DGEMM on Integer Tensor Cores

Hiroyuki Ootomo
Tokyo Institute of Technology

NHR PerfLab Seminar Sep. 05, 2023

$ whoami

- Hiroyuki Ootomo
- Ph.D. candidate at Tokyo Institute of Technology
- Research:
  - Mixed-precision computing
  - Randomized numerical linear algebra
  - Quantum circuit simulation
  - HPC processors

Motivation

Question

Can deep learning processors be used for HPC applications?

- Deep learning hardware
  - Low- and mixed-precision
  - Matrix multiplication
- Usable in HPC apps?
  - Not easily

Myth 11: HPC Will Pivot to Low or Mixed Precision!

A high-performance language is nothing without proper data types, but high-precision operations such as fp64 come at a significant cost in terms of silicon area, energy and speed, according to Myth 6. Lowering this precision can save costs

S. Matsuoka, et al. (2022)
Myths and Legends in High-Performance Computing

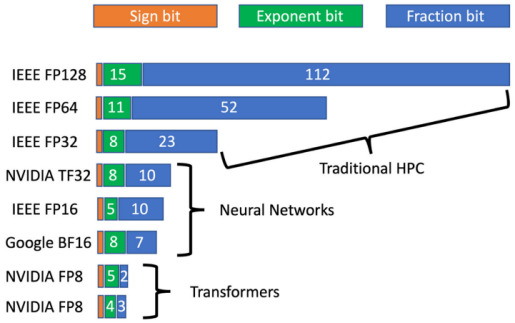


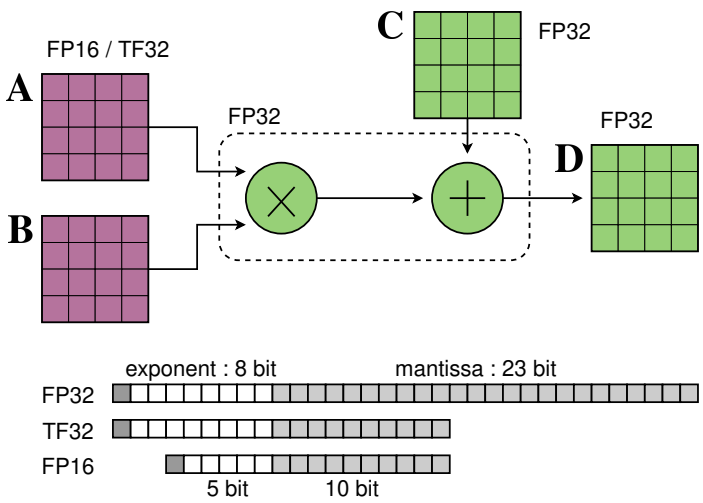
FIGURE 1. Overview of FP representations.

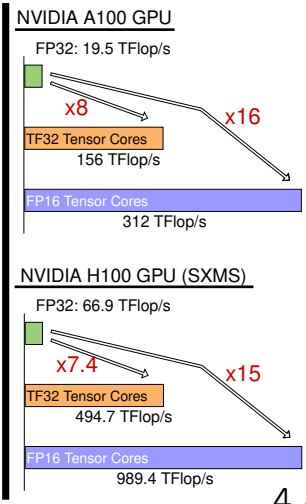
H. Ltaief, et al., (2022)
Responsibly Reckless Matrix Algorithms for HPC Scientific Applications

Outline

NVIDIA Tensor Core

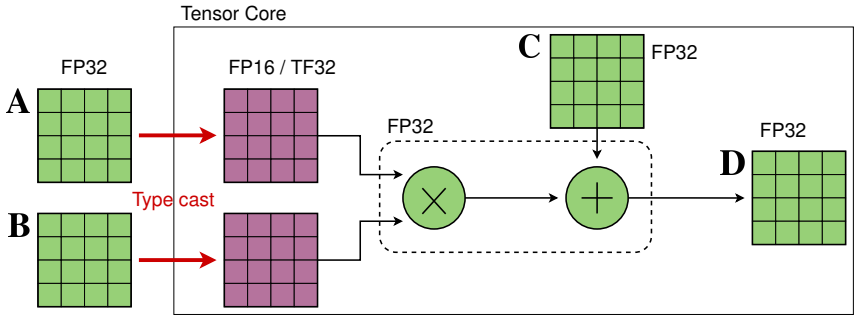
- Mixed-precision matrix multiplication and addition unit $(\mathbf{A} \cdot \mathbf{B} + \mathbf{C})$.
- On the latest NVIDIA GPUs.





