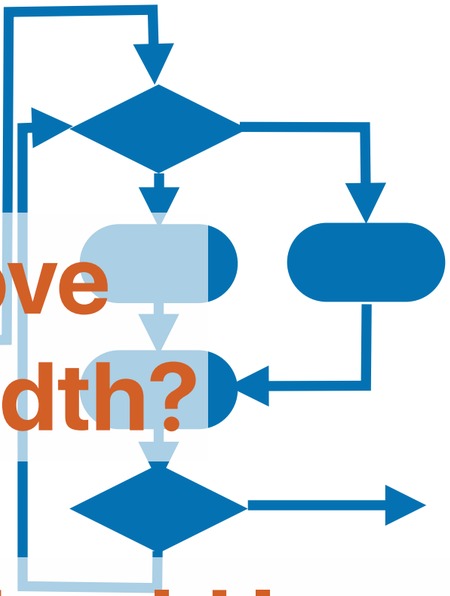
# Hardware accelerators

Hung-Wei Tseng

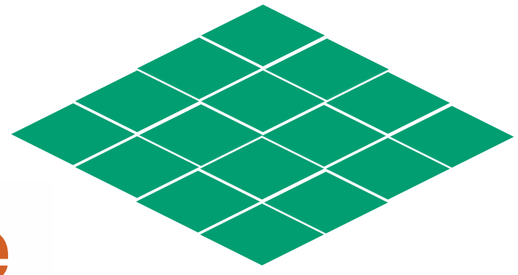
# Recap: From the software perspective


Should I improve memory bandwidth?

Should I use NDP?

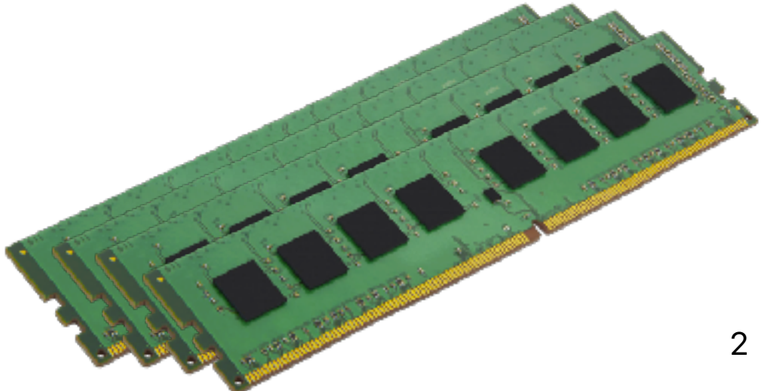


Should I use accelerators?



Source “data"

