ARITH 2026

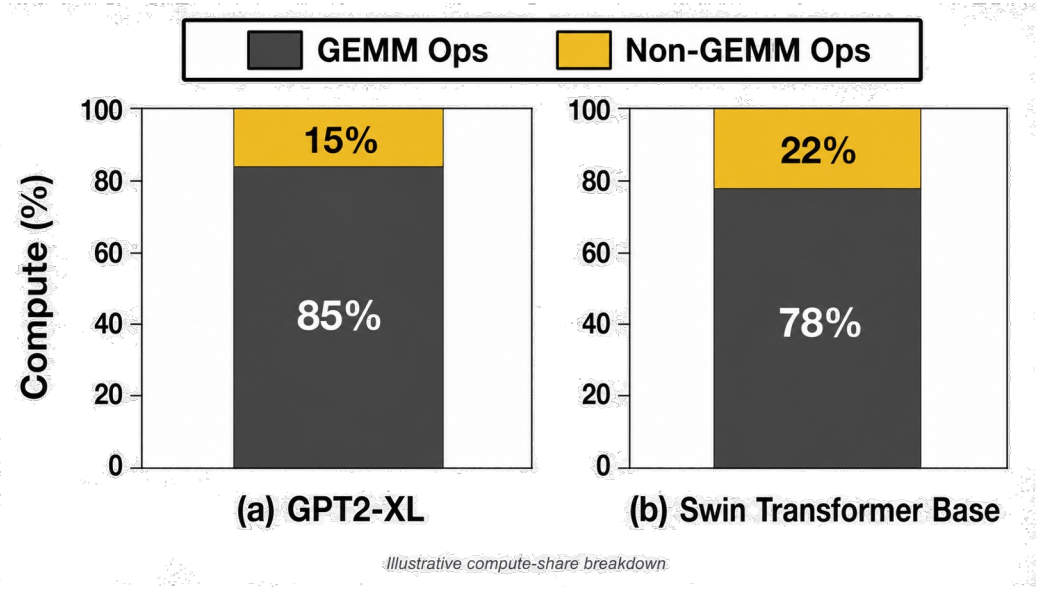
# TensorGauge:
# A Pre-silicon End-to-end Framework for Quantifying Numerical Effects of Tensor Core Microarchitecture in GEMM


*Wei Liu[1], Yi Yu[2], Qianliang Liu[1], Xiaoyou Song[1], Mingyuan Ma[1], Yuhan Wang[1], Haonan Sun[1], Yumin Hou[2], Hu He[1,*]*

1. *School of Integrated Circuits, Tsinghua University, Beijing 100084, China.*
2. *Beijing University of Technology, Beijing 100124, China.*

# Motivation: Why does this problem matter?

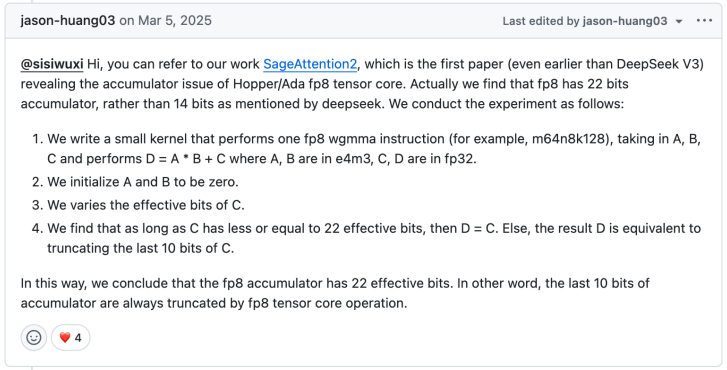
- GEMM numerical behavior depends on implementation:
  - GEMM impacts deep learning workloads. AI accelerators(e.g. GPGPU) therefore rely on low-precision tensor cores or other matrix engine.
  - Floating-point arithmetic is non-associative.
  - IEEE 754 defines formats and rounding mode, but not parallel **reduction** structure.



# Motivation: Why does this problem matter?

- Different tensor-core organizations across platforms and generations can therefore produce different results.
  - Discussions surrounding DeepSeekV3
  - SageAttention2 reports indicate that the accumulation precision in FP8 tensor core paths of NVIDIA's Hopper GPU may be insufficient, potentially preventing stable convergence for specific training configurations.

**Takeaway:**
Microarchitecture choices in tensor cores can propagate from numerical differences to model-level effects.



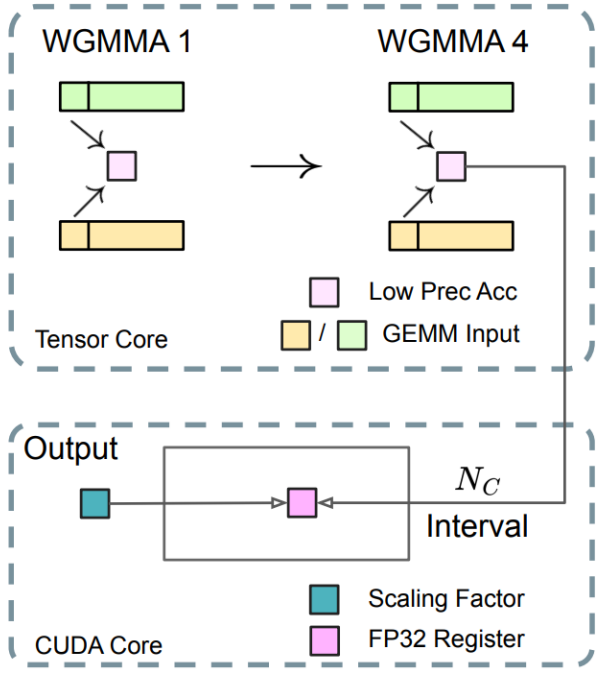
jason-huang03 on Mar 5, 2025 — Last edited by jason-huang03

