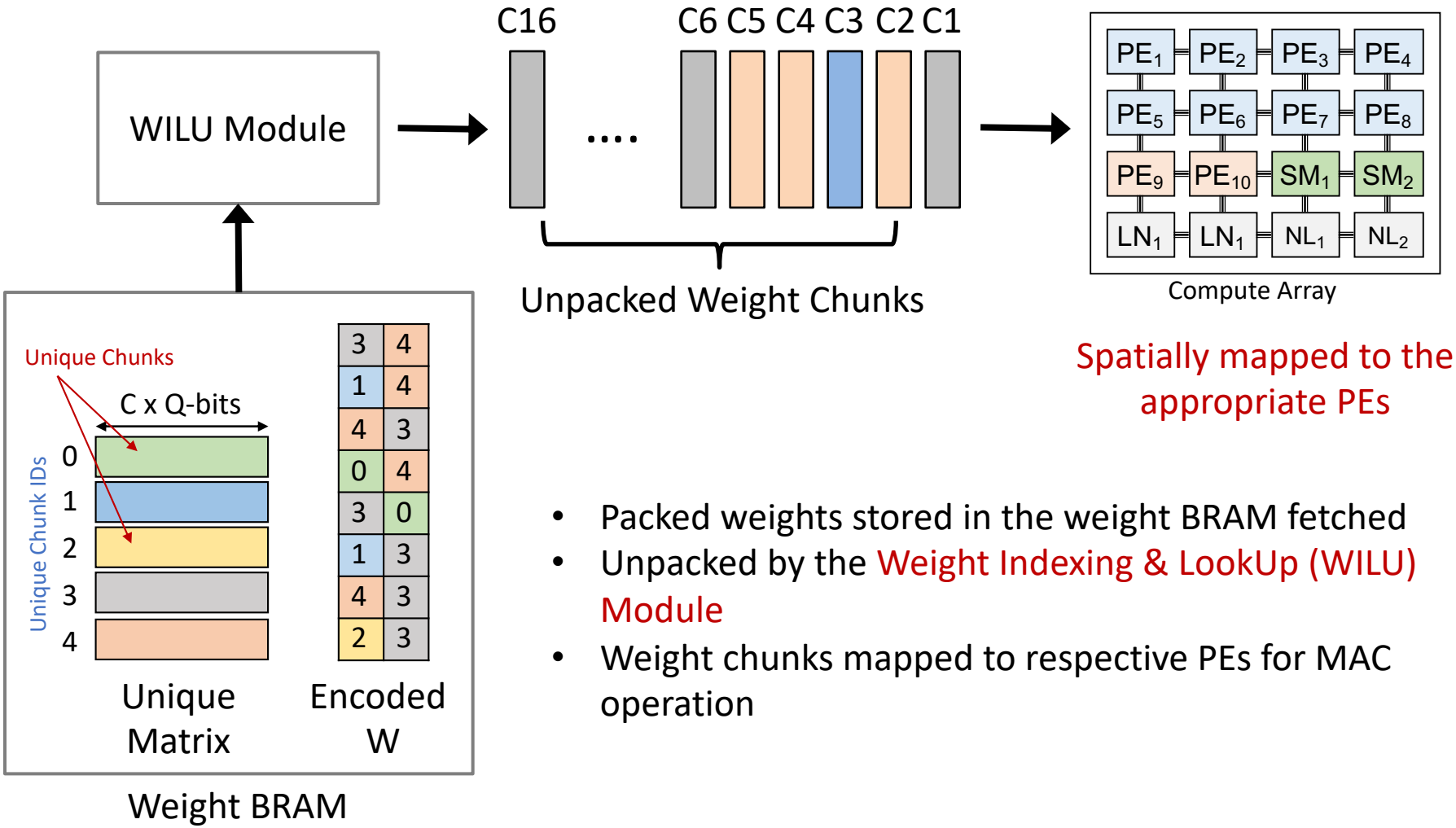
Weight Unpacking and LookUp (WILU) Module fetches packed weights and performs the decompression

- Inputs and Outputs are stored on BRAMs
- Weight BRAM contains the encoded weights and unique matrices

Yale University

Moitra, Abhishek, et al. "MEADOW: Memory-efficient Dataflow and Data Packing for Low Power Edge LLMs." *arXiv preprint arXiv:2503.11663* (2025).