Can we compute SGEMM on Tensor Cores?

- We need to convert the input FP32 matrices $(\mathbf{A}, \mathbf{B})$ to low-precision.

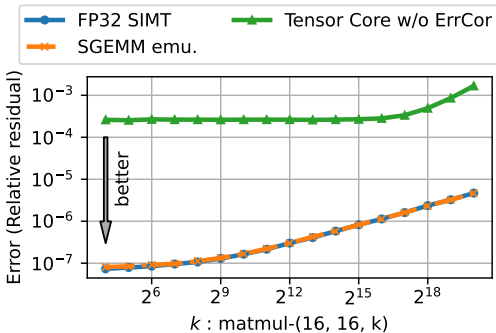


- The type conversion results in **loss of accuracy** in the resulting matrix
  - Almost the same as half-precision
- $\Rightarrow$ Our previous study proposes a method to refine the accuracy by preserving and utilizing the mantissa that is lost during type conversion.

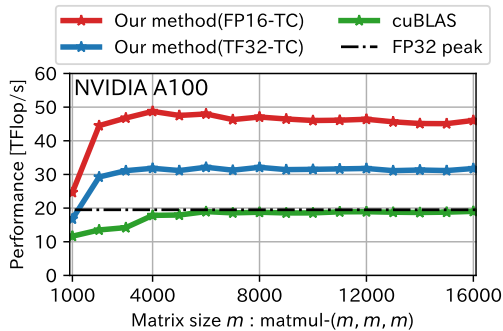
SGEMM emulation on Tensor Cores

- Matrix multiplication almost equivalent to single-precision on Tensor Cores.
- Faster than the theoretical peak throughput of FP32 computing units.

Accuracy

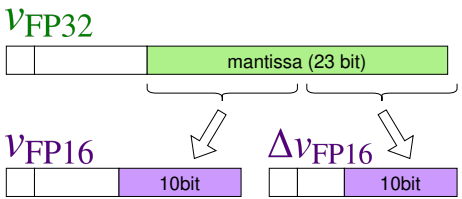


Throughput



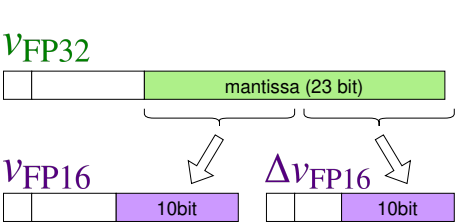
Ref: Ootomo and Yokota, *Recovering single precision accuracy from Tensor Cores while surpassing the FP32 theoretical peak performance*, IJHPCA, 2022

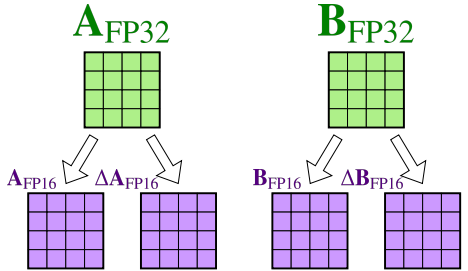
SGEMM emulation on Tensor Cores ($\mathbf{C}_{\mathrm{FP32}} \leftarrow \mathbf{A}_{\mathrm{FP32}} \cdot \mathbf{B}_{\mathrm{FP32}}$)



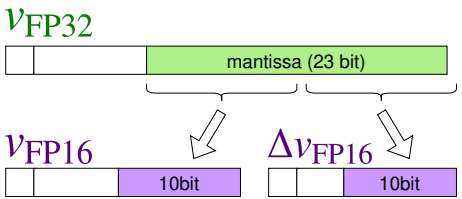
$$v_{\mathrm{FP32}} \approx v_{\mathrm{FP16}} + \Delta v_{\mathrm{FP16}}$$

SGEMM emulation on Tensor Cores ($\mathbf{C}_{\mathrm{FP32}} \leftarrow \mathbf{A}_{\mathrm{FP32}} \cdot \mathbf{B}_{\mathrm{FP32}}$)





SGEMM emulation on Tensor Cores ($C_{FP32} \leftarrow A_{FP32} \cdot B_{FP32}$)

