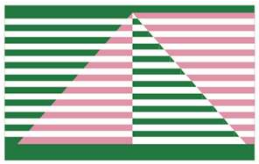


ISCA
2025

LUT Tensor Core: A Software-Hardware Co-Design for LUT-Based Low-Bit LLM Inference

Zhiwen Mo[1,2], Lei Wang[2,3], Jianyu Wei[2,4], Zhichen Zeng[2,5], Shijie Cao[2†], Lingxiao Ma[2], Naifeng Jing[6], Ting Cao[2], Jilong Xue[2], Fan Yang[2], Mao Yang[2]

1 Imperial College London

2 Microsoft Research

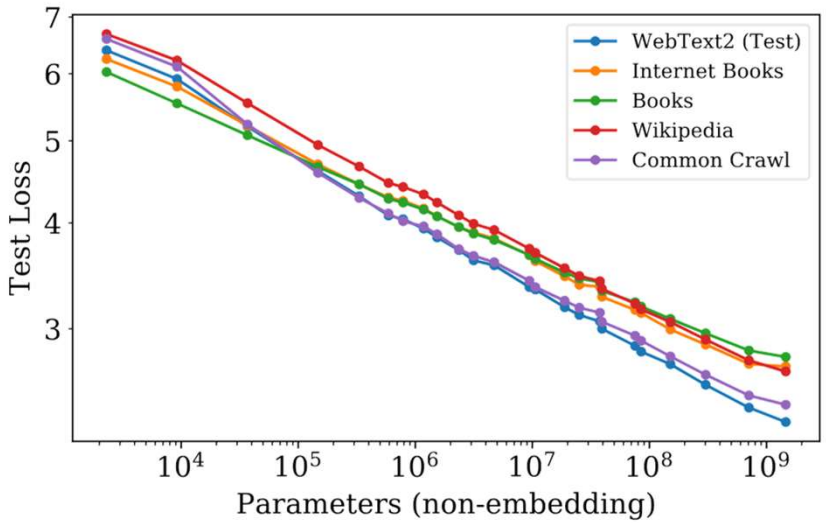
3

4

5

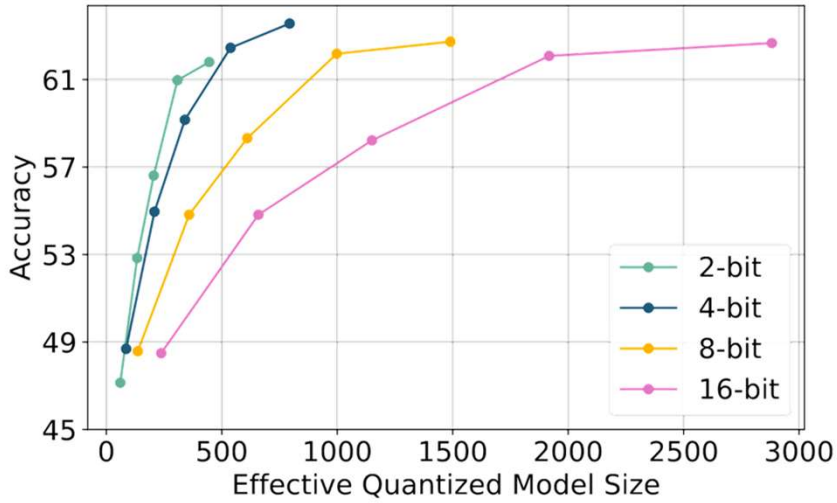
6

The New Scaling Law: Larger model, Fewer Bits!



Scaling Law[1]:

Larger model, better performance



Bit-Level Scaling Law[2]:

Larger model, fewer bits per weight

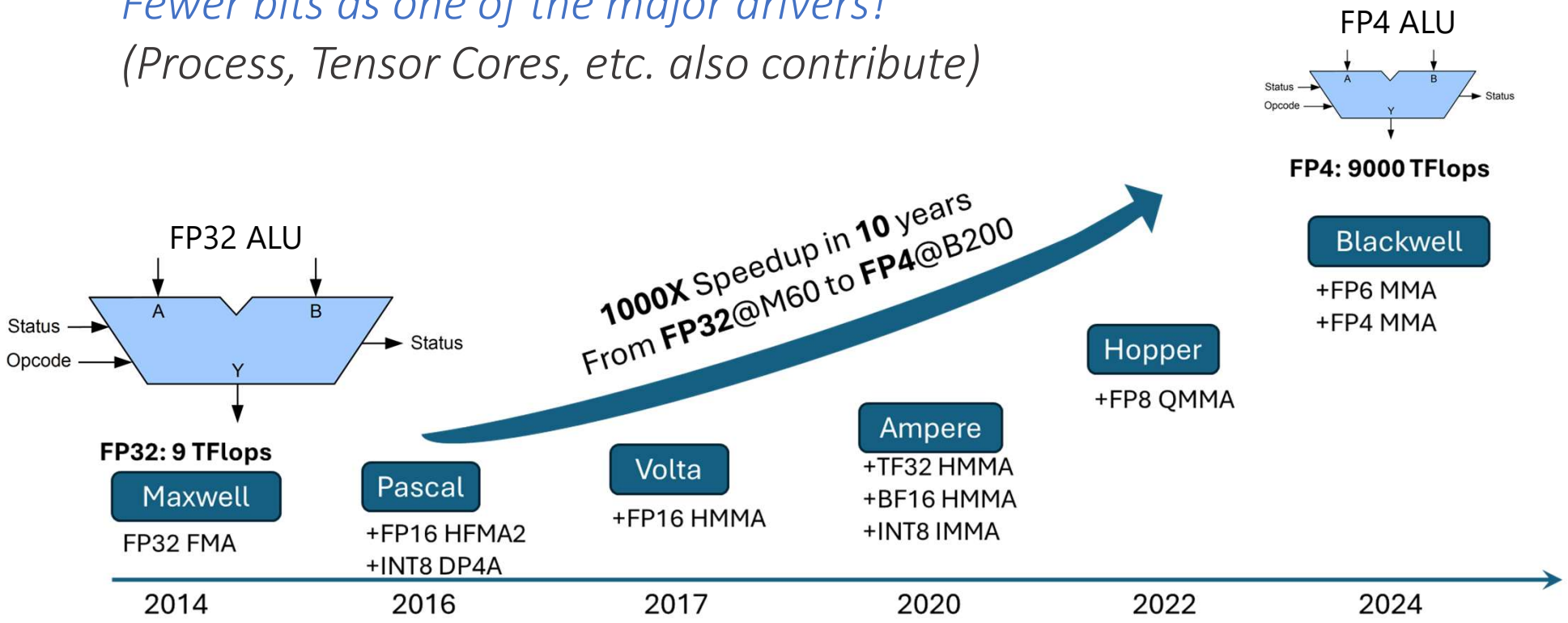
2-bit 8B is better than 8-bit 2B!