Unpacking During Weight Fetch from Weight BRAM

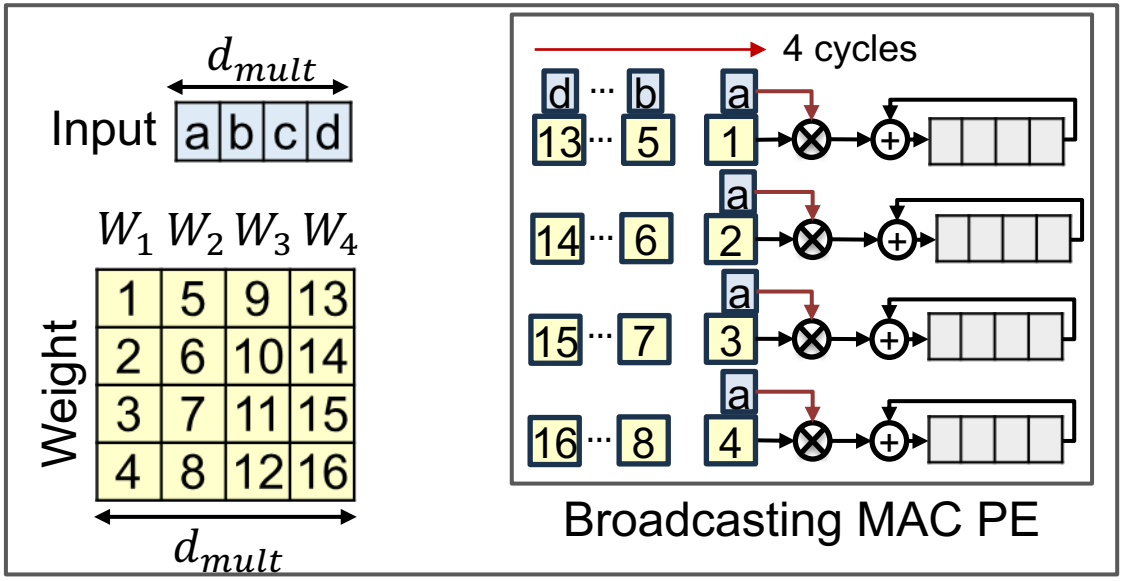


- Packed weights stored in the weight BRAM fetched
- Unpacked by the Weight Indexing & LookUp (WILU) Module
- Weight chunks mapped to respective PEs for MAC operation

Yale University

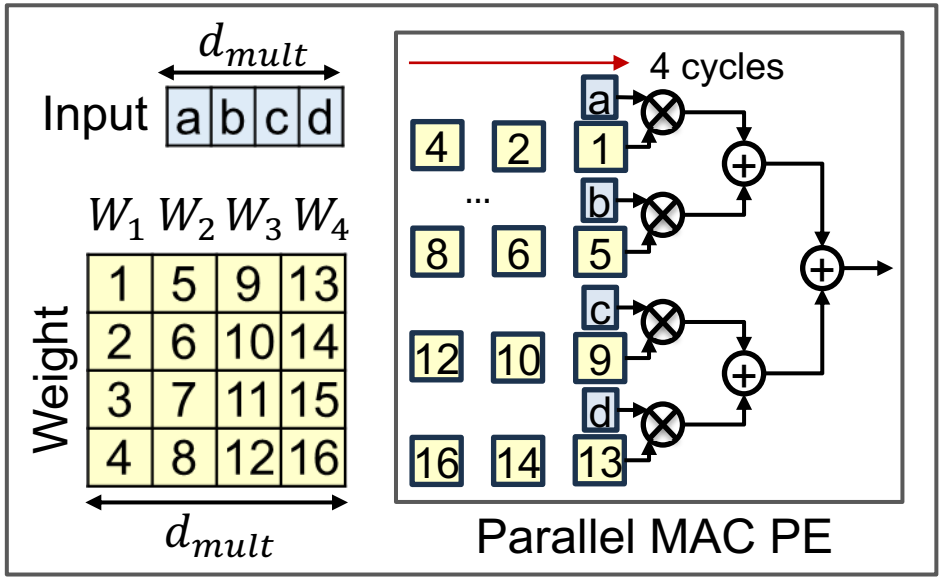
Moitra, Abhishek, et al. "MEADOW: Memory-efficient Dataflow and Data Packing for Low Power Edge LLMs." *arXiv preprint arXiv:2503.11663* (2025).

Parallel and Broadcasting MAC Units



Broadcasting PEs

- Useful for SMxV operation
- Parallelization across output channel
- Multiply-accumulate over multiple cycles