@sisiwuxi Hi, you can refer to our work SageAttention2, which is the first paper (even earlier than DeepSeek V3) revealing the accumulator issue of Hopper/Ada fp8 tensor core. Actually we find that fp8 has 22 bits accumulator, rather than 14 bits as mentioned by deepseek. We conduct the experiment as follows:

1. We write a small kernel that performs one fp8 wgmma instruction (for example, m64n8k128), taking in A, B, C and performs D = A * B + C where A, B are in e4m3, C, D are in fp32.
2. We initialize A and B to be zero.
3. We varies the effective bits of C.
4. We find that as long as C has less or equal to 22 effective bits, then D = C. Else, the result D is equivalent to truncating the last 10 bits of C.

In this way, we conclude that the fp8 accumulator has 22 effective bits. In other word, the last 10 bits of accumulator are always truncated by fp8 tensor core operation.

4




(b) Increasing accumulation precision

[1] https://github.com/deepseek-ai/DeepGEMM/issues/37

# Motivation: Why does this problem matter?

- Mapping GEMM to Compute Arrays

A GEMM problem is defined as:

$$D = AB + C$$

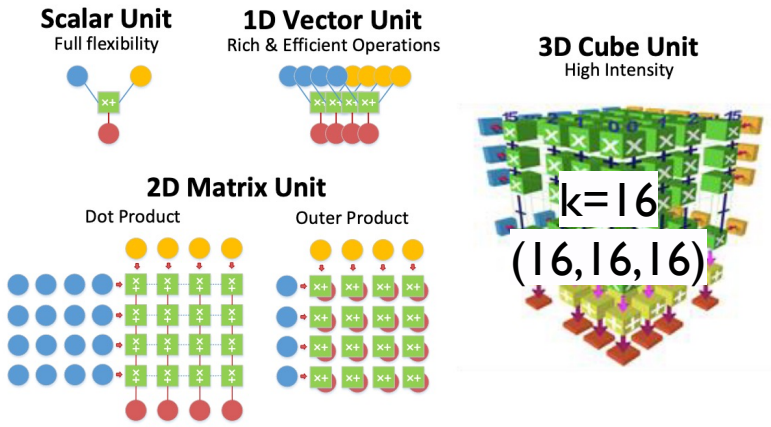
with task size: (M,N,K), where: A: $M \times K$, B: $K \times N$, C,D: $M \times N$

Hardware executes this task by mapping it onto a compute array with tile shape: (m,n,k)

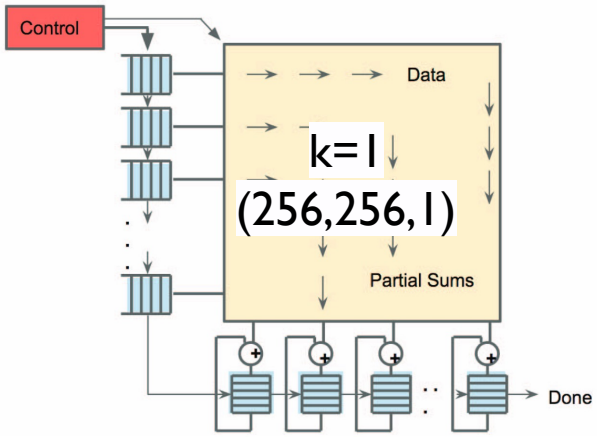
- Different tensor-core organizations choose different (m,n,k) mappings. Therefore, even for the same GEMM task (M,N,K), different platforms and generations can produce different numerical results.

# Motivation: Why does this problem matter?

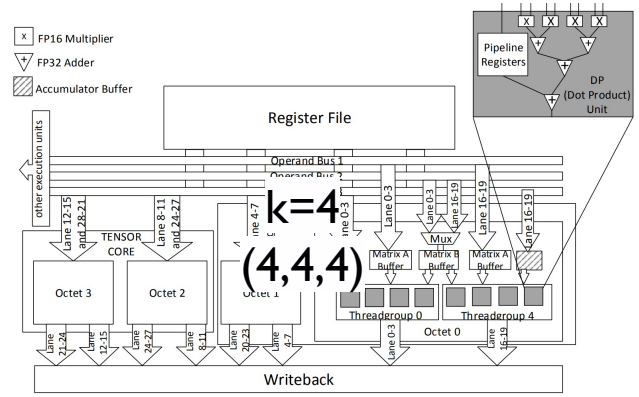
- Different tensor-core organizations across platforms and generations can therefore produce different results.
  - NPUs , TPUs, and NVIDIA GPU tensor cores use different tensor core organizations;




[1] H. Liao *et al.*, "Ascend: a Scalable and Unified Architecture for Ubiquitous Deep Neural Network Computing : Industry Track Paper," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, Feb. 2021, pp. 789–801.




[2] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, Jun. 2017, pp. 1–12.




[3] M. A. Raihan, N. Goli and T. M. Aamodt, "Modeling Deep Learning Accelerator Enabled GPUs," *2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, Madison, WI, USA, 2019, pp. 79-92, doi: 10.1109/ISPASS.2019.00016.