[1]:arxiv.org/pdf/2001.08361

[2]:arxiv.org/pdf/2502.02631

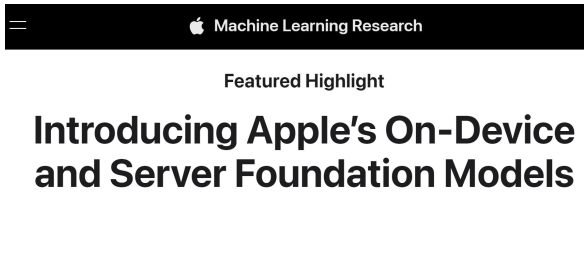
The “GPU Scaling Law”: Lower Bits, Higher TFLOPs

Fewer bits as one of the major drivers!
(Process, Tensor Cores, etc. also contribute)



Low-Bit Models: Widely Adopted in Industry & Academia

Microsoft Phi-3-mini: 4-bit

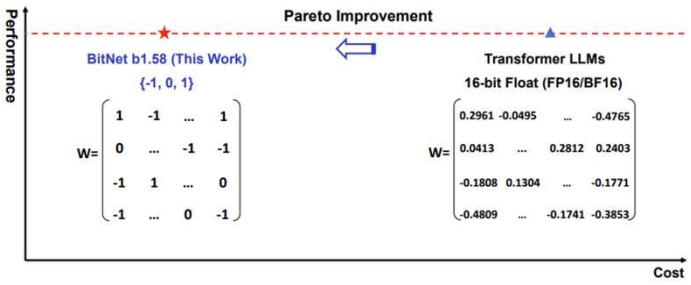


Apple Intelligence Foundation Model: 2-bit + 4-bit

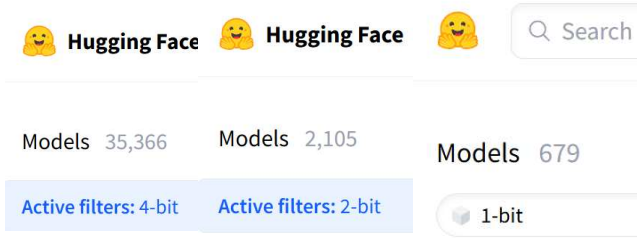
Meta

Method	Weight	Wiki2 PPL ↓	Zero-shot Accuracy ↑					
			ARC-e	ARC-c	PQ	HS	WGe	Avg.
Llama 3[18]	W_{FP16}	6.2	81.0	57.7	81.0	79.5	73.9	74.6
ParetoQ[45]	W_{INT4}	6.8	78.6	55.6	80.4	77.8	71.8	72.8
ParetoQ[45]	W_{INT2}	8.0	78.5	54.5	79.2	73.8	70.0	71.2
ParetoQ[45]	W_{INT1}	9.5	75.5	51.9	47.1	76.7	69.4	64.1

Meta ParetoQ[1]: 2-bit offers promising potential



Microsoft BitNet: 1(.58)-bit



Hugging Face: >38K 4/2/1-bit Models

Google Gemma3: Provides 4-bit model

[1]:arxiv.org/pdf/2502.02631

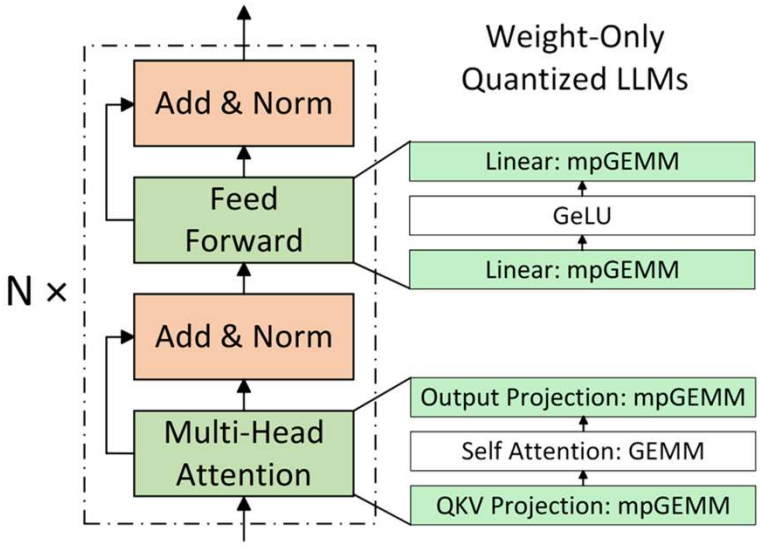
Low-bit LLMs ≈ Weight-Only Quantized LLMs
What's So Hard About Quantizing Activations?

Category	Activations	Weights
Recoverable?	Represents information of input features. Once lost, cannot be recovered.	Can be compensated by deeper networks or retraining/fine-tuning.
Outliers	More prevalent and extreme (e.g., large values, channel-wise outliers).	Fewer and more manageable.
Range	Dynamic and input-dependent. Hard to set optimal quantization range in advance.	Static and predictable; can be analyzed offline.
Bit-width	Typically 8/16 bits. Quantizing to 4 bits or lower often causes large accuracy drops.	1-bit (train from scratch), 2-bit (with distillation/mixed training), 4-bit (post-training quantization) are feasible.

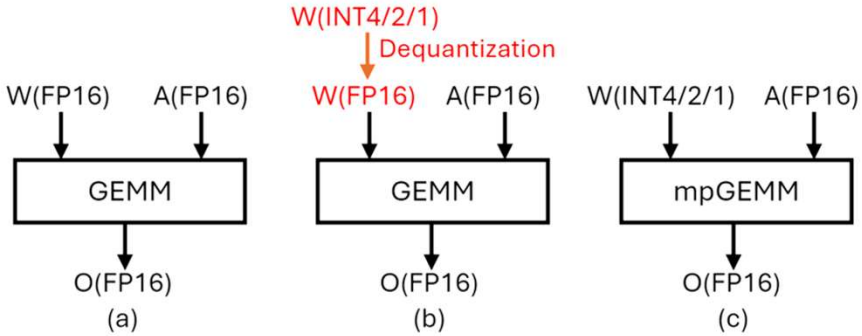
Benchmark (Metric)	Qwen2.5 1.5B-bf16	Qwen2.5 1.5B-GPTQ-int4	Qwen2.5 1.5B-AWQ-int4	BitNet b1.58 2B
Memory (Non-emb)	2.6GB	0.7GB	0.7GB	0.4GB
Activation	bf16	bf16	bf16	int8

BitNet-2B4T and Qwen2.5 weight & act. format

Shifting to New Compute Paradigm: GEMM -> mpGEMM



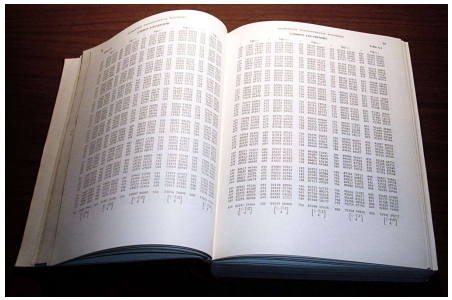
LLMs with **m**ixed-**p**recision **GEMM**
(mpgemm, E.g., $W_{INT2} \times A_{FP16}$)