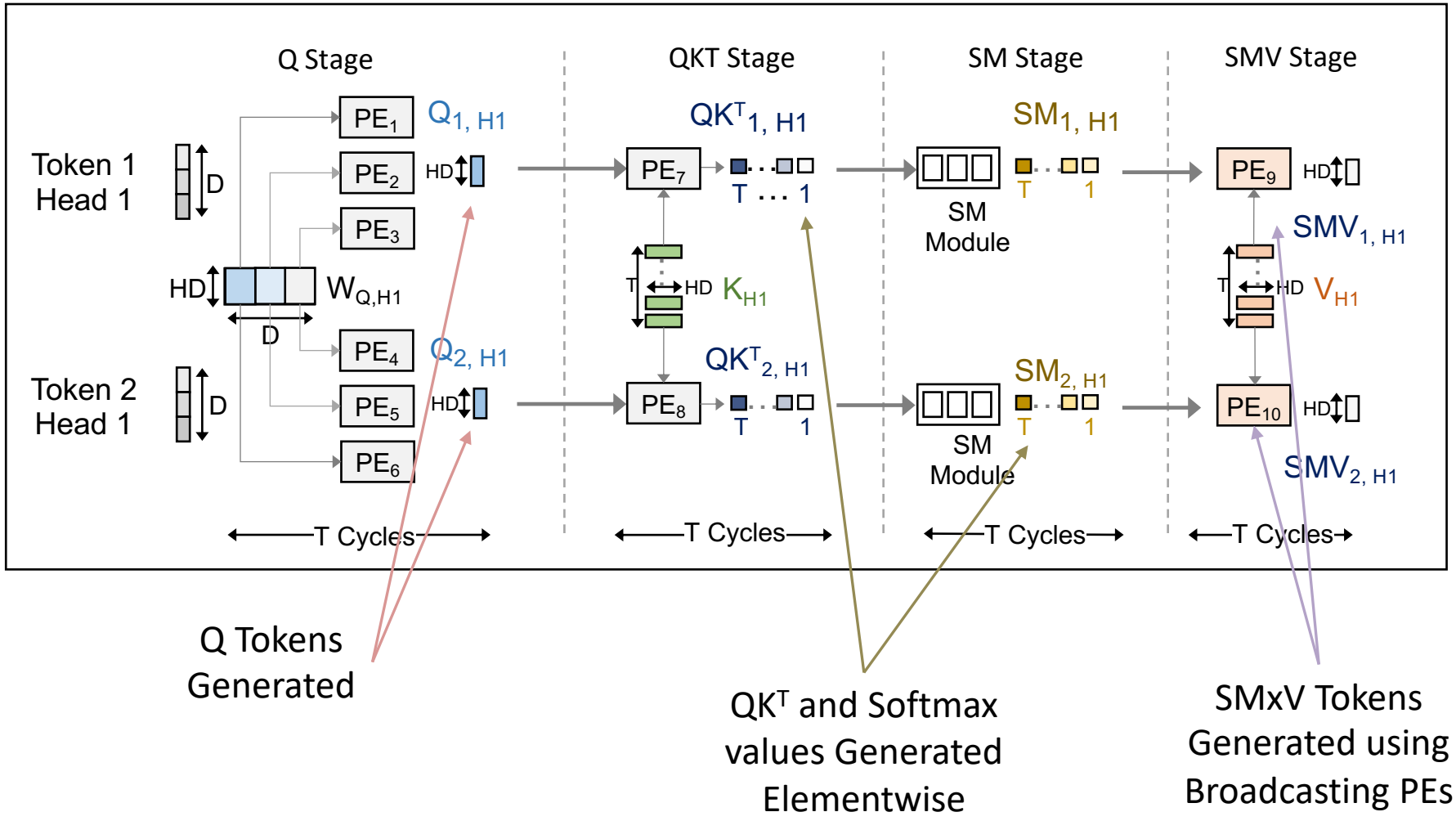


Parallel PE

- Used for all other operations
- Prallelization across input channel
- Multiplication and Accumulation in one cycle

Moitra, Abhishek, et al. "MEADOW: Memory-efficient Dataflow and Data Packing for Low Power Edge LLMs." *arXiv preprint arXiv:2503.11663* (2025).

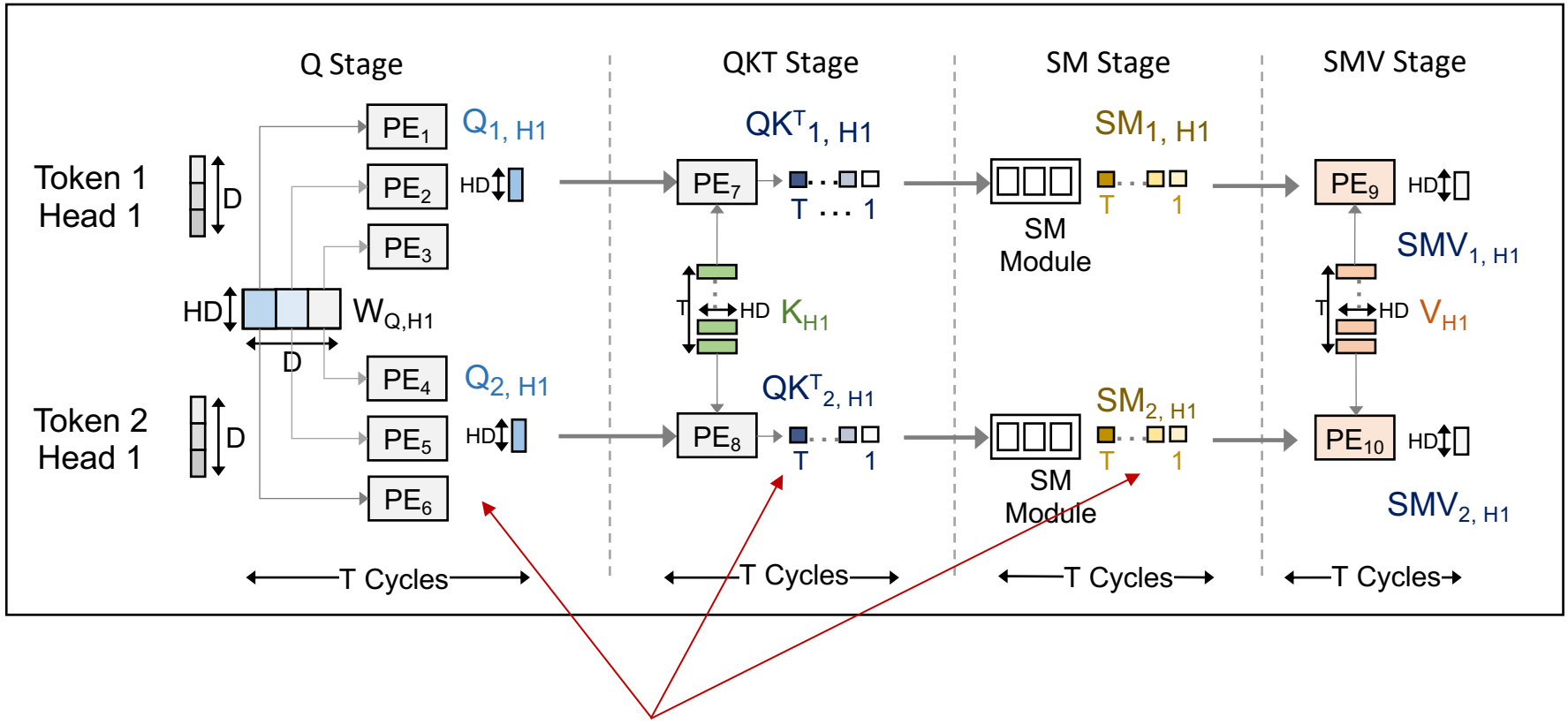
Token Parallel Head Sequential Dataflow



Yale University

Moitra, Abhishek, et al. "MEADOW: Memory-efficient Dataflow and Data Packing for Low Power Edge LLMs." *arXiv preprint arXiv:2503.11663* (2025).

