Numerical Behavior of NVIDIA Tensor Cores

Massimiliano Fasi,[1] Nicholas J. Higham,[2] Mantas Mikaitis,[2] and Srikara Pranesh[2]

[1]Örebro University, Sweden

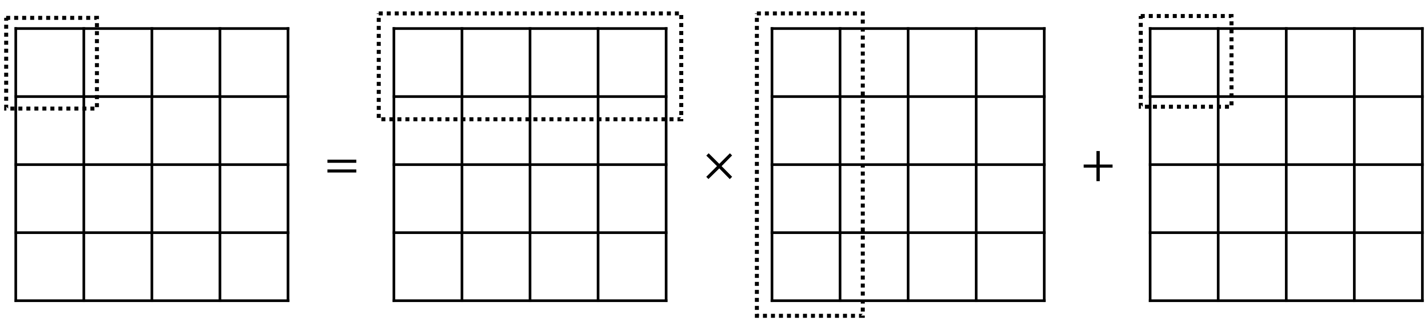
[2]University of Manchester, UK

Contact: *mantas.mikaitis@manchester.ac.uk*

Virtual talk for Numerical Algorithms Group (NAG)
20th May 2021

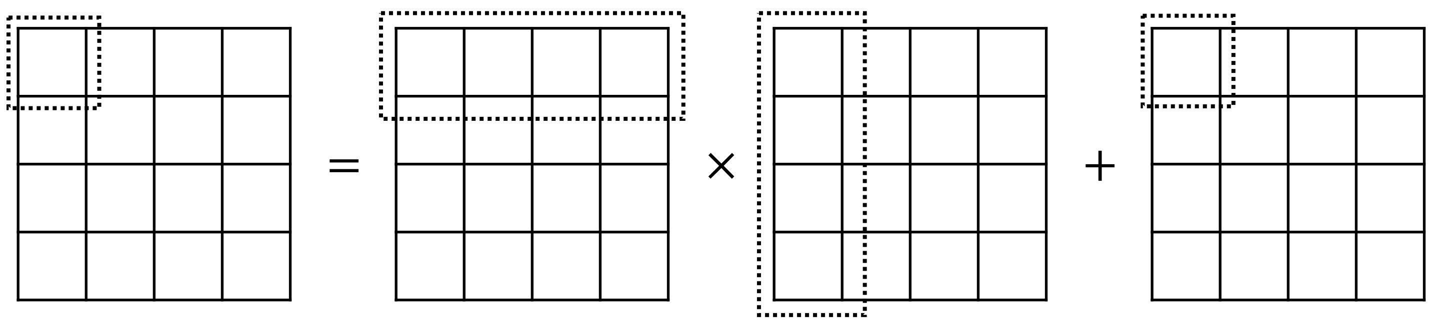
Matrix multiply-add (MMA) operation in hardware

- For many years we have $+$, $\times$, $\div$, FMA, $\sqrt{}$ in hardware
- Required by IEEE 754 floating-point (FP) standard
- Sometimes also 2^x, sin, cos, $rand$ … but *not required*
- Machine learning (ML) applications use MMA a lot
- Therefore, the MMA (or dot prod) operation is being added to most new hardware



Tensor Cores (NVIDIA MMA)