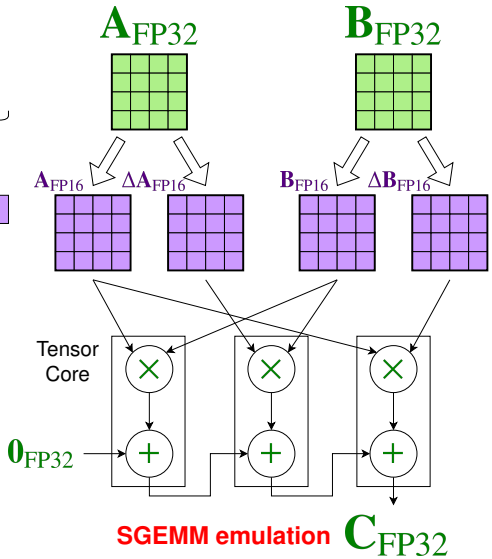


$$v_{FP32} \approx v_{FP16} + \Delta v_{FP16}$$

Scheme

$$C_{FP32} \approx A_{FP16} \bullet B_{FP16}$$
$$+\Delta A_{FP16} \bullet B_{FP16} + A_{FP16} \bullet \Delta B_{FP16}$$

with avoidance of RZ inside Tensor Cores



Can we emulate DGEMM in the same manner?

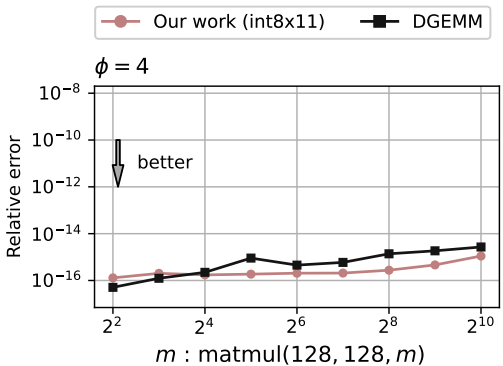
- **No**
  - The accumulation precision limits the emulation accuracy.
    - In the case of Tensor Cores, it is FP32.
  - If there were FP64 accumulation Tensor Core, we could emulate DGEMM
- But we can compute it using another method called the **Ozaki scheme**

Outline

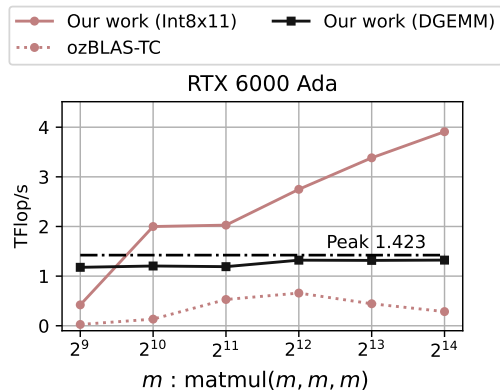
DGEMM emulation on Integer Tensor Cores

DGEMM emulation on Integer Tensor Cores using the Ozaki scheme

Accuracy



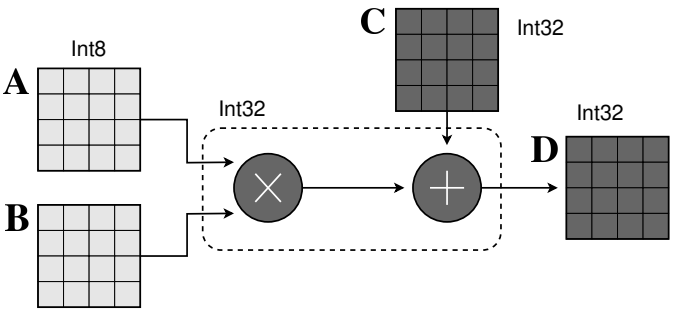
Throughput

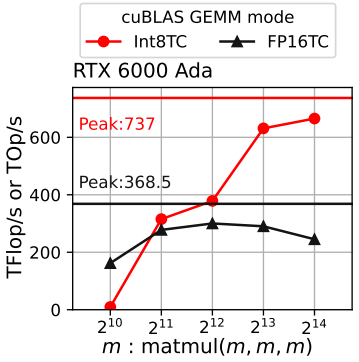


Ref: Ootomo, Ozaki, and Yokota, *DGEMM on Integer Matrix Multiplication Unit*, arXiv, 2023