Token Parallel Head Sequential Dataflow



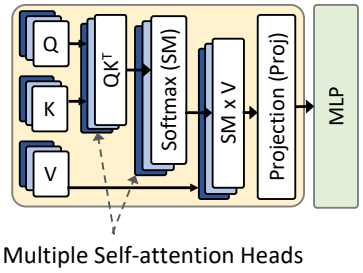
Data generated by one layer is utilized by the next layer

Prevents data fetch and store latency during Prefill

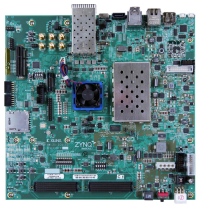
Yale University

Moitra, Abhishek, et al. "MEADOW: Memory-efficient Dataflow and Data Packing for Low Power Edge LLMs." *arXiv preprint arXiv:2503.11663* (2025).

MEADOW Optimization Framework



LLM Parameters
#Tokens
#Attention Heads
Hidden Dimension
MLP Ratio



FPGA Parameters
#DSPs
DRAM Bandwidth
SRAM Capacity
#LUTs

- Optimize Latency
- Heuristic Design Space Search

MEADOW Framework

TPHS Parameters
PE Allocation
Pipelining Factor
PE Architecture

Weight Packing Parameters
Chunk Size
Packing Precision

Yale University

Moitra, Abhishek, et al. "MEADOW: Memory-efficient Dataflow and Data Packing for Low Power Edge LLMs." *arXiv preprint arXiv:2503.11663* (2025).

Experiments and Results

Yale University

Hardware Implementation Parameters

LLM Models	OPT-125M and OPT-1.3B
Dataset - LAMBADA dataset	Finetuned with Smoothquant [1] (8-bit Weights and Activations)

Parameter	Value
FPGA Board	Xilinx ZCU102
LUTs	600K
DSPs	2520
BRAM	32.1Mb

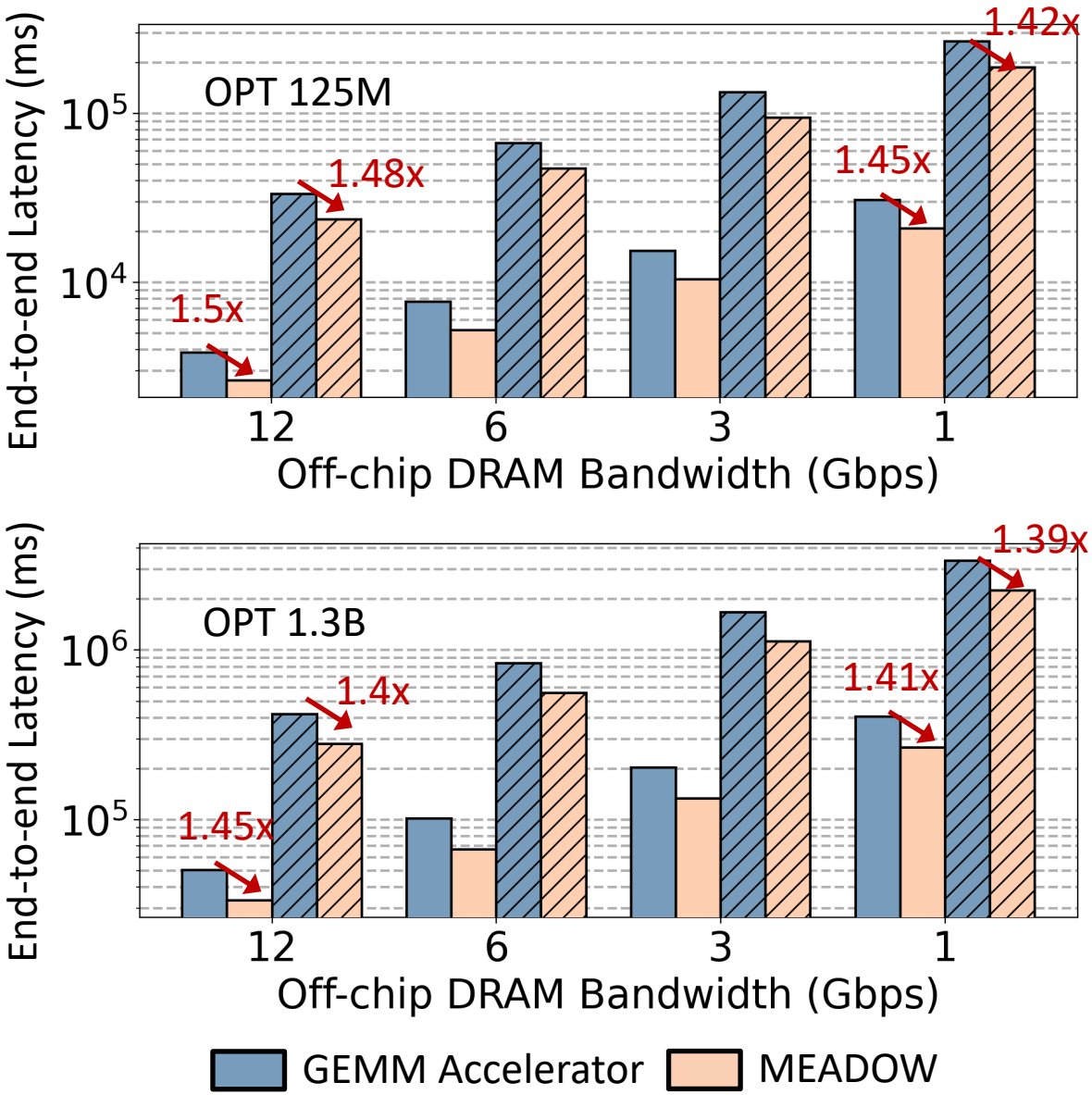
Baseline: LLM implemented with GEMM dataflow with 8-bit weights without weight packing