# Microarchitectural Design Space

- The paper evaluates several compute-unit styles:
  - **Adder/Multiplier chain**
  - **FMA** (Fused multiply-add)
  - **FDA** (Fused Dot-Add)
  - **SDA** (Separated Dot-Add)
  - **GFDA** (Grouped Fused Dot-Add)
- **Axes of variation:**
  - reduction structure along K
  - whether accumulation and add-tree are coupled
  - intermediate precision / FracBits

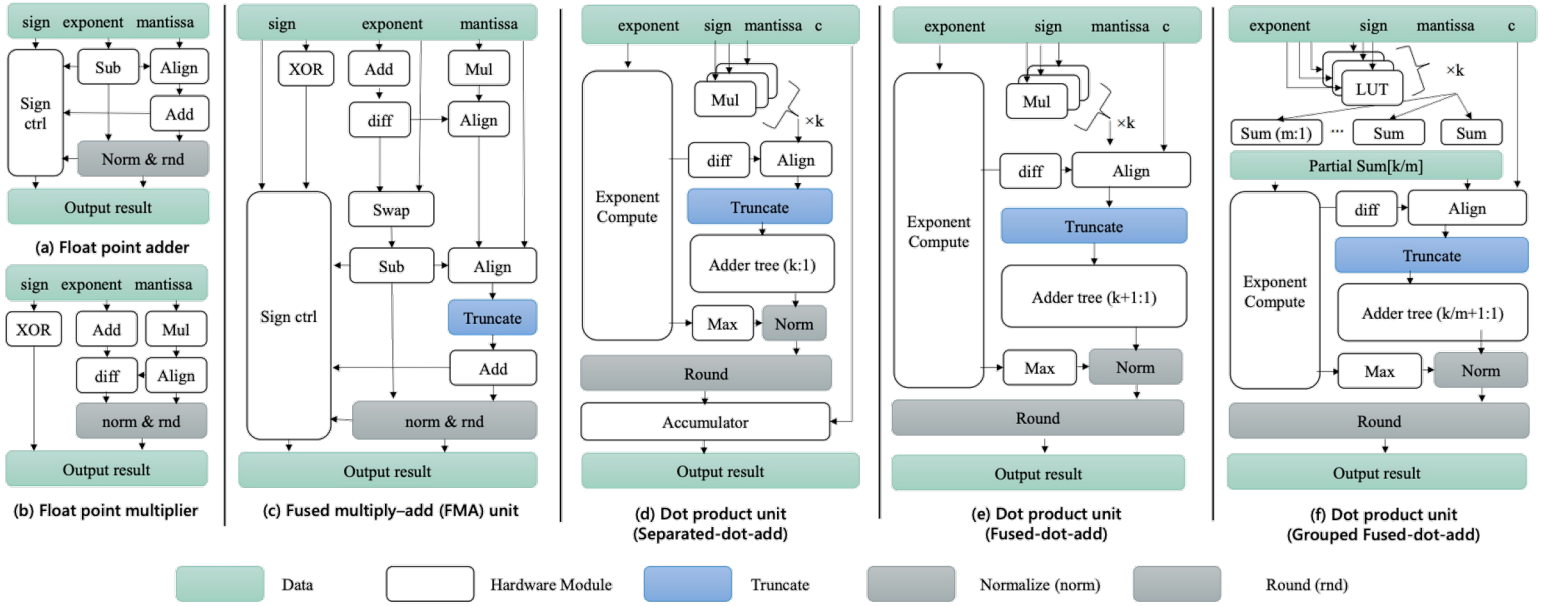



Fig. 1: Microarchitecture of floating-point operations across hardware units. (a)–(c) show scalar add, multiply, and fused multiply-add units; (d)–(f) show dot-product/add organizations that reduce four products before the optional addition of $c$.

# Background: Why Does GEMM Have Numerical Issues?

- Dot-product accumulation involves many partial products.
- Under finite precision, different reduction trees create different rounding/truncation patterns.
- Insufficient intermediate precision causes **swamping**: small values are shifted out or truncated during accumulation.
- **Implication:**
  Intermediate accumulation precision becomes a critical design parameter. The paper explains that swamping persists even with rounding and becomes more severe in dot-product units, where many products must be aligned before accumulation.

| Format | SDA | FDA | NV (Hop.) | NV (Blk.) |
|---|---|---|---|---|
| BF16 (E8M7) | 522 | 522 | 25 | 25 |
| FP16 (E5M10) | 80 | 178 | 25 | 25 |
| FP8 (E5M3) | 66 | 164 | 13 | 25 |
| FP4 (E2M1) | 6 | 132 | - | 25* |

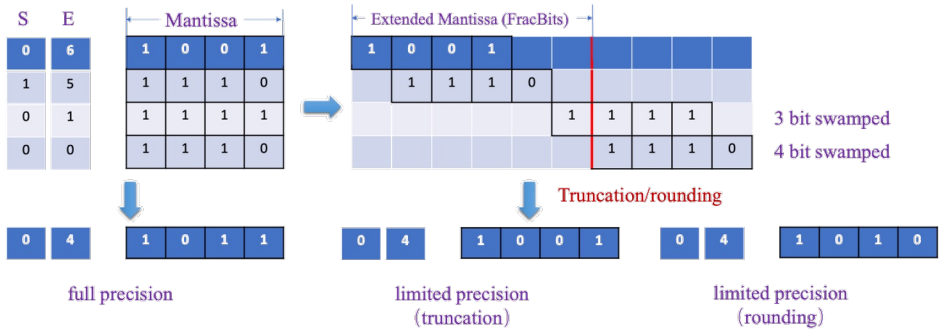



Fig. 2: Swamping in a 4-input aligned addition under limited intermediate precision. Small aligned terms can be shifted out of the retained datapath and become unrecoverable during later cancellation. The truncation and rounding examples use the same output width; rounding additionally keeps sticky information at the least significant bit (LSB), but still cannot recover values lost before accumulation.