DGEMM on Integer Tensor Cores

Integer Tensor Cores

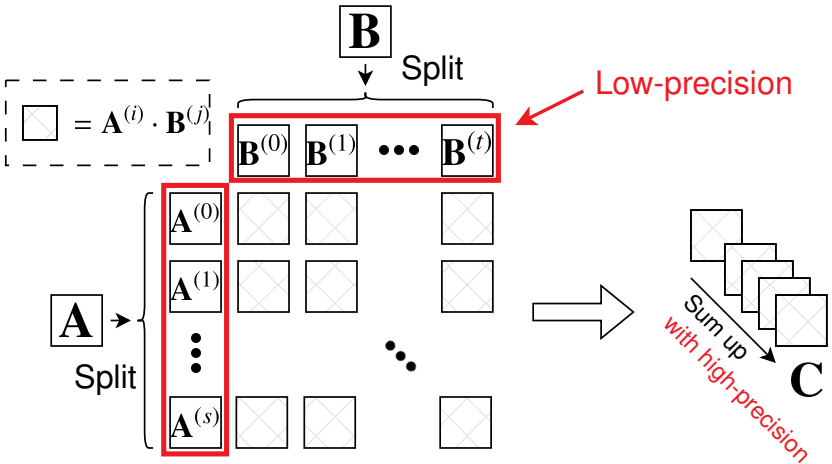




- The throughput is $2\times$ higher than FP16 Tensor Cores
- The inference computation in deep learning

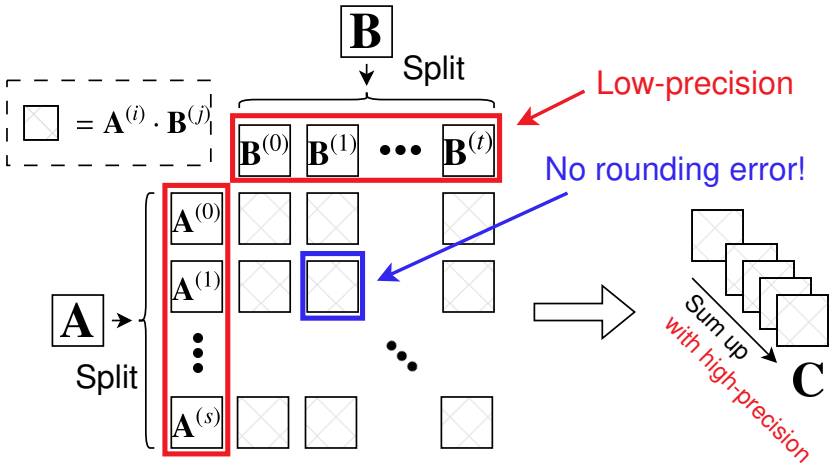
Ozaki scheme (Ozaki et al., 2012)

- The Ozaki scheme is a high-precision matrix multiplication algorithm using lower-precision arithmetic



Ozaki scheme (Ozaki et al., 2012)

- The Ozaki scheme is a high-precision matrix multiplication algorithm using lower-precision arithmetic



Ozaki scheme

Examle (in decimal number)

- $\mathbf{a} = \begin{bmatrix} a_1 & a_2 \end{bmatrix}$ where $a_1 = 1.11111111 \times 10^5, a_2 = 2.22222222 \times 10^3$
- Split mantissa by 3 digits

× **Similar method to the SGEMM emulation (Not the Ozaki scheme)**

$$\begin{array}{rlllll} a_1 = 1.11\times & 10^5 + 1.11\times & 10^2 + 1.11\times & 10^{-1} & & \\ a_2 = 2.22\times & 10^3 + 2.22\times & 10^0 + 2.22\times & 10^{-3} & & \\ \hline & \mathbf{a}^{(1)}+ & \mathbf{a}^{(2)}+ & & \mathbf{a}^{(3)} & \end{array}$$

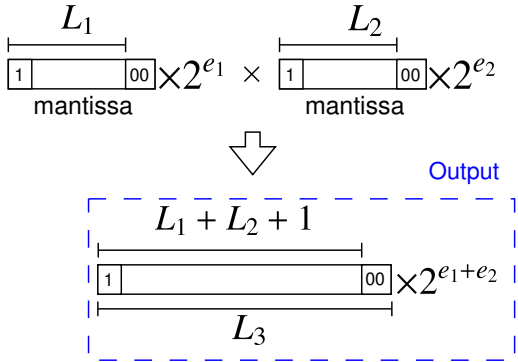
✓ **The Ozaki scheme**

$$\begin{array}{rlllll} a_1 = 1.11\times & 10^5 + 1.11\times & 10^2 + 1.11\times & 10^{-1} + 0.00\times & 10^{-4} & \\ a_2 = 0.02\times & 10^5 + 2.22\times & 10^2 + 2.22\times & 10^{-1} + 2.20\times & 10^{-4} & \\ \hline & \mathbf{a}^{(1)}+ & \mathbf{a}^{(2)}+ & \mathbf{a}^{(3)}+ & \mathbf{a}^{(4)} & \end{array}$$

- Split the mantissa while aligning the exponent.