- NVIDIA V100 Tensor Cores provide: $D = AB + C$ on 4×4 matrices
- Performs 64 inherently mixed prec. FMAs per cycle
- Note that this is comprised of FMAs, but is not an FMMA
- Researchers try to use this for other problems in SC
- Precise numerical properties are not specified



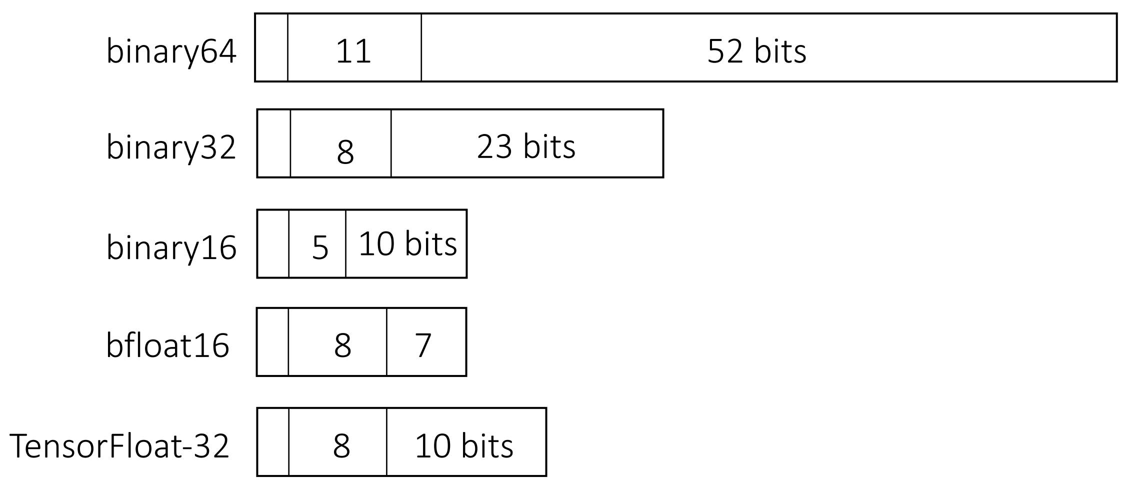
Hardware with MMA or dot products

Year of release	Device	Matrix dimensions	Input format	Output format	Reference
2016	Google TPU v2	$128 \times 128 \times 128$	bfloat16	binary32	(Google, 2020)
2017	Google TPU v3	$128 \times 128 \times 128$	bfloat16	binary32	(Google, 2020)
2017	NVIDIA V100	$4 \times 4 \times 4$	binary16	binary32	(NVIDIA, 2017)
2018	NVIDIA T4	$4 \times 4 \times 4$	binary16	binary32	(NVIDIA, 2018)
2019	Arm v8.6-A	$2 \times 4 \times 2$	bfloat16	binary32	(Arm Ltd., 2020)
2020	NVIDIA A100	$8 \times 8 \times 4$	bfloat16	binary32	(NVIDIA, 2020b)
		$8 \times 8 \times 4$	binary16	binary32	
		$4 \times 8 \times 4$	TensorFloat-32	binary32	
		$4 \times 2 \times 2$	binary64	binary64	

- 116 machines in the Nov. TOP500 list contain NVIDIA V100 and A100 devices with Tensor Cores
- Papers on dot prod or MMA in the ARITH conference:
  - 2017: 4/22
  - 2019: 5/29
  - 2020: 2/20

FP formats in the hardware of 2021

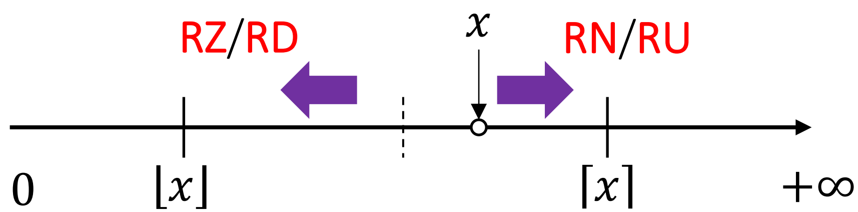
	binary16	bfloat16	TensorFloat-32	binary32	binary64
p	11	8	11	24	53
$e_{\max}$	15	127	127	127	1023
$e_{\min}$	-14	-126	-126	-126	-1022
ε	2^{-10}	2^{-7}	2^{-10}	2^{-23}	2^{-52}
$f_{\min}$	2^{-14}	2^{-126}	2^{-126}	2^{-126}	2^{-1022}
$s_{\min}$	2^{-24}	2^{-133}	2^{-136}	2^{-149}	2^{-1074}