MEADOW Implementation: LLM with TPHS dataflow and 8-bit weights with weight packing

Yale University

[1] Xiao, Guangxuan, et al. "Smoothquant: Accurate and efficient post-training quantization for large language models." *International Conference on Machine Learning*. PMLR, 2023.

End-to-end Latency Reduction with MEADOW



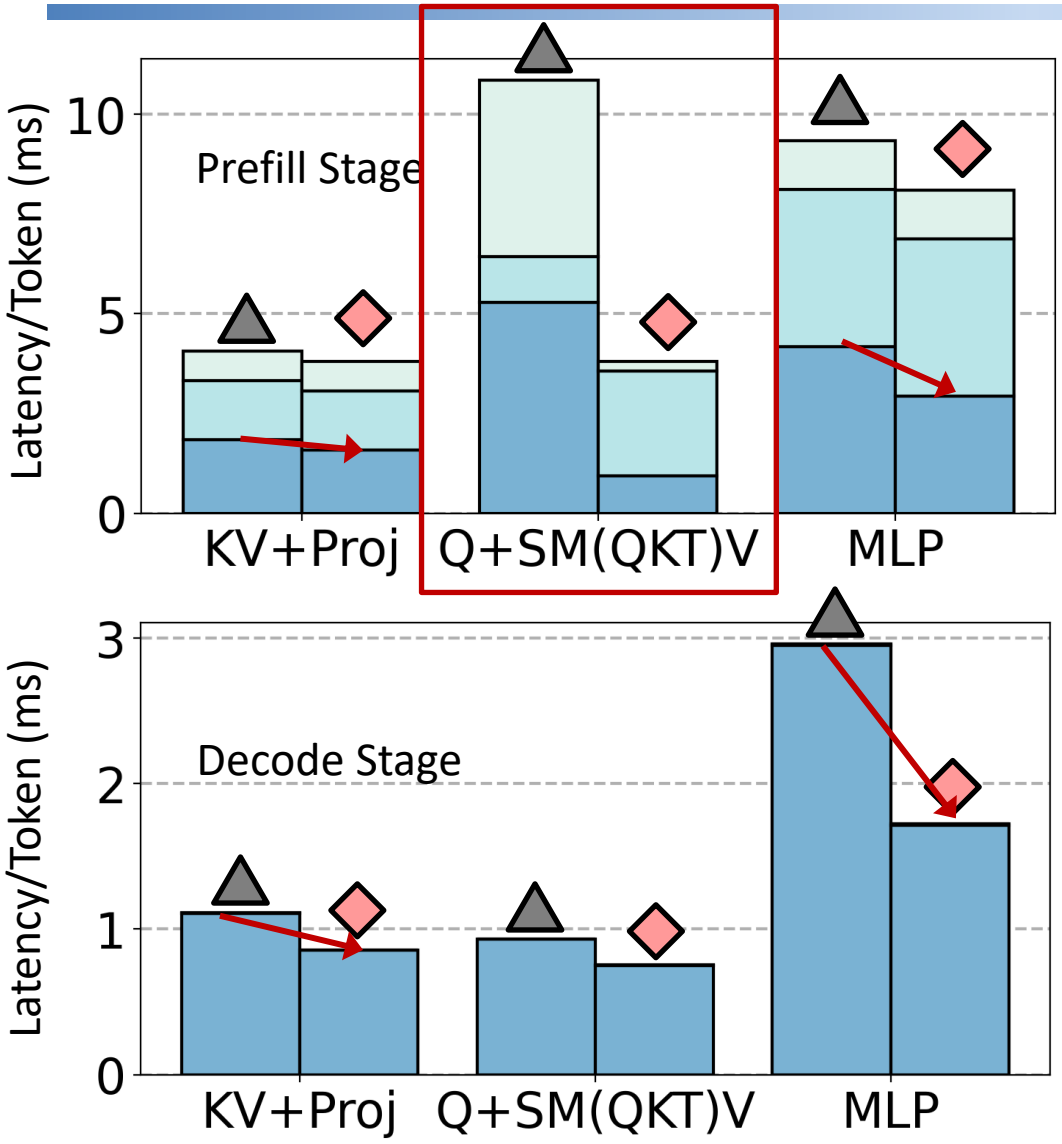
- MEADOW Lowers End-to-end Latency by 1.5x for OPT 125M and OPT 1.3B LLMs compared to GEMM-based Acceleration