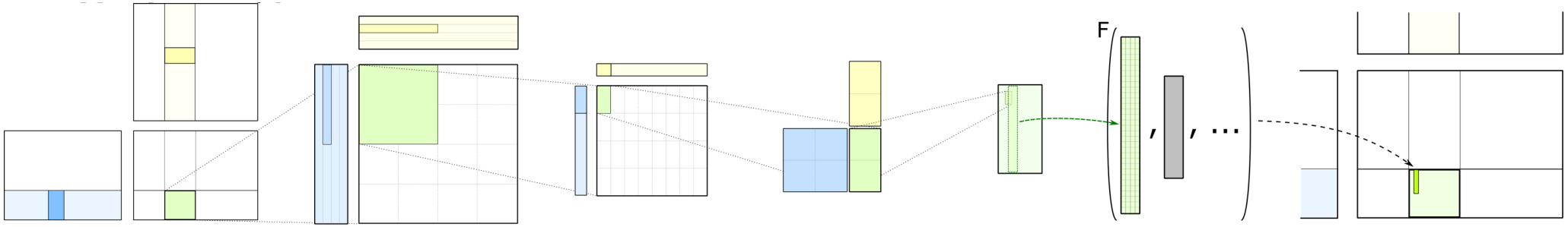
# Problem Statement

- **Central question:**
  How do GEMM numerical discrepancies introduced by tensor-core design propagate to end-to-end neural network behavior?
- **Two main sources of differences:**
  - **Software-side**
  - **Hardware-side:** tensor core organization, rounding mode, fusion strategy, fractional bits.

| Architecture | Supported instruction families | Precision / datatype | Numerical implementation |
|---|---|---|---|
| **Volta** | HMMA.884 | FP16 / FP32 accumulation variants | FDA(F=23) |
| **Turing** | HMMA.884, HMMA.1688 | FP16 / FP32 accumulation variants | FDA(F=24) |
| **Ampere** | HMMA.1688, HMMA.16816, HMMA.1684, DMMA.884, DMMA.8x8x4 | FP16, BF16, TF32, FP32 accumulation; DMMA path | HMMA: FDA(F=24) or CoFDA(F=24); DMMA: SFMA |
| **Ada Lovelace** | Ampere-style HMMA + QMMA.16832, QMMA.16816, DMMA | FP16, BF16, TF32; FP8 E4M3/E5M2; DMMA path | HMMA: FDA(F=24) / CoFDA(F=24); FP8 QMMA: FDA(F=13) / CoFDA(F=13); DMMA: SFMA |
| **Hopper** | HMMA, DMMA, HGMMA, QGMMA | FP16, BF16, TF32; FP8; DMMA path | HMMA/HGMMA: FDA(F=25); FP8 QGMMA: FDA(F=13); DMMA: SFMA |
| **Blackwell** | HMMA, UTCHMMA, UTCQMMA, UTCOMMA | FP16, BF16, TF32; UTC tensor-core paths | HMMA/UTCMMA: FDA(F=25); UTCOMMA: GDFS |
| **RTX Blackwell** | HMMA, QMMA, OMMA | FP16, BF16, TF32; FP8; FP6/FP4 formats including E2M1 | HMMA/QMMA: FDA(F=25); OMMA low-bit scaled path: GDFS |

# Problem Statement

- **Central question:**
  How do GEMM numerical differences introduced by tensor-core design propagate to end-to-end neural network behavior?
- **Two main sources of differences:**
  - **Software-side:** accumulation order, tiling, parallel reduction, merge policy
    - Sequential reduction accumulates products in a fixed order.
    - Parallel GEMM kernels may split the K dimension into multiple chunks.
    - Partial sums are then merged in an implementation-dependent order, e.g., Split-K or Stream-K.



# Key Idea

- TensorGauge is a **pre-silicon, end-to-end framework** that connects:
  - arithmetic unit semantics,
  - bit-accurate numerical modeling,
  - PyTorch-level execution,
  - full-model training and inference evaluation.
- **Goal:**
  - Provide actionable guidance for tensor-core microarchitecture design before silicon tape-out.

# TensorGauge Workflow

- **Framework overview**
  - Hardware specification / RTL implementation
  - Bit-accurate CUDA model
  - PyTorch integration via custom operators
  - Workload execution
  - Statistical analysis
  - Closed-loop improvement

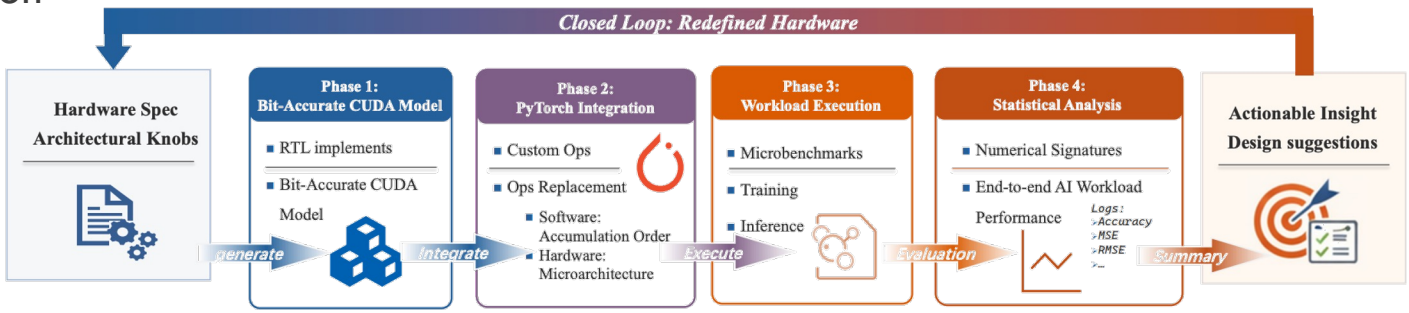



Fig. 3: TensorGauge workflow. Parameterized RTL and bit-exact CUDA models expose tensor-core semantic knobs, including reduction grouping, accumulation order, rounding, and intermediate precision; the PyTorch integration then propagates each hardware choice to unit-level metrics and end-to-end model quality.