Rounding in hardware

Rounding modes in IEEE 754:

- RN—Round to nearest, even on ties,
- RZ—round towards zero,
- RU—round (up) towards $+\infty$, and
- RD—round (down) towards $-\infty$.



Also stochastic and round to odd in some hardware.

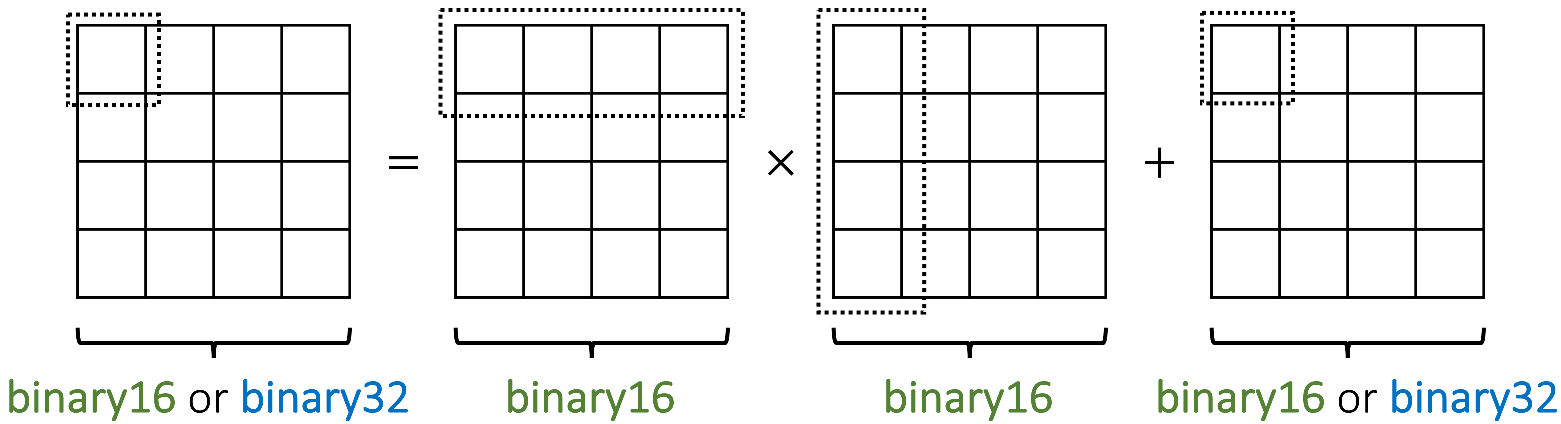
Motivation

- Low precision arrived with ML
- A wide array of old and new FP features in HW
- Some follow IEEE 754, some diverge
- New HW such as MMA
- Understanding, developing testsuites and documenting the low level FP details is beneficial for
  - Creating new numerical error analysis tools
  - Explaining experimental results
  - Simulating HW precisely
  - Informing upcoming hardware
  - Reproducibility (or explain why not achieved)

Many choices for prec. and rounding—
not straightforward to use/understand anymore

Tensor Core MMA operation in the V100

$$D = AB + C$$



$$d_{11} = a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} + a_{14}b_{41} + c_{11}$$

- Nonhomogeneous arithmetic operation (i/o formats different) + mixed prec.—both new features
- Assumption: all the elements of D are calculated in the same fashion (and all tensor cores are the same)