Ozaki scheme

Rounding error in multiplication of two floating-point values

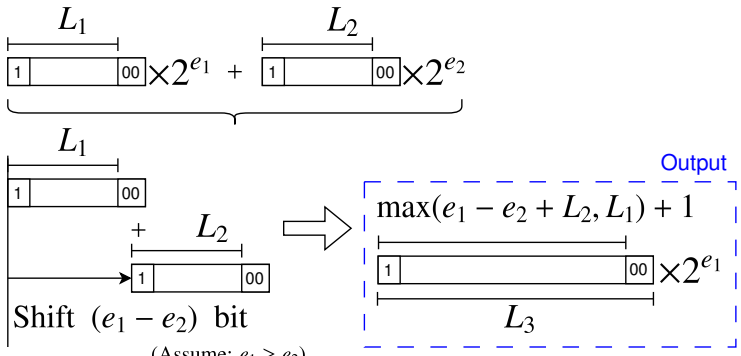


The rounding error occurs if $L_1 + L_2 + 1$ is larger than the mantissa length of the output format (L_3).

$\Rightarrow$ **No rounding error occurs if L_1 and L_2 are small enough.**

Ozaki scheme

Rounding error in addition of two floating-point values

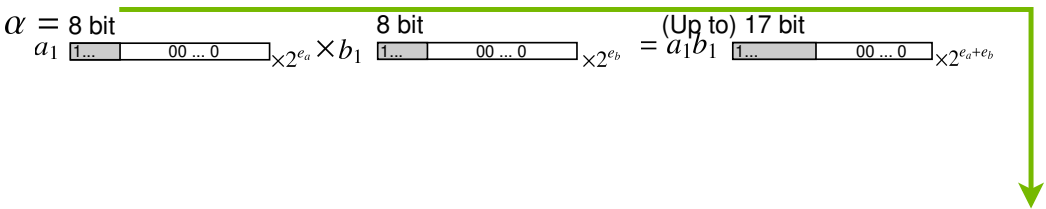


The rounding error occurs if $\max(e_1 - e_2 + L_2, L_1) + 1$ is larger than the mantissa length of the output format (L_3).

$\Rightarrow$ **No rounding error occurs if L_1, L_2, and $e_1 - e_2$ are small enough.**

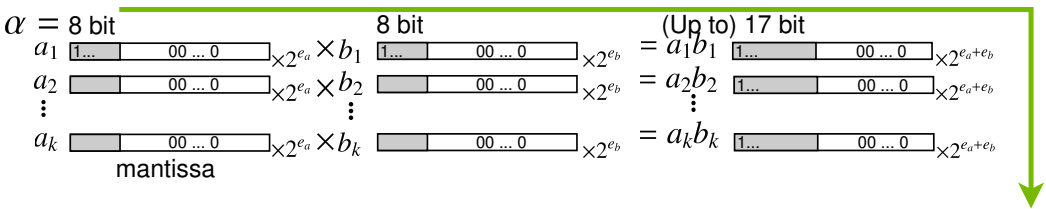
Ozaki scheme

- Consider an inner product of two length-k vectors $\mathbf{a}$ and $\mathbf{b}$.
- We can compute their inner product without rounding error if
  1. the mantissa lengths of all elements a_i, b_i are small enough
  2. the exponents of $a_i b_i$ for $i = 1..k$ are close enough
- Intuitive example: (not rigorous)
  (For simplicity, assume that all elements in a vector have the same exponent)



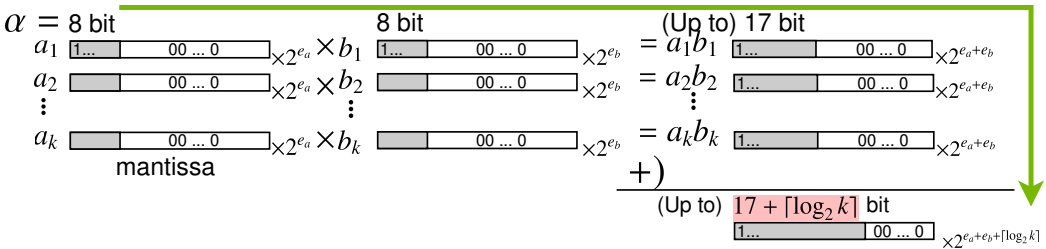
Ozaki scheme

- Consider an inner product of two length-k vectors $\mathbf{a}$ and $\mathbf{b}$.
- We can compute their inner product without rounding error if
  1. the mantissa lengths of all elements a_i, b_i are small enough
  2. the exponents of $a_i b_i$ for $i = 1..k$ are close enough
- Intuitive example: (not rigorous)
  (For simplicity, assume that all elements in a vector have the same exponent)