*Results based on Xilinx ZCU102 FPGA (10W) at different DRAM Bandwidths

Prefill = 512 Decode = 64

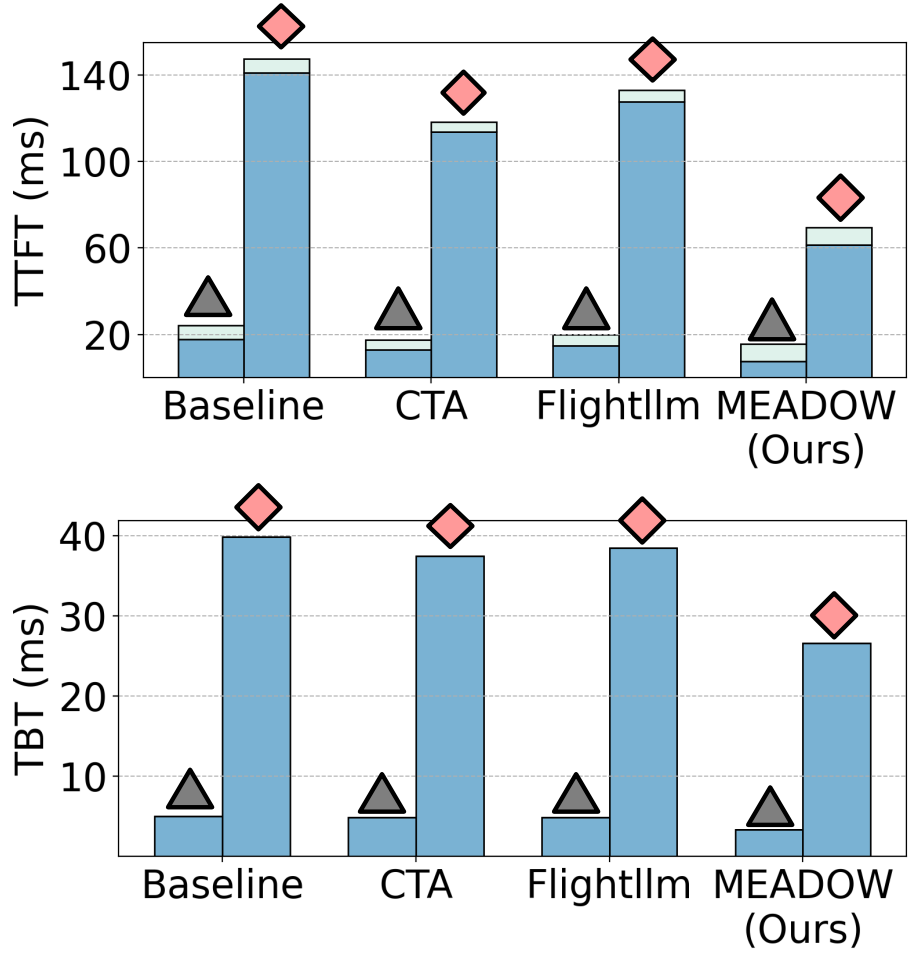
Prefill = 512 Decode = 512

Understanding the Latency Reduction



- Attention computation latency reduction by 2.5x
- TPHS Dataflow reduces data fetch & store latency by pipelining attention
- Weight Packing reduces weight data fetch latency by 1.8x

Comparison with Prior Works



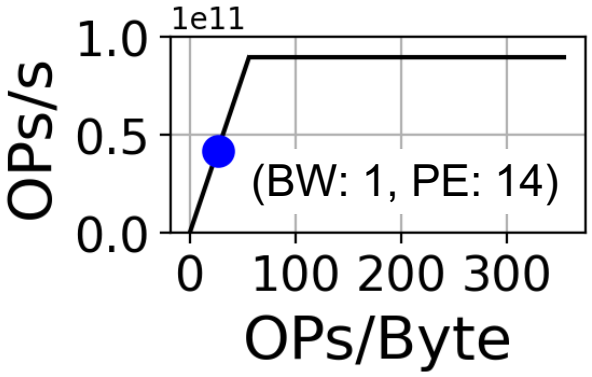
- CTA – Token Pruning and FlightLLM – Sparse Multiplication
  - GEMM dataflow and No Weight Packing
- MEADOW – TPHS + Weight Packing
- MEADOW wins by minimizing memory fetch and storage latency
  - Prefill → 2x lower Latency than Baseline
  - Decode → 1.6x lower latency than baseline

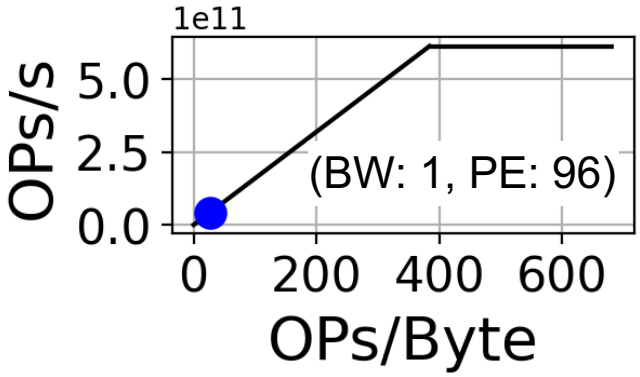
Yale University

Co-optimizing Dataflow with Memory Bandwidth & #PEs

BW (Gbps) \ # PEs	14	36	48	96
1	38.3	14.7	46.5	45.4
6	18.9	9.5	13.3	12.2
25	10.6	5.8	5.1	3.9
51	9.2	4.4	3.7	2.5