Reduction operations in IEEE 754

- *"Unlike the other operations in this standard, these operate on vectors of operands in one format and return a result in the same format. Implementations may associate in any order or evaluate in any wider format."* (IEEE-754 2019, [7])
- This provides very relaxed requirements for dot products
    1. Does not say if intermediate results must be normalized
    2. Does not specify the rounding mode
    3. No mention when rounding should happen
- Due to this, implementations will give different results

IEEE 754 is relaxed on dot prod implementations

Numerical specification of Tensor Cores

- *"Tensor Cores operate on FP16 input data with FP32 accumulation. The FP16 multiply results in a full precision product that is then accumulated using FP32 addition with the other intermediate products for a 4 × 4 × 4 matrix multiply."* (NVIDIA 2017, [4])
- *"Element-wise multiplication of matrix A and B is performed with at least single precision. When .ctype or .dtype is .f32, accumulation of the intermediate values is performed with at least single precision. When both .ctype and .dtype are specified as .f16, the accumulation is performed with at least half precision. The accumulation order, rounding and handling of subnormal inputs is unspecified."* (NVIDIA CUDA doc, [6])

Predictions from the documentation

$$\overbrace{d_{11}}^{\text{binary32}} = \underbrace{\underbrace{\underbrace{a_{11}}_{\text{binary16}} b_{11}}_{\substack{\text{Exact mult. (not rounded to binary16):} \\ \text{22 signif. bits, 6 exp. bits, 1 sign bit}}} + a_{12}b_{21} + a_{13}b_{31} + a_{14}b_{41} + \underbrace{c_{11}}_{\text{binary32}}}_{\text{binary32 5-operand adder}} \qquad (1)$$

Further questions about Tensor Cores

- Are subnormal inputs supported or flushed to zero?
- Can tensor cores produce subnormal numbers?
- Are the multiplications exact and the additions performed in binary32 arithmetic, resulting in four rounding errors for each element of D?
- In what order the four adds in (1) are performed?
- What rounding mode is used in (1)?
- Where is FP normalization done in tensor cores and what rounding mode is used?

A set of experiments on the **NVIDIA V100, T4** and **A100** graphics cards—here we focus on **V100**.

Subnormal numbers, test 1

$$d_{11} = \underbrace{a_{11}}_{2^{-24}}\overbrace{b_{11}}^{2^2} + \underbrace{a_{12}}_{0}\overbrace{b_{21}}^{0} + \underbrace{a_{13}}_{0}\overbrace{b_{31}}^{0} + \underbrace{a_{14}}_{0}\overbrace{b_{41}}^{0} + \underbrace{c_{11}}_{0}$$

✓ $d_{11} = 2^{-24} \times 2^2 = 2^{-22}$ if binary16 subnormals are supported, $d_{11} = 0 \times 2^2 = 0$ otherwise.

Subnormal numbers, test 2

$$d_{11} = \underbrace{a_{11}}_{0}\overbrace{b_{11}}^{0} + \underbrace{a_{12}}_{0}\overbrace{b_{21}}^{0} + \underbrace{a_{13}}_{0}\overbrace{b_{31}}^{0} + \underbrace{a_{14}}_{0}\overbrace{b_{41}}^{0} + \underbrace{c_{11}}_{2^{-149}}$$

✓ $d_{11} = 2^{-149}$ if binary32 subnormals are supported,
$d_{11} = 0$ otherwise.

Subnormal numbers, test 3

$$d_{11} = \underbrace{a_{11}}_{2^{-14}}\overbrace{b_{11}}^{2^{-1}} + \underbrace{a_{12}}_{0}\overbrace{b_{21}}^{0} + \underbrace{a_{13}}_{0}\overbrace{b_{31}}^{0} + \underbrace{a_{14}}_{0}\overbrace{b_{41}}^{0} + \underbrace{c_{11}}_{0}$$

$$d_{11} = \underbrace{a_{11}}_{2^{-14}}\overbrace{b_{11}}^{1} + \underbrace{a_{12}}_{0}\overbrace{b_{21}}^{0} + \underbrace{a_{13}}_{0}\overbrace{b_{31}}^{0} + \underbrace{a_{14}}_{0}\overbrace{b_{41}}^{0} + \underbrace{c_{11}}_{-2^{-15}}$$

✓ $d_{11} = 2^{-14} \times 2^{-1} = 2^{-14} - 2^{-15} = 2^{-15}$ if subnormals can be produced,
$d_{11} = 0$ otherwise.