(a) GEMM (b) Indirect mpGEMM
(c) Direct mpGEMM

“LUT Renaissance” For mpGEMM

Lookup Table = any *n*-input Boolean function



LUT in math handbook, Wiki

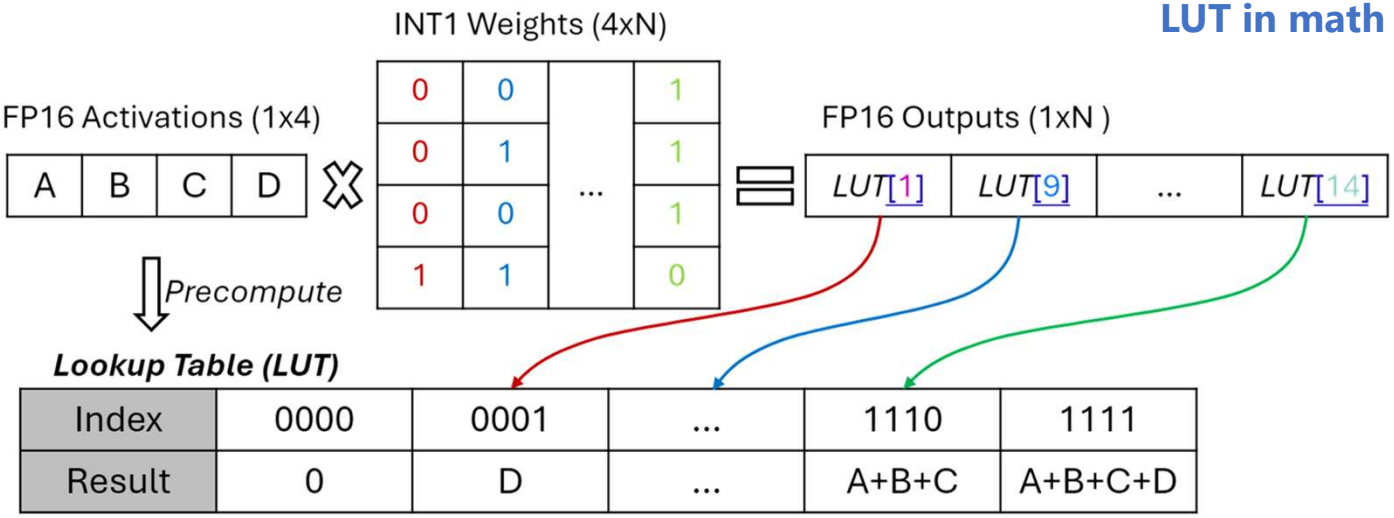
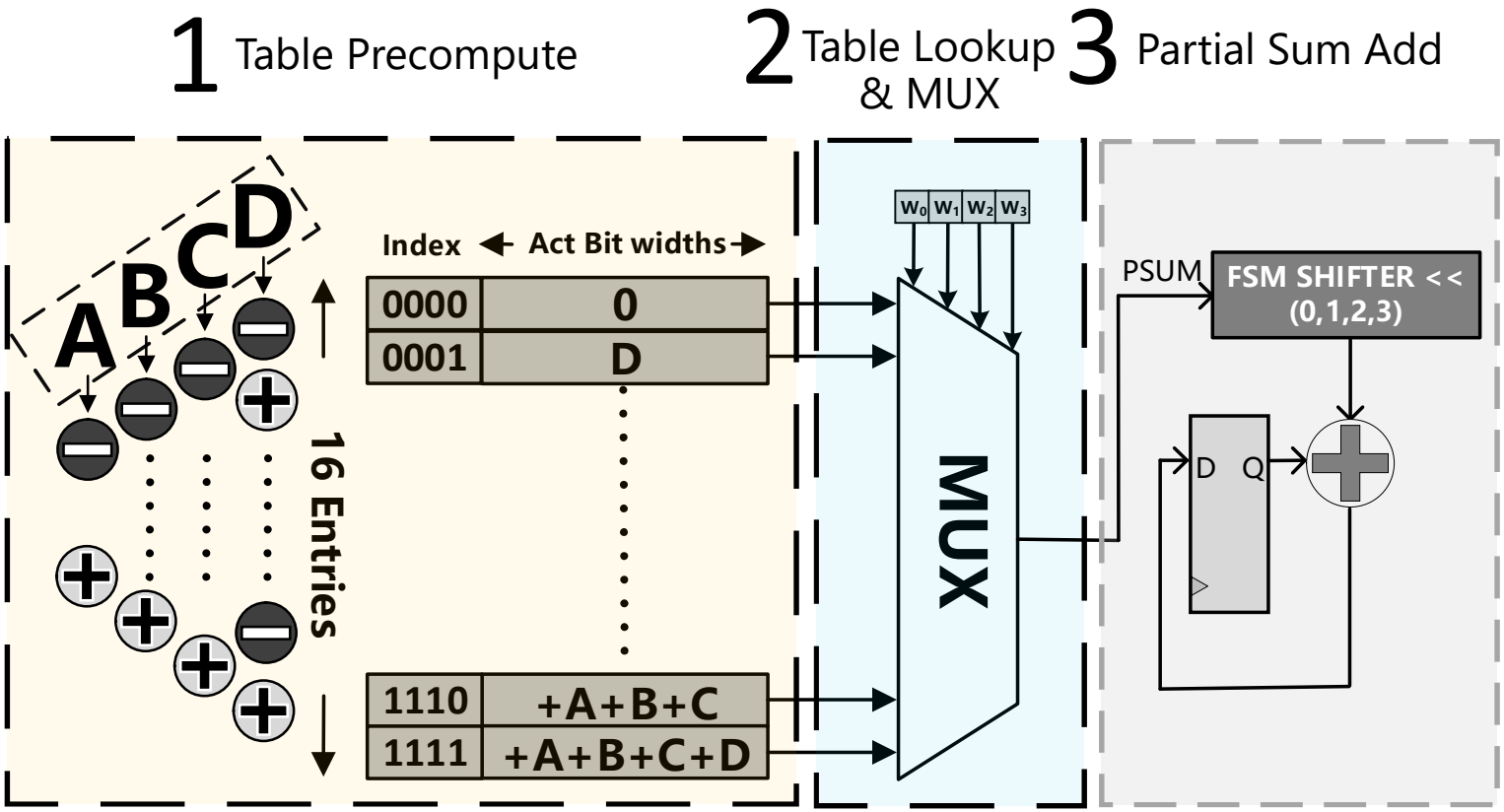


Table Lookup replaces MAC/ADD

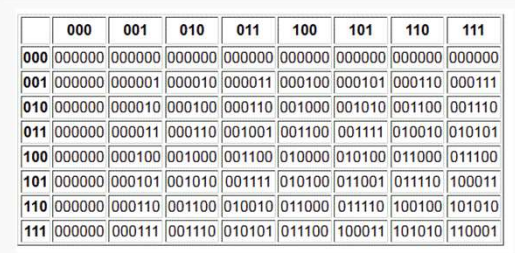
Proceeding of LUT-based dot product:



Suffer from heavy table precompute overhead!