Ozaki scheme

- Consider an inner product of two length-k vectors $\mathbf{a}$ and $\mathbf{b}$.
- We can compute their inner product without rounding error if
  1. the mantissa lengths of all elements a_i, b_i are small enough
  2. the exponents of $a_i b_i$ for $i = 1..k$ are close enough
- Intuitive example: (not rigorous)
  (For simplicity, assume that all elements in a vector have the same exponent)



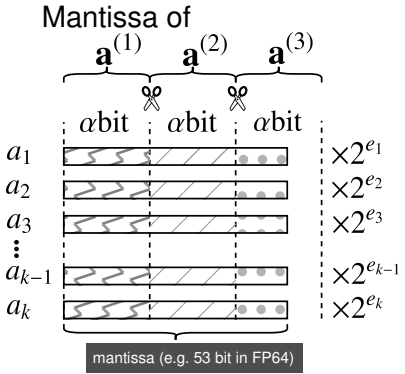
- If $17 + \lceil \log_2 k \rceil \leq$ the mantissa length of the output format, no rounding error occurs

Ozaki scheme

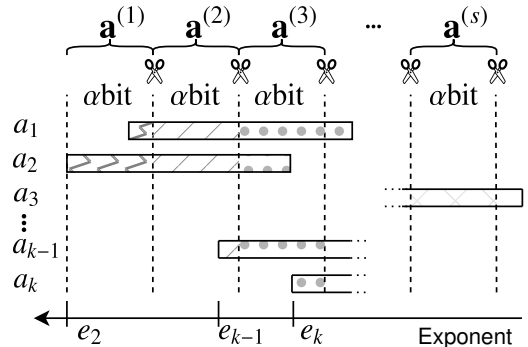
Split the mantissa so that no rounding error occurs during inner product computation

- $\begin{bmatrix} a_1 & a_2 & \cdots & a_k \end{bmatrix}$ is a row vector of matrix $\mathbf{A}$.

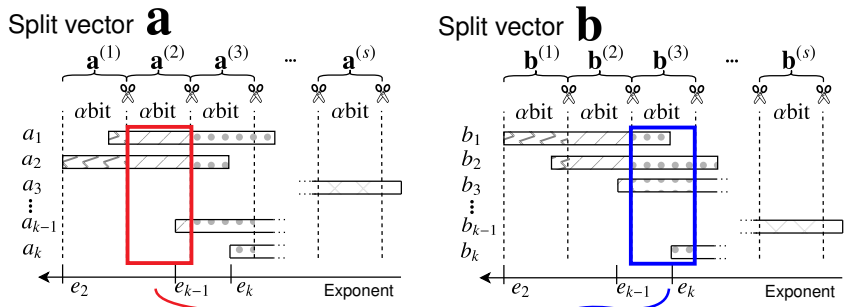
Similar method to the SGEMM emu.



The Ozaki scheme



Ozaki scheme



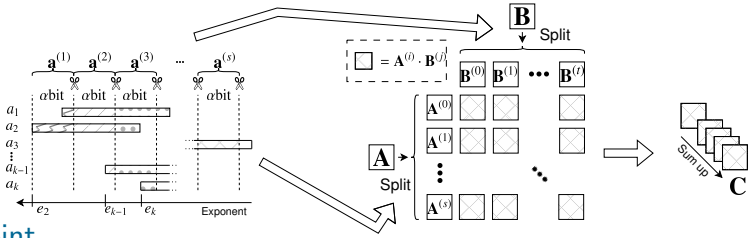
$$\mathbf{a} \cdot \mathbf{b} = \sum_{i,j} \mathbf{a}^{(i)} \cdot \mathbf{b}^{(j)}$$

We choose the mantissa length α appropriately so that $\mathbf{a}^{(i)} \cdot \mathbf{b}^{(j)}$ is computed without rounding error.

$$\alpha = \lceil (-\log_2 u - \log_2 k)/2 \rceil$$

where u is the unit round-off of the accumulator.