Full subnormal number support

Exact product of binary16 inputs

$$d_{11} = \underbrace{a_{11}}_{1-2^{-11}}\overbrace{b_{11}}^{1-2^{-11}} + \underbrace{a_{12}}_{1-2^{-11}}\overbrace{b_{21}}^{1-2^{-11}} + \underbrace{a_{13}}_{1-2^{-11}}\overbrace{b_{31}}^{1-2^{-11}} + \underbrace{a_{14}}_{1-2^{-11}}\overbrace{b_{41}}^{1-2^{-11}} + \underbrace{c_{11}}_{0}$$

Here each product $(1-2^{-11})\times(1-2^{-11}) = 1-2^{-10}-2^{-22}$.
If not held exactly, binary16 would represent products as $1-2^{-10}$.
Therefore,

✓ $d_{11} = 4\times(1-2^{-10}-2^{-22})$ if the products are held exactly,
$d_{11} = 4\times(1-2^{-10})$ otherwise.

Binary16 products exact

Accuracy of the 5-operand adder in the Tensor Cores

Set the first row of A to 1 which reduces d_{11} to

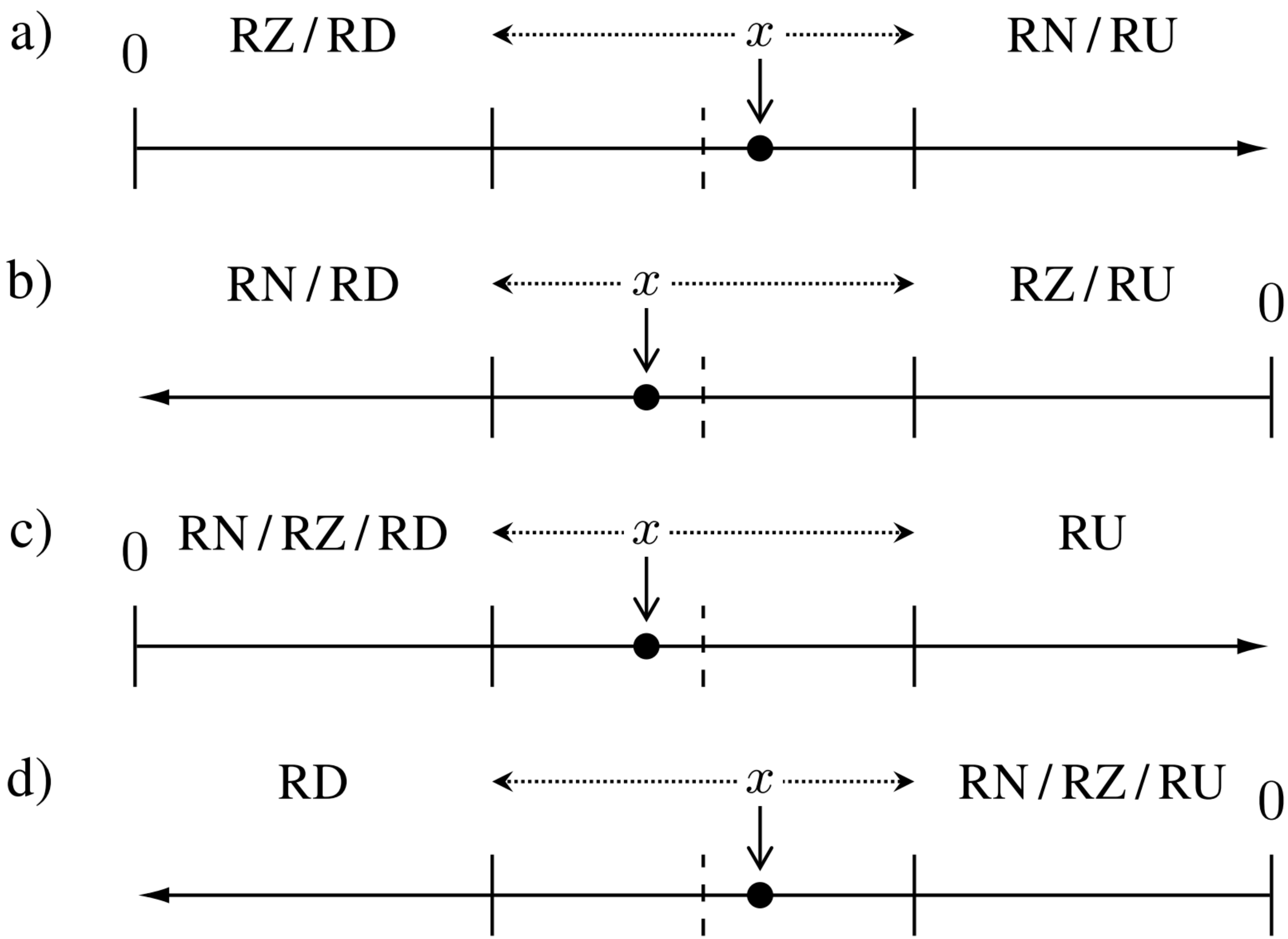
$$d_{11} = \underbrace{b_{11}}_{\substack{2^{-24}\\2^{-24}\\2^{-24}\\2^{-24}\\1}} + \underbrace{b_{21}}_{\substack{2^{-24}\\2^{-24}\\2^{-24}\\1\\2^{-24}}} + \underbrace{b_{31}}_{\substack{2^{-24}\\2^{-24}\\1\\2^{-24}\\2^{-24}}} + \underbrace{b_{41}}_{\substack{2^{-24}\\1\\2^{-24}\\2^{-24}\\2^{-24}}} + \underbrace{c_{11}}_{\substack{1\\2^{-24}\\2^{-24}\\2^{-24}\\2^{-24}}} \qquad (2)$$