Three Challenges in LUT-based methods:

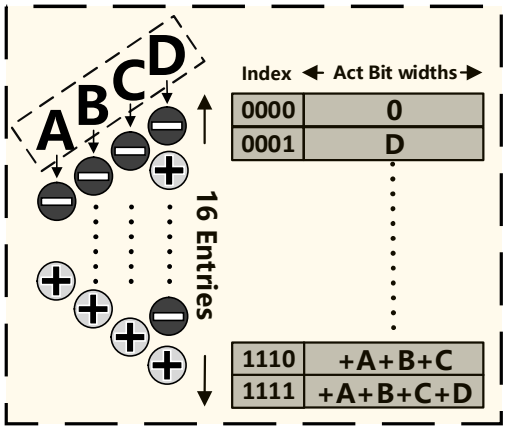
Huge Table Size



	000	001	010	011	100	101	110	111
000	000000	000000	000000	000000	000000	000000	000000	000000
001	000000	000001	000010	000011	000100	000101	000110	000111
010	000000	000010	000100	000110	001000	001010	001100	001110
011	000000	000011	000110	001001	001100	001111	010010	010101
100	000000	000100	001000	001100	010000	010100	011000	011100
101	000000	000101	001010	001111	010100	011001	011110	100011
110	000000	000110	001100	010010	011000	011110	100100	101010
111	000000	000111	001110	010101	011100	100011	101010	110001

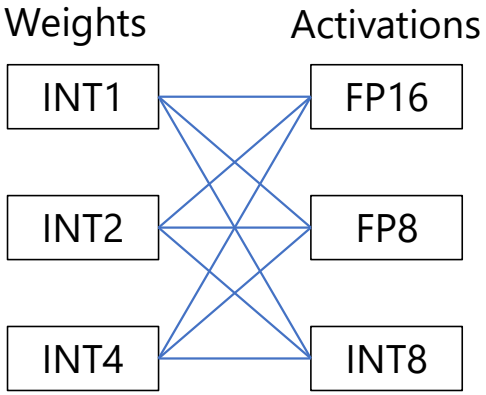
Table Runtime Generation
+ Symmetrization
+ Bind weight-bit to time

Table Precompute Overhead



DFG Transformation
+ Kernel Fusion
+ Symmetrization

Diverse Precision Combination



Bit-Serial
+ Compiler Support
+ Table Quantization

Our LUT Tensor Core

Comprehensive optimization for both software and hardware

Software's optimization for activation:

- DFG transformation to avoid duplicated precompute
- Further kernel fusion to reduce I/O

Software's optimization for Weight:

- Reinterpretation to enable symmetrization
- One time transform to simplify hardware

Hardware's halved LUT table

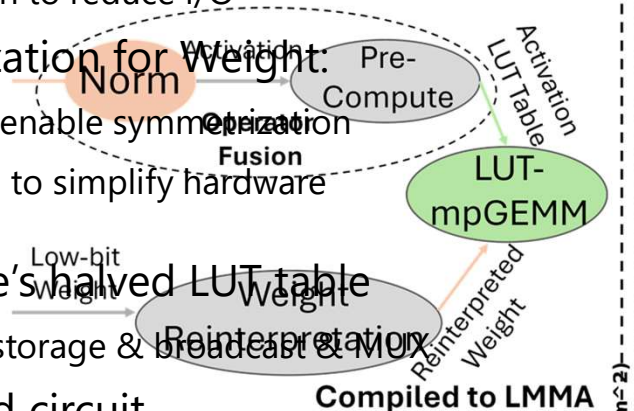
- Halved storage & broadcast & MUX

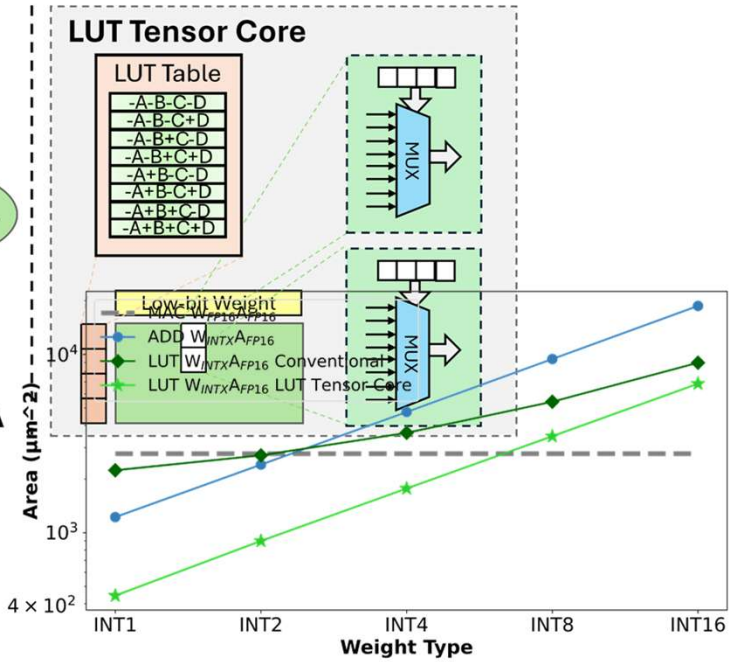
Simplified circuit

- Eliminated negation logic during compilation

Support for weight bit-width variance

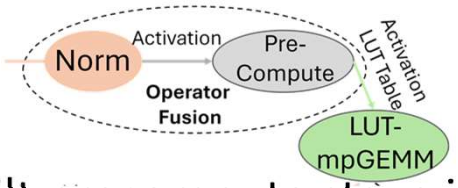
- Bit-serial like design for flexibility