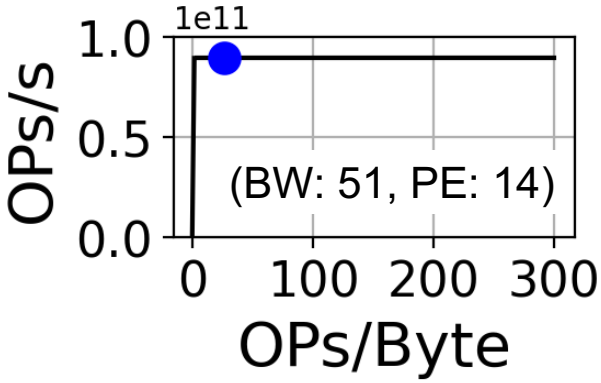
GEMM TPHS

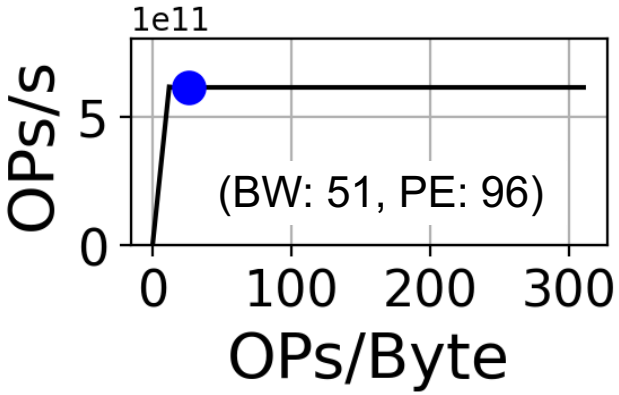




High PEs → Low BW → Memory Bottleneck

→ TPHS Preferred





Low PEs → High BW → Compute Bottleneck

→ GEMM Preferred

Yale University

Summary

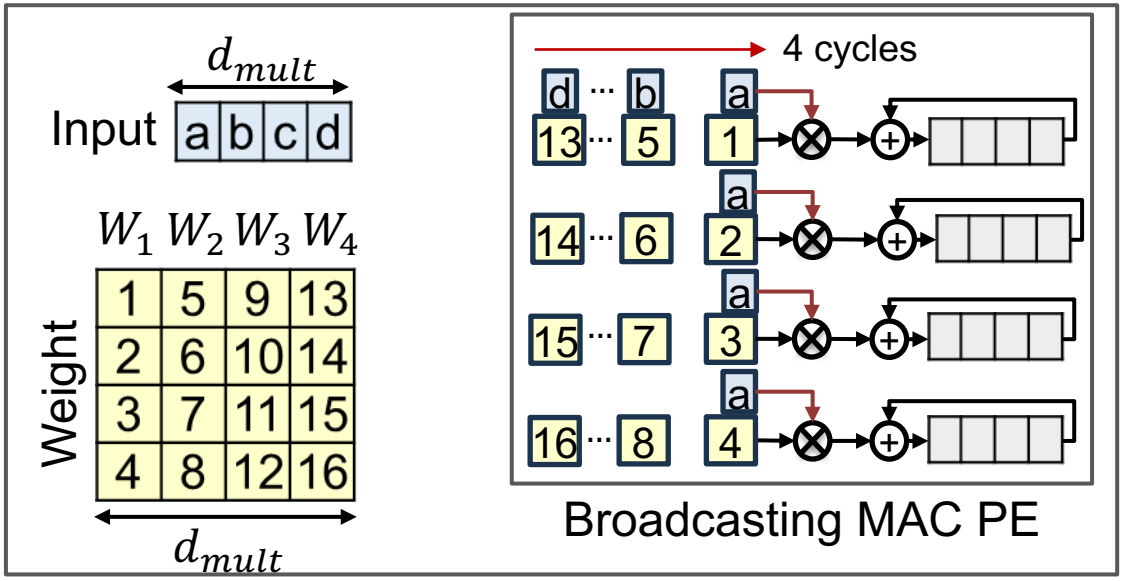
- Edge LLM Implementation is severely memory bound due to low memory bandwidth
- MEADOW Proposes Weight Packing and the Token Parallel Head Sequential Dataflow
- **Weight Packing:** Approximation-less Weight Packing Strategy → 1.8x reduction in weight fetch latency
- **TPHS Dataflow:** Prevents data movements and better pipeline efficiency → 2x lower prefill latency