- Assume that the 5-operand adder is working in binary32.
- The smallest value that can be added to 1 is $\varepsilon_{binary32} = 2^{-23}$.
- All permutations returned 1.

Therefore:

1. There are up to 4 rounding errors in each element of D,
2. 5-operand add starts from the largest magnitude addend.

Rounding modes



Rounding modes

Set the first row of A to 1 which reduces d_{11} to

$$d_{11} = \underbrace{b_{11}}_{2} + \overbrace{b_{21}}^{2^{-23}+2^{-24}} + \underbrace{b_{31}}_{0} + \underbrace{b_{41}}_{0} + \underbrace{c_{11}}_{0} \qquad (2)$$

- Assuming binary32 arithmetic, $\mathrm{RN}(2+x) > 2$ if $x > 2^{-23}$ whereas $\mathrm{RZ}(2+y) > 2$ if $y \geq 2^{-22}$.
- The choice $b_{21} = (3/4)\times 2^{-22}$ is such that $2^{-23} < b_{21} < 2^{-22}$, therefore $\mathrm{RN}(b_{11} + b_{21}) = \mathrm{RU}(b_{11} + b_{21}) = 2 + 2^{-22}$, whereas $\mathrm{RZ}(b_{11} + b_{21}) = \mathrm{RD}(b_{11} + b_{21}) = 2$.
- Tensor cores return 2, which means the rounding mode is either RZ or RD.
- Running with the inputs negated (symmetrical with respect to 0) -2 is returned, which means the rounding mode is RZ.

Normalization of floating-point values

In IEEE 754, a normalized floating-point value with a sign bit s, precision p, exponent e and a significand $m < 2^{p-1}$ has the form

$$-1^{s} \times 2^{e} \times (1 + m \cdot 2^{1-p}), \text{ where}$$

$$1 \le M := (1 + m \cdot 2^{1-p}) < 2 \text{ is a normalized significand.}$$

Set the first row of A to 1 which reduces d_{11} to

$$d_{11} = b_{11} + b_{21} + b_{31} + b_{41} + c_{11}. \qquad (2)$$

Are intermediate results in (2) normalized (each sum)?