- **Strength:**
  It avoids heavyweight full-system simulation while preserving numerical consistency for GEMM evaluation.

# Experimental Setup

- **Training:**
  - BERT, SST-2 from GLUE
  - Five random seeds per hardware setting
- **Inference:**
  - Qwen3 for WikiText-2 perplexity
- **Precisions studied:**
  FP16/FP8/FP4/MXFP4/NVFP4/BF16 inference
- **Training recipe:**
  - low precision in forward pass
  - high precision for backpropagation and gradient accumulation

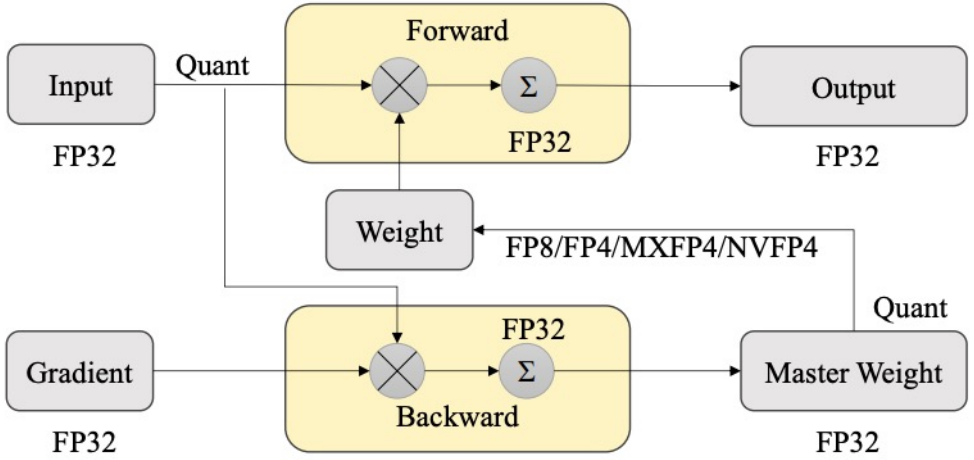
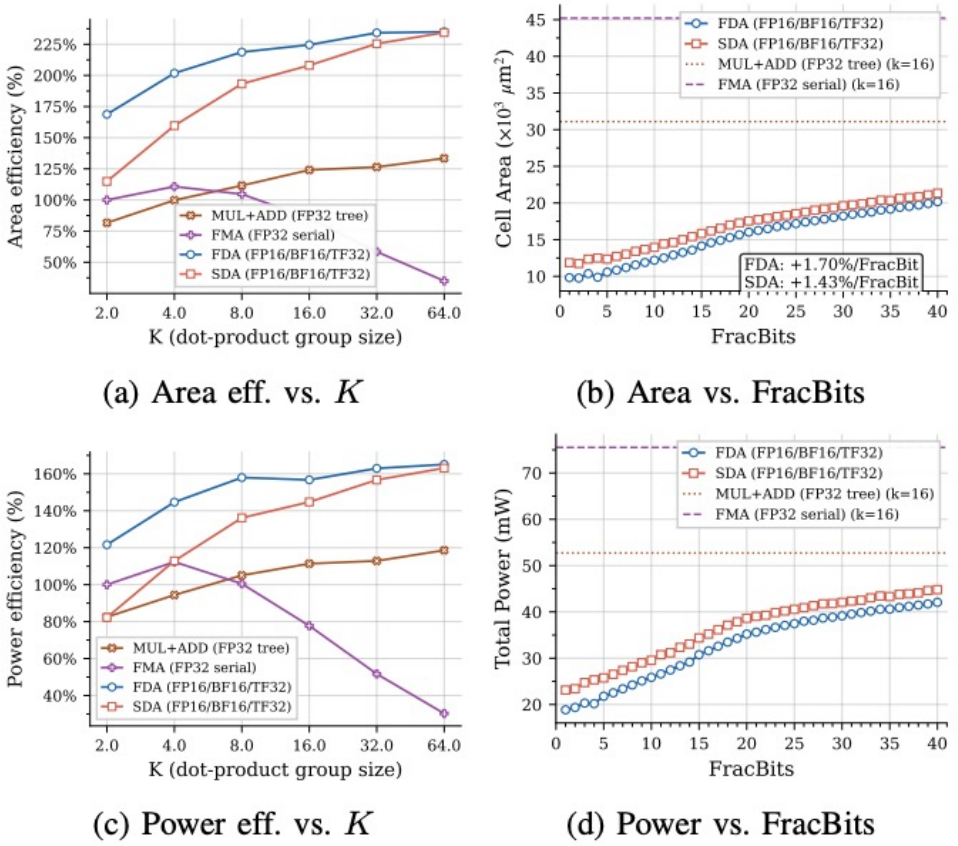



Fig. 4: Low-precision training recipe used in the end-to-end experiments. TensorGauge replaces GEMM semantics while keeping the optimizer, loss scaling, and model/training pipeline fixed, so differences can be attributed to arithmetic and reduction behavior.