Ozaki scheme



Strong point

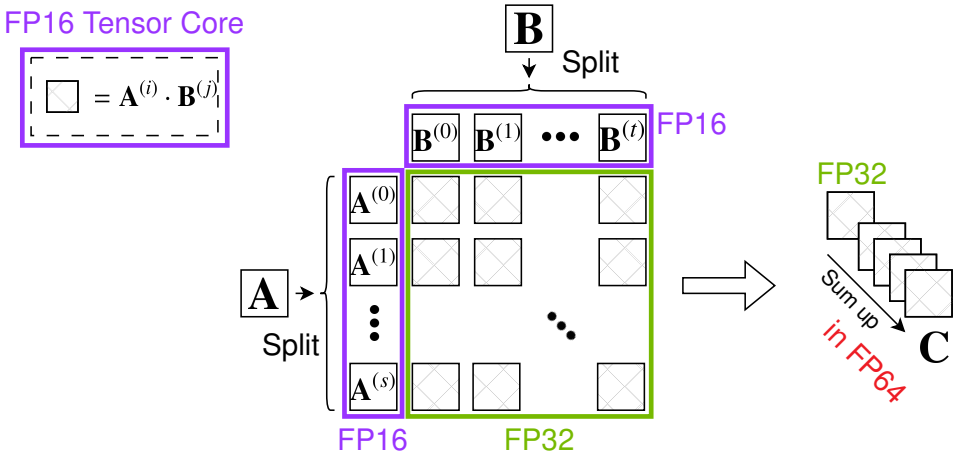
1. Compute high-precision GEMM by the accumulation of lower-precision GEMMs
2. Can be applied to arbitrary precision GEMM

Weak point

1. Requires additional memory space to store split matrices (typically large)
2. More splits are required when the exponent distribution is large

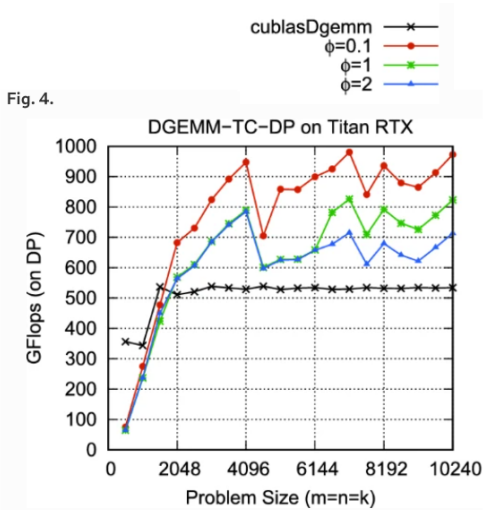
Ozaki scheme on FP16 Tensor Cores (by Mukunoki et al.)

- Use FP16 Tensor Cores for each matrix multiplication



Mukunoki et al. DGEMM Using Tensor Cores, and Its Accurate and Reproducible Versions, ISC 2020

Ozaki scheme on FP16 Tensor Cores (by Mukunoki et al.)

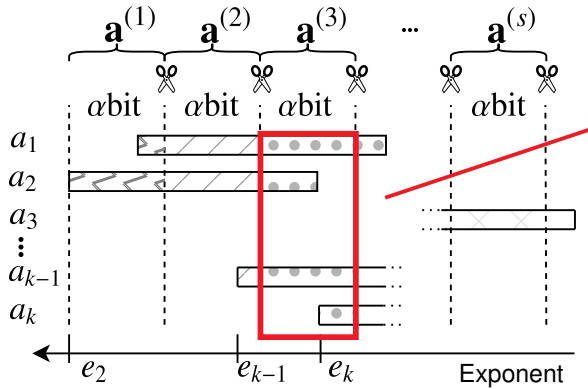


- Larger ϕ
  $\Rightarrow$ larger exponent distribution
  $\Rightarrow$ more splitting is required
  $\Rightarrow$ throughput degradation
- Faster than cublasDgemm

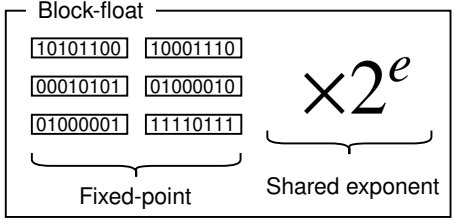
Problem

- FP16 is inefficient for the Ozaki scheme.
- The effective mantissa length is $5 \sim 7$ bits, whereas FP16 has 11 bit mantissa.