Normalization of floating-point values

$$d_{11} = \underbrace{b_{11}}_{2^{-24}} + \underbrace{b_{21}}_{2^{-24}} + \underbrace{b_{31}}_{2^{-24}} + \underbrace{b_{41}}_{2^{-24}} + \underbrace{c_{11}}_{1-2^{-24}} \qquad (2)$$

- We know that the 5-operand adder starts from c_{11} here.
- $c_{11} + 2^{-24} = 1$ in binary32 – this would cause the significand to be denormalized.
- After normalization, $\mathrm{RZ}(1 + 2^{-24}) = 1$ in binary32.
- If there is no normalization, $\mathrm{RZ}(1 + 2^{-24}) = 1 + 2^{-24}$ because there is an extra bit.
- Tensor cores return $d_{11} = 1 + 2^{-24} + 2^{-24}$, meaning that only the final result is normalized.

Monotonicity of multi-operand add hardware

Floating-point sum is monotonic if

$$x_1 + \cdots + x_n \leq y_1 + \cdots + y_n \text{ when }$$
$$x_i \leq y_i \text{ for all } 1 \leq i \leq n.$$

Set the first row of A to 1 which reduces (1) to

$$d_{11} = \underbrace{b_{11}}_{2^{-24}} + \underbrace{b_{21}}_{2^{-24}} + \underbrace{b_{31}}_{2^{-24}} + \underbrace{b_{41}}_{2^{-24}} + \underbrace{c_{11}}_{\substack{1-2^{-24} \\ \\ 1}}. \tag{2}$$

With tensor cores we get

$d_{11} = 1 + 2^{-23}$, when $c_{11} = 1 - 2^{-24}$,

$d_{11} = 1$, when $c_{11} = 1$.

Not monotonic around powers of 2

Other results

V100:

- The accumulation in the 5-term add is not pure binary32
- Extra 3 bits for overflows (left side), since no intermediate normalization

T4:

- Same properties as the V100, except one extra bit on the right of the significand for alignment

A100:

- All properties in binary16, bfloat16 and TensorFloat-32 are the same as T4

Conclusion

- MMA or dot prod operations are being added to HW
- Tensor cores provide MMA in GPUs—present in the TOP500
- Numerical features of these units are not standard
- This work demonstrates a method to explore such hardware
- With some mild assumptions, we find main numerical features of the NVIDIA V100/T4/A100 GPU tensor cores
- This is useful both in developing software and in building new MMA hardware
- Future work could involve better automation of testing
- OA paper: https://peerj.com/articles/cs-330/
- Code: https://mmikaitis.github.io/software

References

1. M. Fasi, N. J. Higham, M. Mikaitis, and S. Pranesh. Numerical Behavior of the NVIDIA Tensor Cores. PeerJ Comput. Sci. 7:e330. Feb. 2021.
2. P. Blanchard, N. J. Higham, F. Lopez, T. Mary, and S. Pranesh. Mixed Precision Block Fused Multiply-Add: Error Analysis and Application to GPU Tensor Cores. SIAM J. Sci. Comput., 42(3). May 2020.
3. B. Hickmann and D. Bradford. Experimental Analysis of Matrix Multiplication Functional Units. 26th IEEE Symposium Comput. Arithmetic. Kyoto, Japan. Jun. 2019.
4. NVIDIA Volta microarch. whitepaper: https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf
5. NVIDIA Turing microarch. whitepaper: https://www.nvidia.com/content/dam/en-zz/Solutions/design-visualization/technologies/turing-architecture/NVIDIA-Turing-Architecture-Whitepaper.pdf
6. https://docs.nvidia.com/cuda/parallel-thread-execution/index.html#warp-level-matrix-instructions-mma
7. 754-2019 - IEEE Standard for Floating-Point Arithmetic https://ieeexplore.ieee.org/document/8766229
8. NVIDIA Ampere microarch. whitepaper: https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/nvidia-ampere-architecture-whitepaper.pdf