# Hardware Cost Results

- **Area and power trends**
  - FDA and SDA become more area/energy efficient as accumulation depth **K** increases.
  - FMA chains scale worse because of more registers and pipeline overhead.
  - Increasing intermediate fractional precision incurs roughly linear hardware cost.
- For the BF16/FP16/TF32 shared unit, each additional FracBit adds about **1.70% area** and **1.56% power**. The paper also shows that FDA and SDA become closer in implementation cost as FracBits grows.

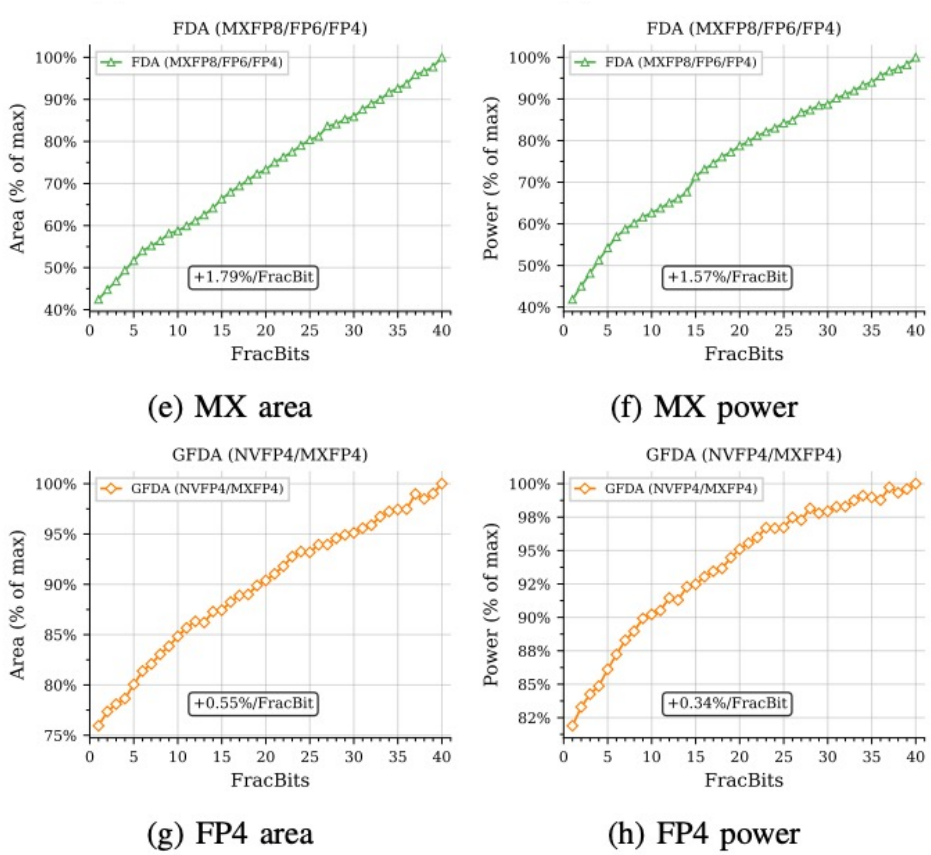



(a) Area eff. vs. $K$ (b) Area vs. FracBits

(c) Power eff. vs. $K$ (d) Power vs. FracBits

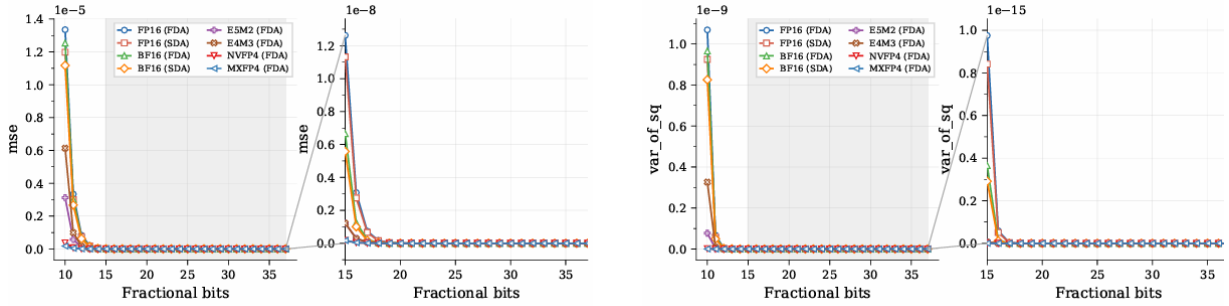
# Hardware Cost Results

- **Area and power trends**
  - FDA and SDA become more area/energy efficient as accumulation depth **K** increases.
  - FMA chains scale worse because of more registers and pipeline overhead.
  - Increasing intermediate fractional precision incurs roughly linear hardware cost.
- For the NVFP4/MXFP4 shared unit, each additional FracBit adds about **0.55% area** and **0.34% power**. The paper also shows that FDA and SDA become closer in implementation cost as FracBits grows.