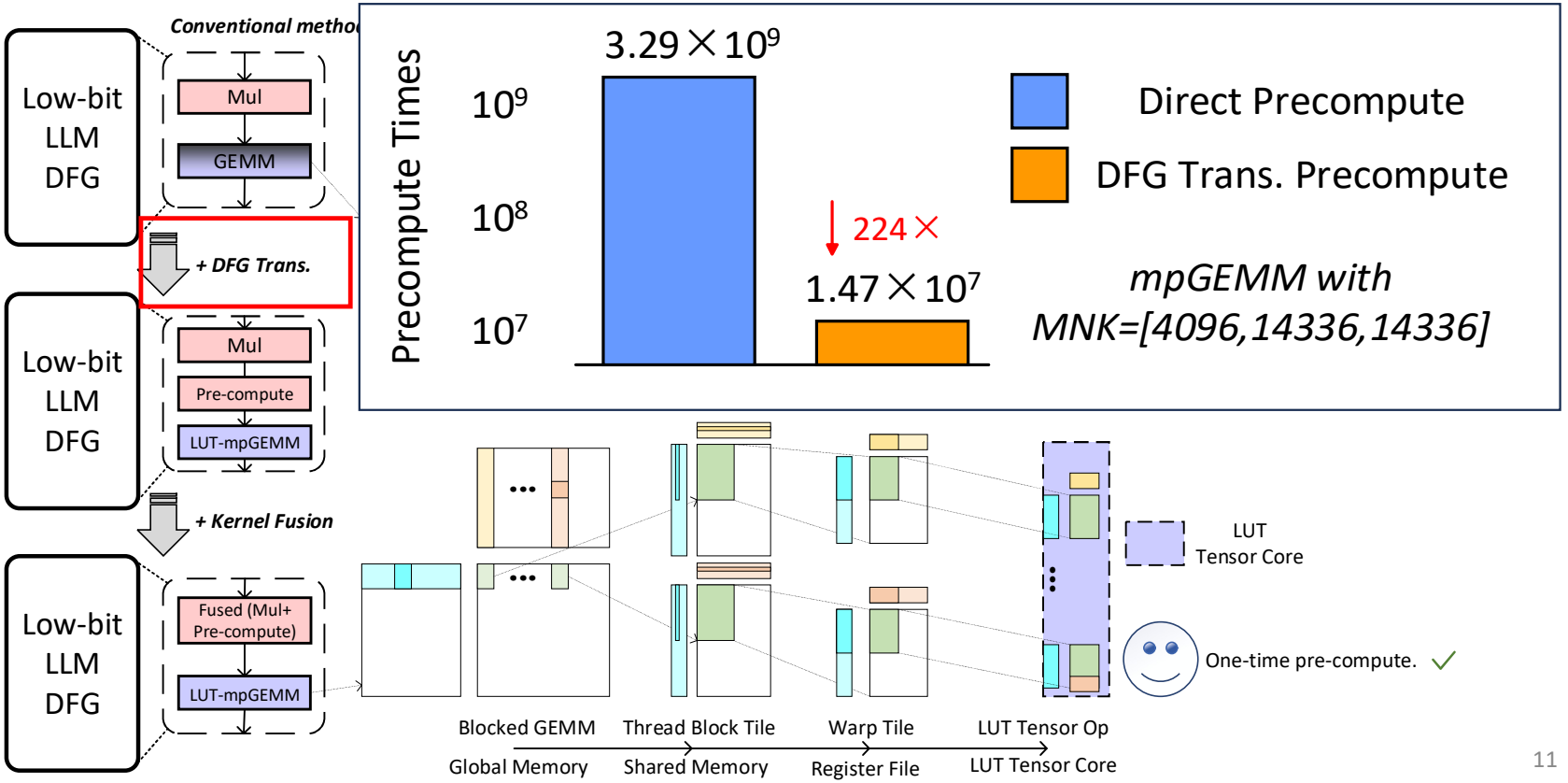
LUT Tensor Core Significantly outperforms conventional LUT

Activation

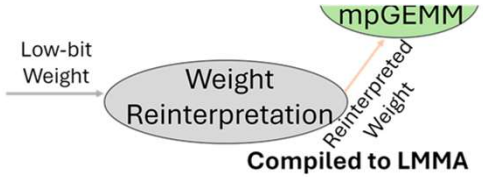


DFG Transform: Spilt precompute stage into an individual operator

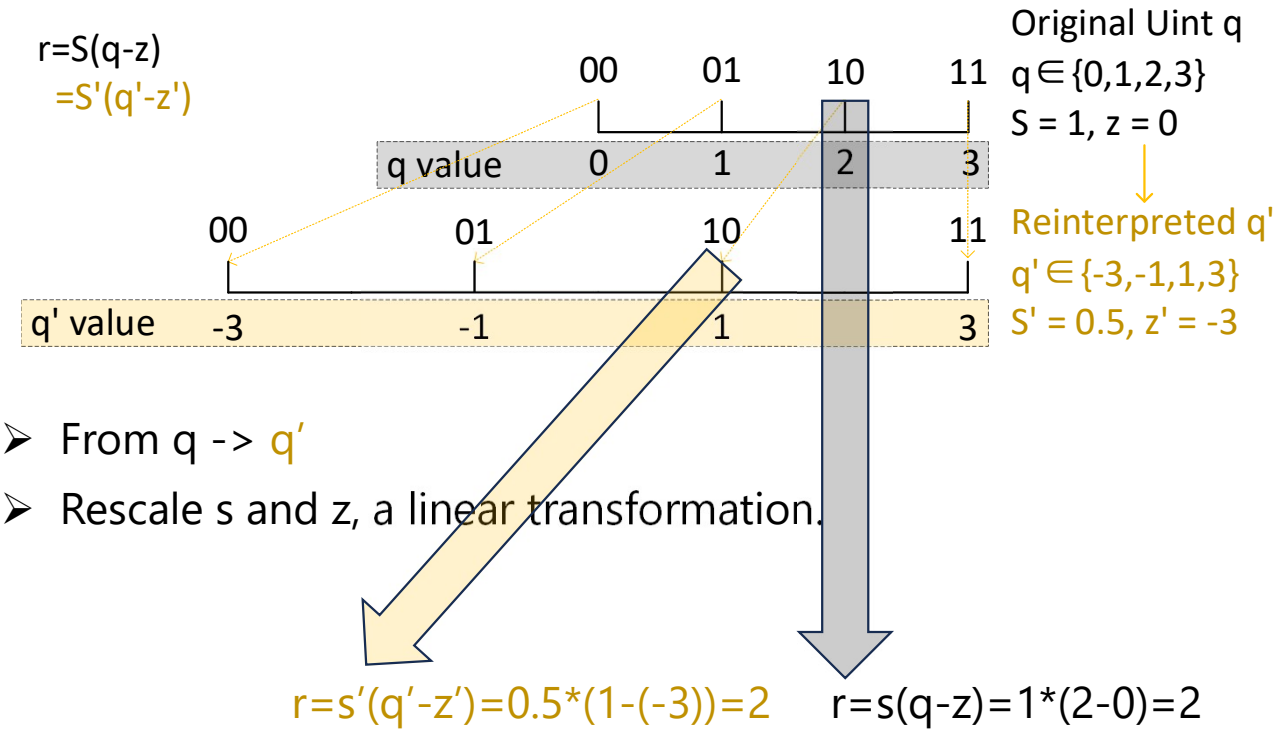
Operator Fusion: Reduce I/O



Weight



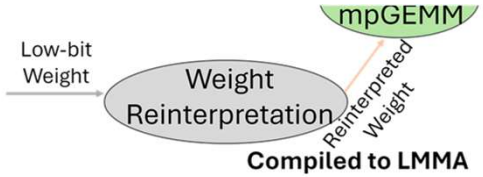
Weight Reinterpretation: enables symmetrization