Paper QR Code

Github Repository Coming Soon…

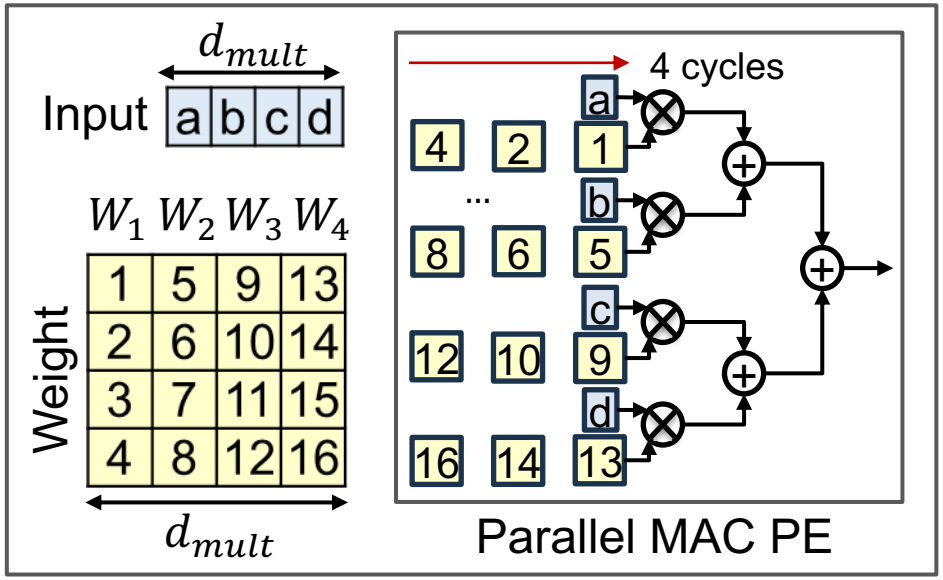
Yale University

Parallel and Broadcasting MAC Units



Broadcasting PEs

- Useful for SMxV operation
- Parallelization across output channel
- Multiply-accumulate over multiple cycles

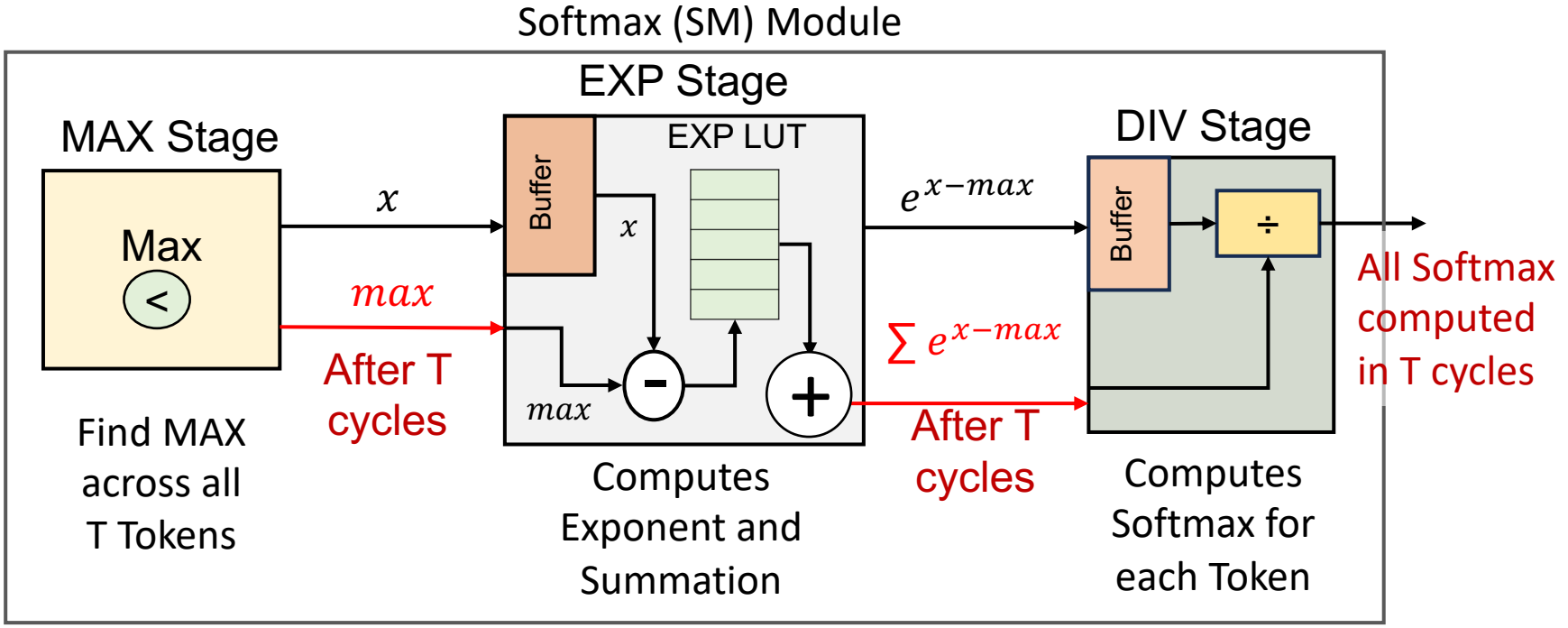


Parallel PE

- Used for all other operations
- Prallelization across input channel
- Multiplication and Accumulation in one cycle

Moitra, Abhishek, et al. "MEADOW: Memory-efficient Dataflow and Data Packing for Low Power Edge LLMs." *arXiv preprint arXiv:2503.11663* (2025).

Pipelined Softmax Unit



- Each Stage is pipelined each requiring T cycles
- Token values are compared and stored in EXP Stage Buffer
- After T cycles, max is computed
- e^{x-max} is computed and stored in DIV Stage Buffer
- $\sum e^{x-max}$ is computed in T cycles
- Softmax value is computed over T cycles

Yale University

Moitra, Abhishek, et al. "MEADOW: Memory-efficient Dataflow and Data Packing for Low Power Edge LLMs." *arXiv preprint arXiv:2503.11663* (2025).