# Operator-Level Numerical Results

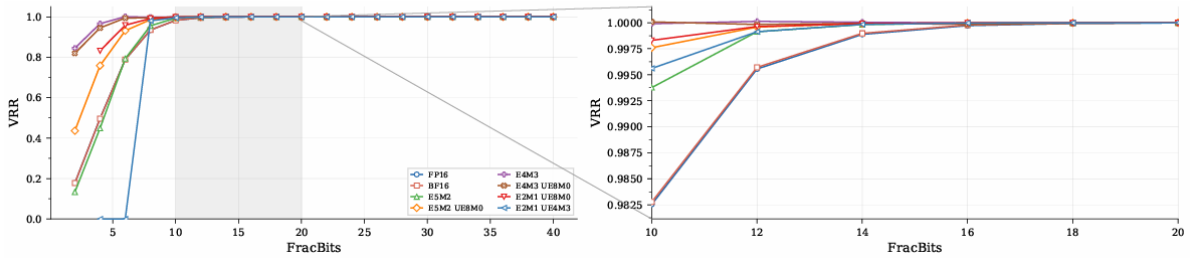
- **Metrics used:** MSE Variance of squared error VRR (variance retention ratio)
- **Observed thresholds:**
  - VRR approaches 1 at around **FracBits ≥ 16**
  - variance-related metrics stabilize around **FracBits ≈ 17–19**
- **Important insight:**
  Unit-level numerical stability appears at relatively modest precision, but this does **not** guarantee model-level stability.




(a) Mean squared error (MSE).




(b) Variance of squared error.

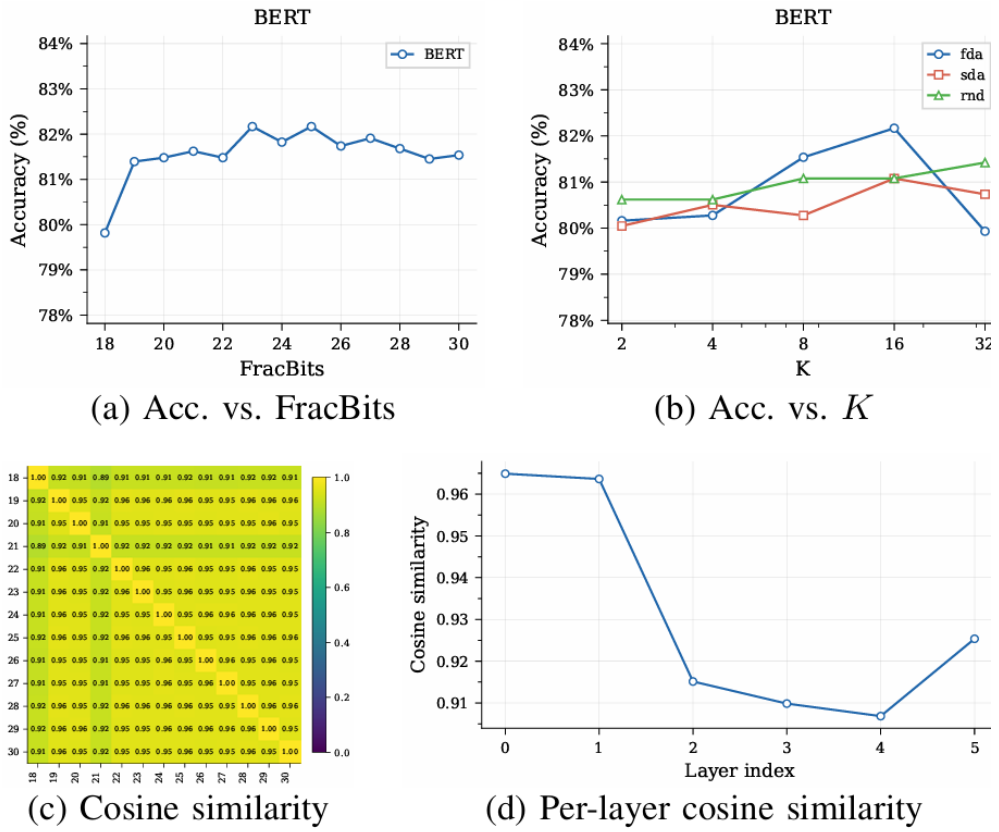



(c) variance retention ratio (VRR).

Fig. 6: Overall error statistics across configurations.

# End-to-End Training Results

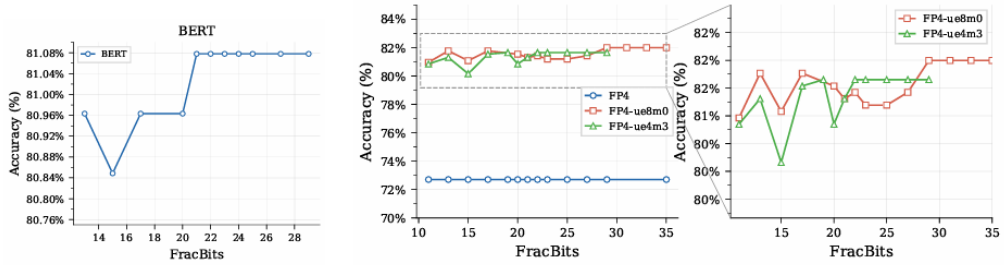
- **Model-level behavior differs from operator-level conclusions**
  - **For FP16 training:** training becomes stable when **FracBits > 23**
  - **For FP8 training:** stability appears around **FracBits = 21**
  - **For FP4 training:** stability requirements are stricter and format-dependent
    - NVFP4 stabilizes above about **FracBits > 23**
    - MXFP4 requires about **FracBits > 28**
- This gap between operator-level and end-to-end thresholds is one of the paper's main findings.




(a) Acc. vs. FracBits (b) Acc. vs. $K$

(c) Cosine similarity (d) Per-layer cosine similarity

Fig. 7: Evaluation across different hardware settings.