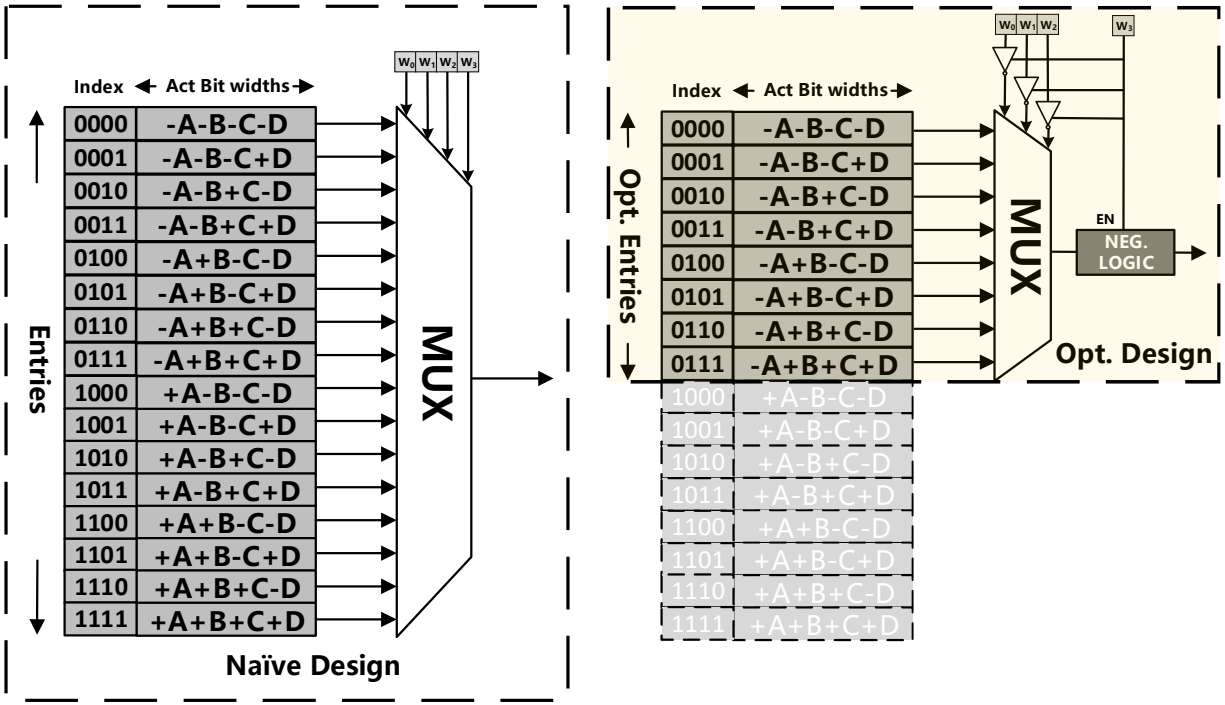
- From q -> q'
- Rescale s and z, a linear transformation.

Abbr.	Meaning
r	Real value
s	Scale factor
q	Quantized value
z	bias

Weight

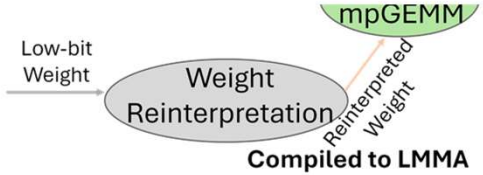


Symmetrization enables halved table item



$$\mathrm{LUT}[W_3W_2W_1W_0] = -\mathrm{LUT}[\sim(W_3W_2W_1W_0)]$$

Weight

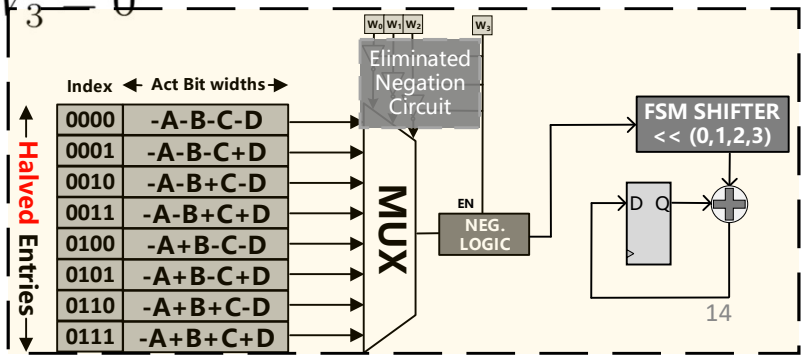


Negation logic in circuit can be further eliminated

$$\mathrm{LUT}[W_3W_2W_1W_0] = \begin{cases} -\mathrm{LUT}[\sim(W_2W_1W_0)], & \text{if } W_3 = 1 \\ \mathrm{LUT}[W_2W_1W_0], & \text{if } W_3 = 0 \end{cases}$$

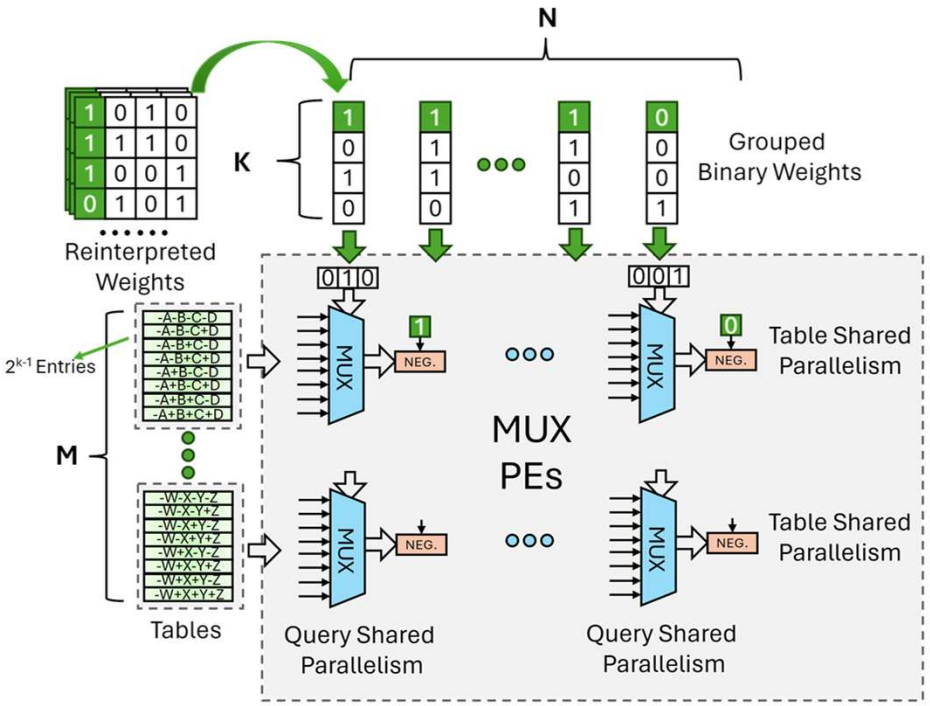
$$W'_3 = W_3 \quad W'_2 = \begin{cases} \sim W_2, & if\ W_3 = 1 \\ W_2, & if\ W_3 = 0 \end{cases} \quad W'_1 = \begin{cases} \sim W_1, & if\ W_3 = 1 \\ W_1, & if\ W_3 = 0 \end{cases} \quad W'_0 = \begin{cases} \sim W_0, & if\ W_3 = 1 \\ W_0, & if\ W_3 = 0 \end{cases}$$

$$\mathrm{LUT}[W'_3W'_2W'_1W'_0] = \begin{cases} -\mathrm{LUT}[W'_2W'_1W'_0], & \text{if } W'_3 = 1 \\ \mathrm{LUT}[W'_2W'_1W'_0], & \text{if } W'_3 = 0 \end{cases}$$