Ozaki scheme on Integer Tensor Cores



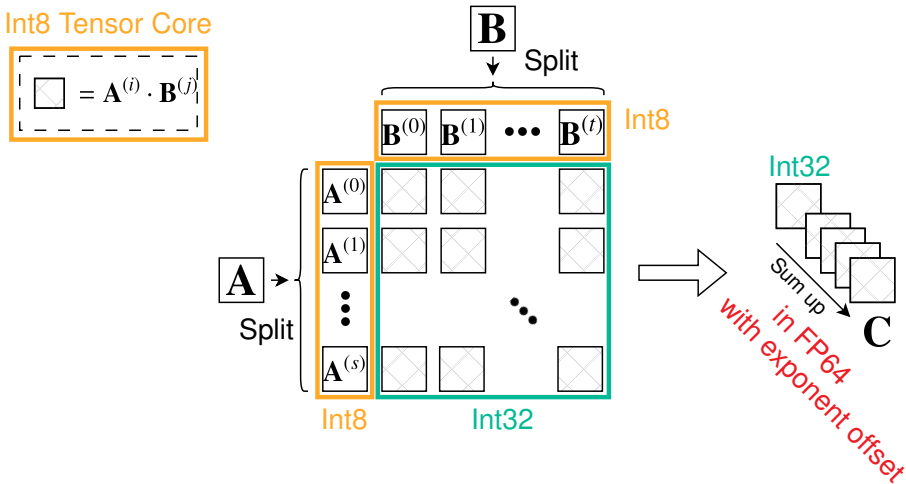
This can be obviously represented as block-float!



- $\mathbf{a}^{(i)} \cdot \mathbf{b}^{(j)}$ can be calculated by integer operations
- $\mathbf{A}^{(i)} \cdot \mathbf{B}^{(j)}$ can be calculated by integer Tensor Cores!

Ozaki scheme on Integer Tensor Cores (Our work)

- Use Int8 Tensor Cores for each matrix multiplication
  - Int8-input Int32-accumulation



Pros/Cons of using Integer Tensor Cores in the Ozaki scheme

Pros

- Less working memory usage than FP16 Tensor Cores
- Int8 Tensor Cores have higher throughput than FP16
- Higher efficiency in the actual mantissa usage
- Fewer computations are required than FP16 to obtain the same accuracy
- etc

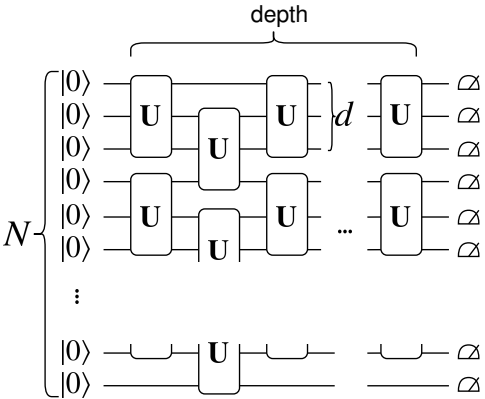
Cons

No significant Cons.

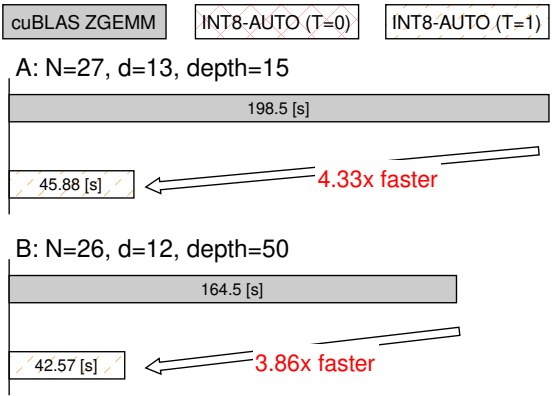
Details are in the paper.

Quantum circuit simulation using DGEMM emu.

Random Unitary Circuit



Computing time



- NVIDIA RTX A6000 Ada
- The accuracy is almost the same

Summary

Question

Can deep learning processors be used for HPC applications?

⇒Yes

- We can compute double-precision equivalent GEMM using integer matrix multiplication unit.
- More research is required to put the Ozaki scheme into practice.
  - because the Ozaki scheme does not emulate the floating-point arithmetic, whereas most HPC applications are conducted by floating-point.