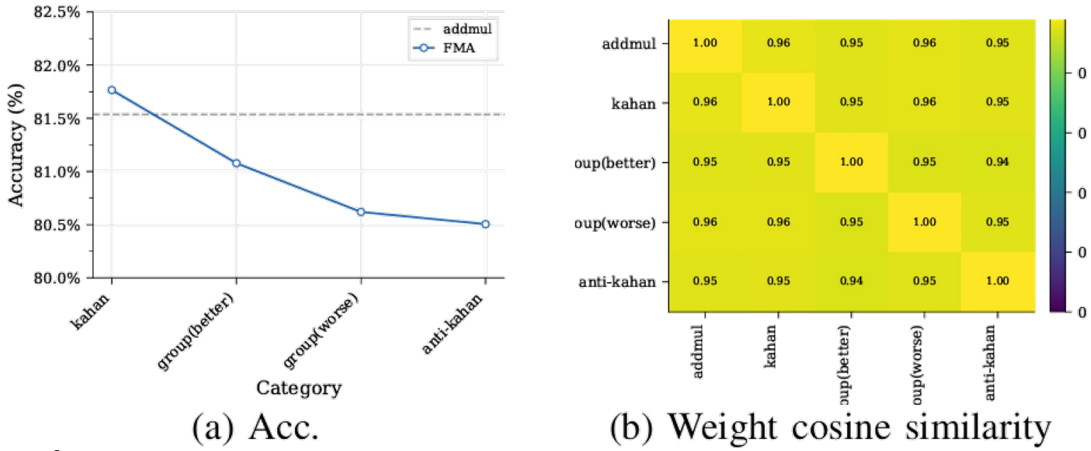



(a) FP8 training (b) FP4 training

Fig. 9: Low-bit training Acc. across FracBits.

# Effect of Accumulation Order

- The paper compares Kahan-like, grouped, random, and adversarial orderings.
- **Result:**
  - different accumulation orders lead to measurable training accuracy differences
  - best vs. worst ordering differs by about **1.26%** accuracy
  - learned weights also diverge across settings
- **Interpretation:**
  Numerical differences can steer optimization toward different local solutions.

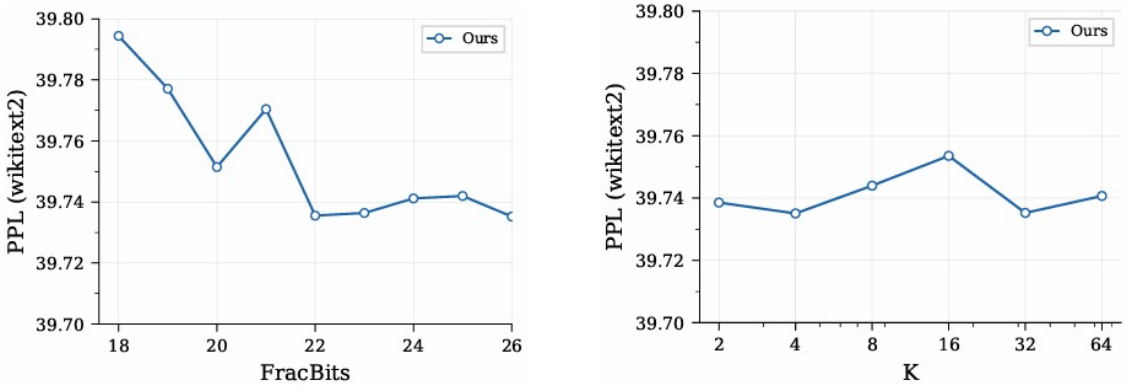



(a) Acc. (b) Weight cosine similarity

Fig. 8: Impact of accumulation order.

# Inference Results

- **BF16 LLM inference**
  - For Qwen3-0.6B on WikiText-2:
  - best perplexity occurs around **FracBits = 22**
  - perplexity differences across K are very small (below **0.02**)
- **Meaning:**
  Inference is less sensitive than training in this setup, but still shows a measurable best condition in intermediate precision.




(a) Perplexity vs. FracBits (b) Perplexity vs. $K$

Fig. 10: Qwen3 inference results across hardware design.

# Evaluation speed

- **Simulation throughput.** Table II reports the simulator throughput for GPT-2 decoding that generates 100 new tokens. Compared with a real GPU, the simulator achieves **10.74%** of native GPU speed.
- Despite this slowdown, it is 4–5 orders of magnitude faster than conventional GPU simulators and 564.02× faster than an FPGA-based alternative, which makes it practical for end-to-end model-level experiments.

TABLE II: Simulation throughput for GPT-2 decoding, reported in KIPS (kilo instructions per second).

| Simulator/Platform | Speed (KIPS) |
|---|---|
| Multi2Sim [23] | 0.8 |
| Accel-Sim (exec-driven) [23] | 6 |
| Accel-Sim (trace-driven) [23] | 12.5 |
| MGPU-Sim [23] | 28 |
| Macsim [24] | 50.5 |
| FPGA for Ventus [25] | 781.25 |
| TensorGauge (Ours) | 440640 |
| Real GPU [24] | 4103750 |

# Core Takeaways

- **Takeaway**
  - Tensor-core arithmetic semantics matter at the model level.
  - Operator-level error metrics are useful but insufficient.
  - Intermediate accumulation precision should be provided based on **end-to-end stability**, not only local numerical signatures.
  - The target precision is **format-dependent**, not one-size-fits-all.

# Conclusion

- TensorGauge provides:
  - a practical pre-silicon methodology,
  - controllable tensor-core numerical semantics,
  - and a bridge from microarchitecture to model behavior.
- **Final message:**
  Designing tensor cores requires joint consideration of **accuracy, stability, and PPA**, especially under low-precision training and inference. The conclusion states that intermediate accumulation precision should be provided according to model-level stability and that the target depends on data format.

# Q & A

Q & A ?

Thanks for your listening!

Contact: liuw24@mails.tsinghua.edu.cn

*This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFB4500101*