LUT Tensor Core Microarchitecture & Workflow

MNK parallelism of LUT Tensor Core



Compilation Support for LUT Tensor Core:

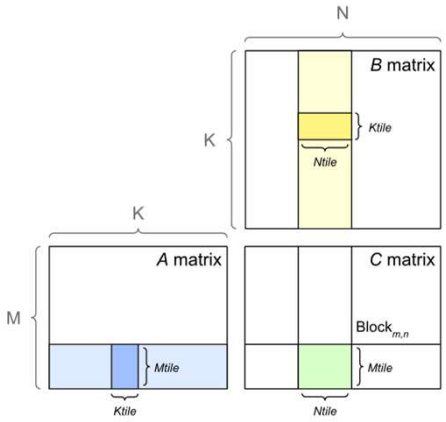
LUT-based Matrix Multiply-Accumulate (LMMA)

$$\text{lmma.}\{\text{M}\}\{\text{N}\}\{\text{K}\}.\{A_{dtype}\}\{W_{dtype}\}\{Accum_{dtype}\}\{O_{dtype}\}$$

Compilation Flow

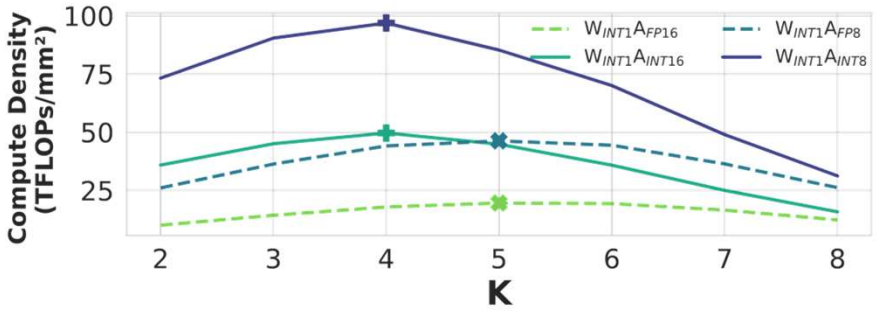
- DFG Transformation.
- Operator Fusion.
- Weight Reinterpretation.
- LUT-based mpGEMM Scheduling.
  - Elongated tiling strategy by *Roller* & *Ladder*.
- Code Generation.
  - LMMA inst. registered as intrinsics in *TVM*.

Hardware Evaluation



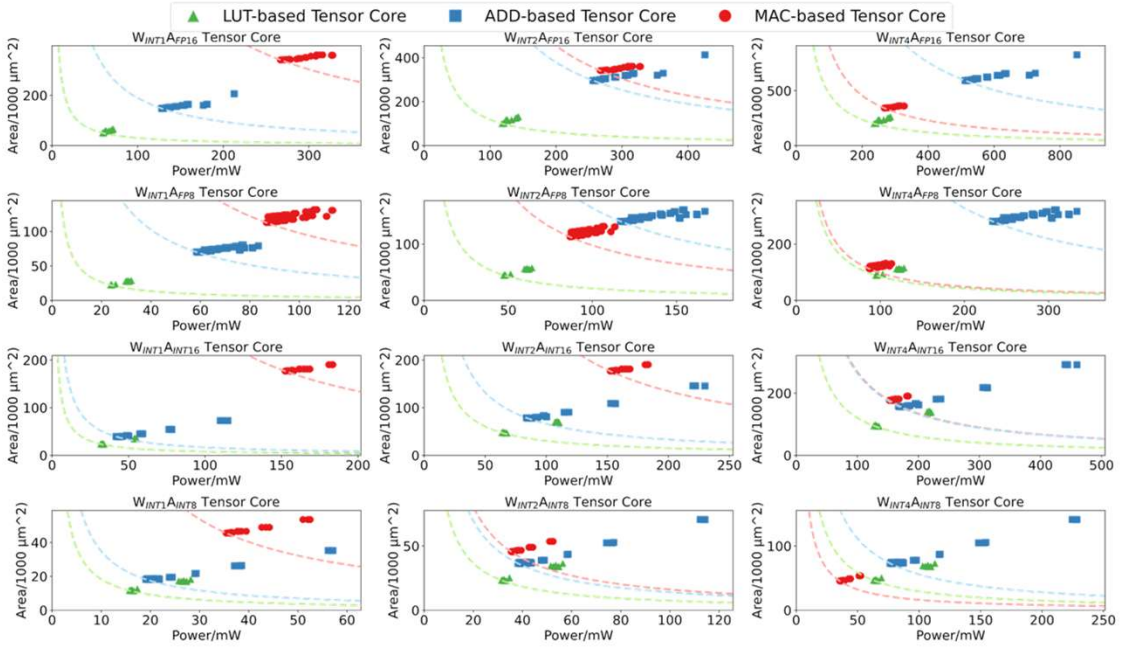
Meaning of M, N, and K in GEMM (from NVIDIA Docs)

Dot Product Level K DSE



M=N=1

Tensor Core Level MNK DSE

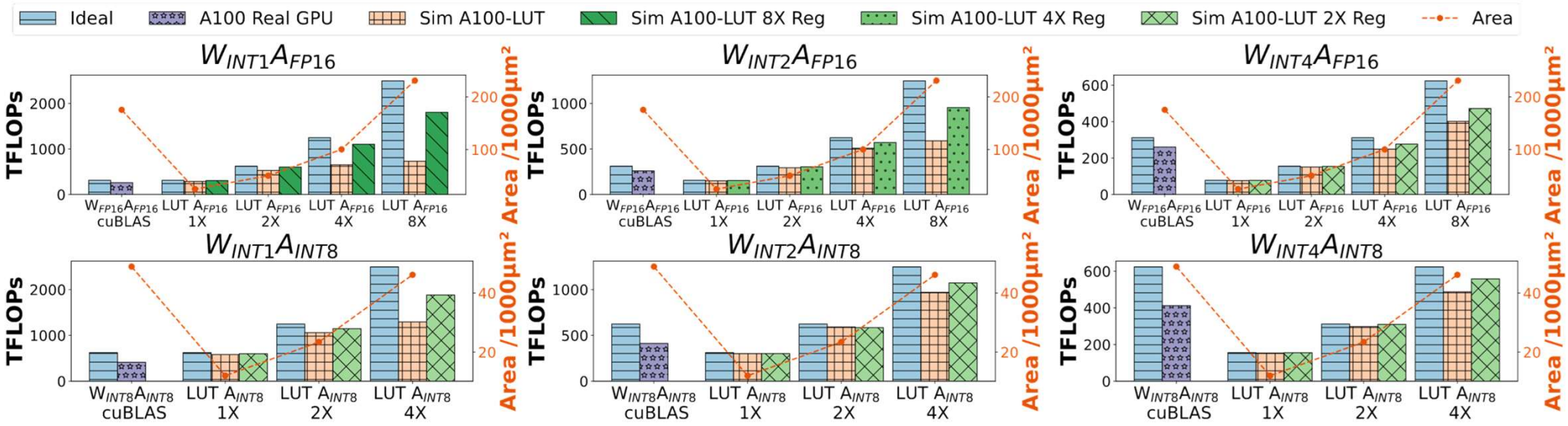


@TSMC 28nm hpc, DC 2018, medium effort, 1GHz

Dotted line means minimum product of power*area

Operator Level Evaluation

mpGEMM kernel on Accel-Sim across $W_{IN1\sim INT4}\ A_{INT8,FP16}$



Up to 6.9x acceleration under same area

Overall Comparison

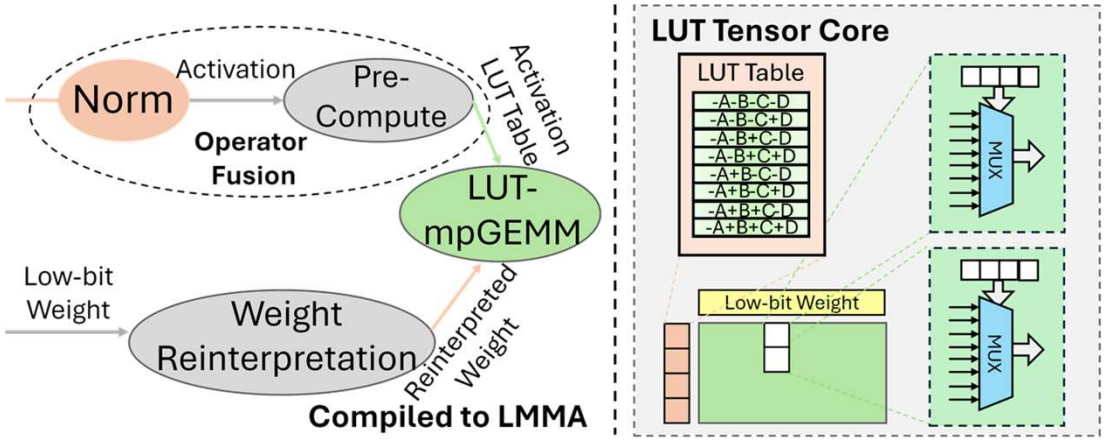
Table 1: Overall comparison.

HW. Config.	Model	Model Avg. Acc.	BS1 SEQ2048 Latency	BS1024 SEQ1 Latency	Peak Perf.	TC. Area Per SM	TC. Compute Density	TC. Energy Efficiency
A100† FP16 TC.	LLAMA 3B ($W_{FP16}A_{FP16}$)	49.7%	106.71ms	41.15ms	312 TFLOPs	0.975mm^2	2.96 TFLOPs/mm^2	2.98 TFLOPs/W
A100† INT8 TC	BitNet b1.58 3B ($W_{INT2}A_{INT8}$)	49.4%	67.06ms	21.70ms	624 TOPs	0.312mm^2	17.73 TOPs/mm^2	19.94 TOPs/W
A100†-LUT-4X*	BitNet b1.58 3B ($W_{INT2}A_{INT8}$)	49.4%	42.49ms	11.41ms	1248 TOPs	0.187mm^2	61.84 TOPs/mm^2	33.32 TOPs/W
A100†-LUT-8X*	BitNet b1.58 3B ($W_{INT2}A_{INT8}$)	49.4%	38.02ms	7.47ms	2496 TOPs	0.373mm^2	61.95 TOPs/mm^2	33.65 TOPs/W
H100† FP8 TC	BitNet b1.58 3B ($W_{FP8}A_{FP8}$)	-	38.20ms	12.30ms	1525 TFLOPs	0.918mm^2	12.59TFLOPs/mm^2	12.24TFLOPs/W
H100†-LUT-4X*	BitNet b1.58 3B ($W_{INT2}A_{FP8}$)	-	28.70ms	9.90ms	1525 TFLOPs	0.488mm^2	23.69TFLOPs/mm^2	16.35TFLOPs/W
H100†-LUT-8X*	BitNet b1.58 3B ($W_{INT2}A_{FP8}$)	-	23.48ms	5.97ms	3049 TFLOPs	0.909mm^2	25.40TFLOPs/mm^2	17.32TFLOPs/W

Up to 6.93x acceleration with 38.3% of conventional FP16*Fp16 Tensor Core's area

Accuracy reported from BitNet https://arxiv.org/abs/2402.17764

Ablation Study with previous SOTA



Configuration	Area (mm^2)	Normalized Compute Intensity	Power (mW)	Normalized Power Efficiency
UNPU (DSE Enabled)	17,271.71	1×	23.39	1×
+ Weight Reinterpretation	13,116.60	1.317×	17.98	1.301×
+ Negation Circuit Elimination	12,780.05	1.351×	17.37	1.347×
+ DFG Trans. + Kernel Fusion				
=LUT Tensor Core (Proposed)	11,991.29	1.440×	16.22	1.442×

1.44 × Efficiency than UNPU[JSSC'19], previous LUT-based SOTA accelerator work

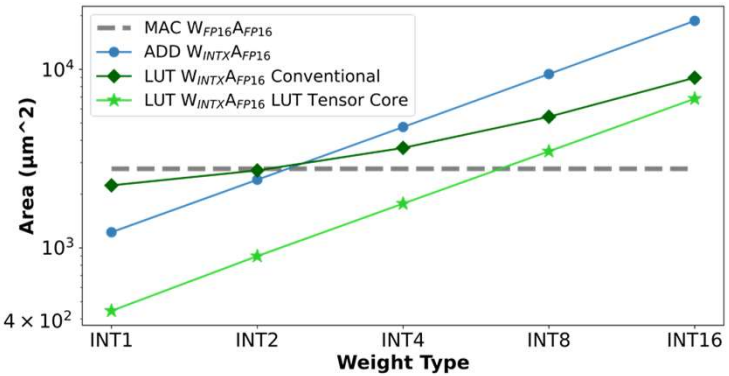
Summary

Low-bit LLMs provide new opportunities for scaling law.

- LUT Tensor Core: LUT-based direct support for mpGEMM
- A native support for mpGEMM, compatible to GPU ecosystems
- Software-hardware co-design solving three challenges in LUT-based methods:
  - Table storage, Table precompute overhead, Diverse Precision Combination

Limitations & Future Directions:

- Inference only.
  - mpGEMM is not adopted in training.
- Self-attention is still in high bits
  - Currently offloaded to cuda cores.
- mpGEMM $W_{INTX} \times A_{FP16}$, turning point is W_{INT6}.
- Direct non-integer weights supports (1.58 bits)



1 Imperial College London

2 Microsoft Research

3 

4 

5 

6 

LUT Tensor Core: A Software-Hardware Co-Design for LUT-Based Low-Bit LLM Inference

Zhiwen Mo[1,2], Lei Wang[2,3], Jianyu Wei[2,4], Zhichen Zeng[2,5], Shijie Cao[2†], Lingxiao Ma[2], Naifeng Jing[6], Ting Cao[2], Jilong Xue[2], Fan Yang[2], Mao Yang[2]

Thank you!

zhiwen.mo25@ic.ac.uk
Corresponding† : shijiecao@microsoft.com

GitHub