“Data” structures



Algorithms

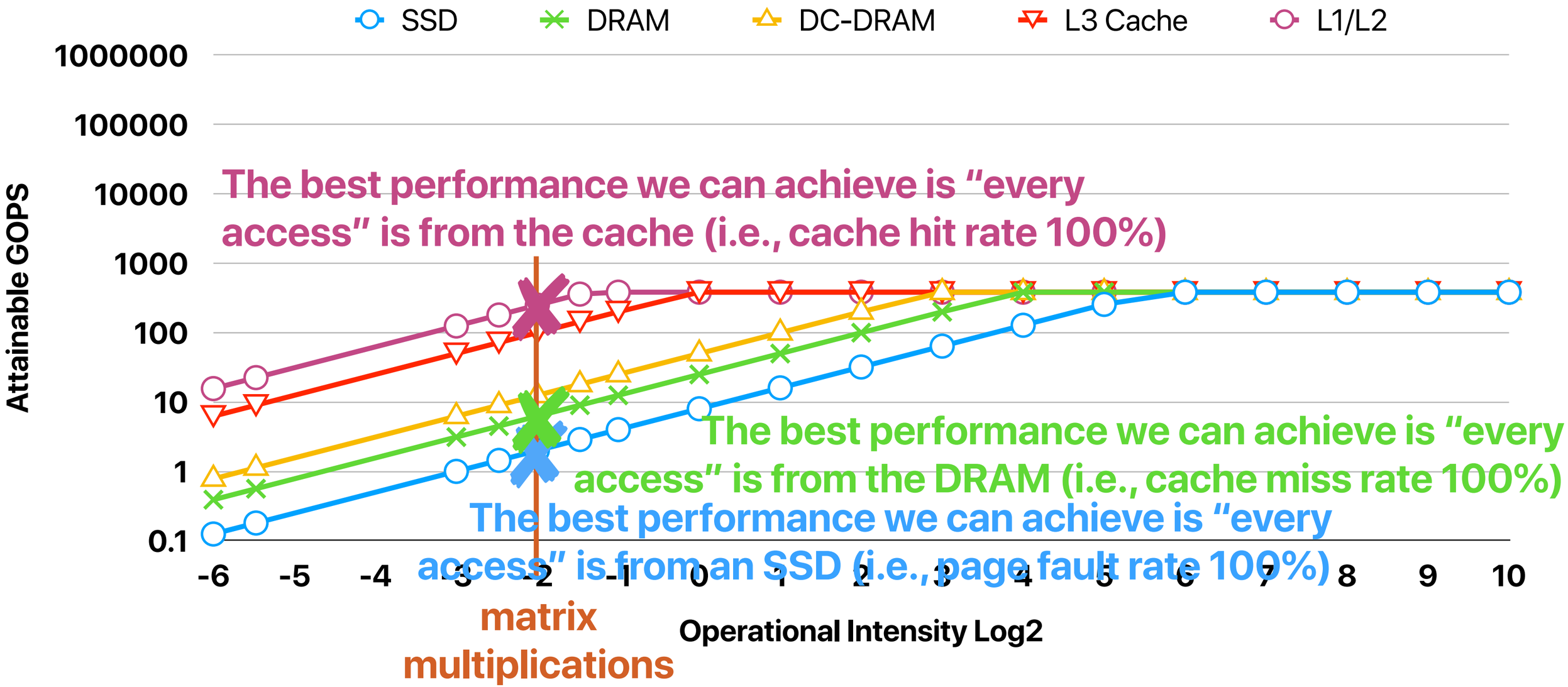

Result

# Recap: the roofline model

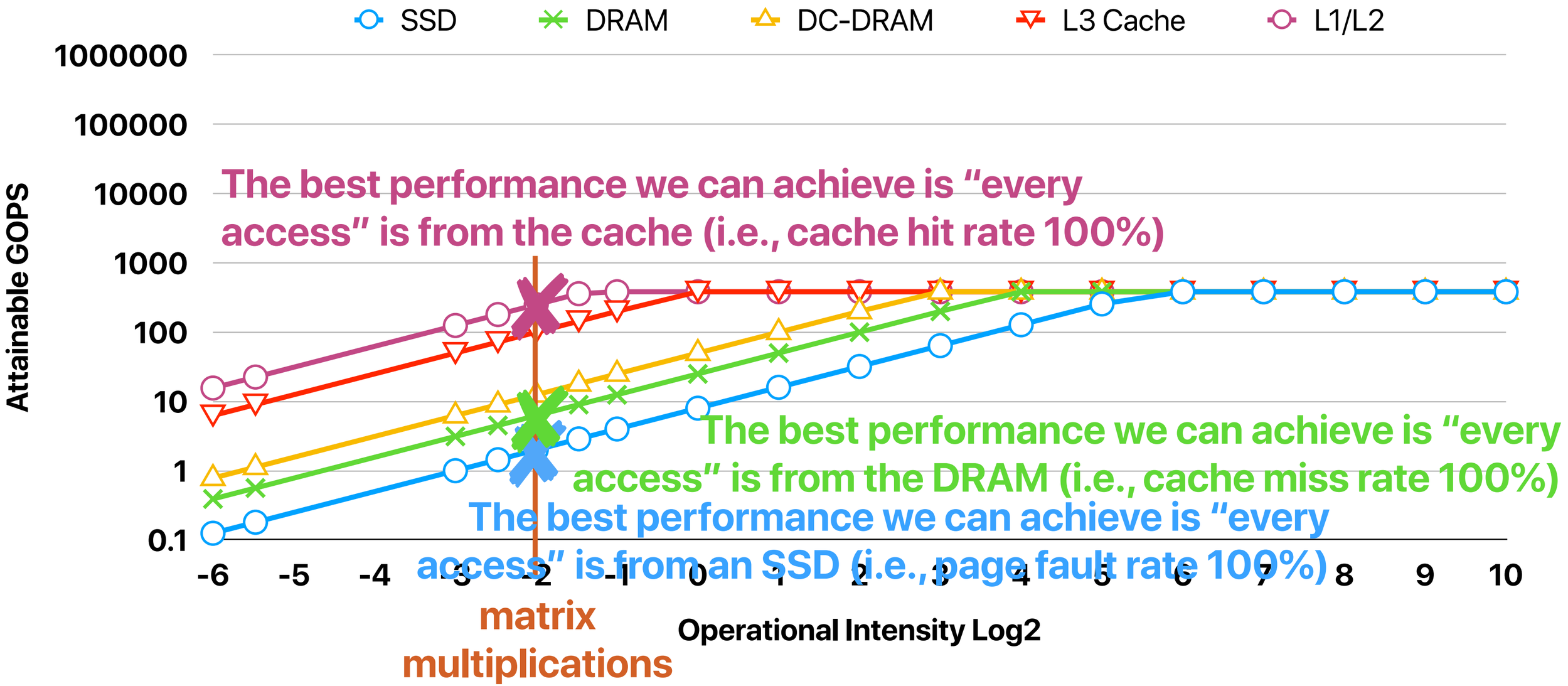
- The speed of computation — determined by the "OPS" (operations per second) of the processing unit
- The speed of data supply — determined by the "effective bandwidth" of the data path
- At any point, the slower one determines the performance
  - $$\text{Attainable GFlops/sec} = min \left\{ \begin{array}{l} \text{Peak Floating-Point Performance} \\ \text{Peak Memory Bandwidth} \times \text{Operational Intensity} \end{array} \right.$$

Samuel Williams, Andrew Waterman, and David Patterson. 2009. Roofline: an insightful visual performance model for multicore architectures. Commun. ACM 52, 4 (April 2009), 65–76.

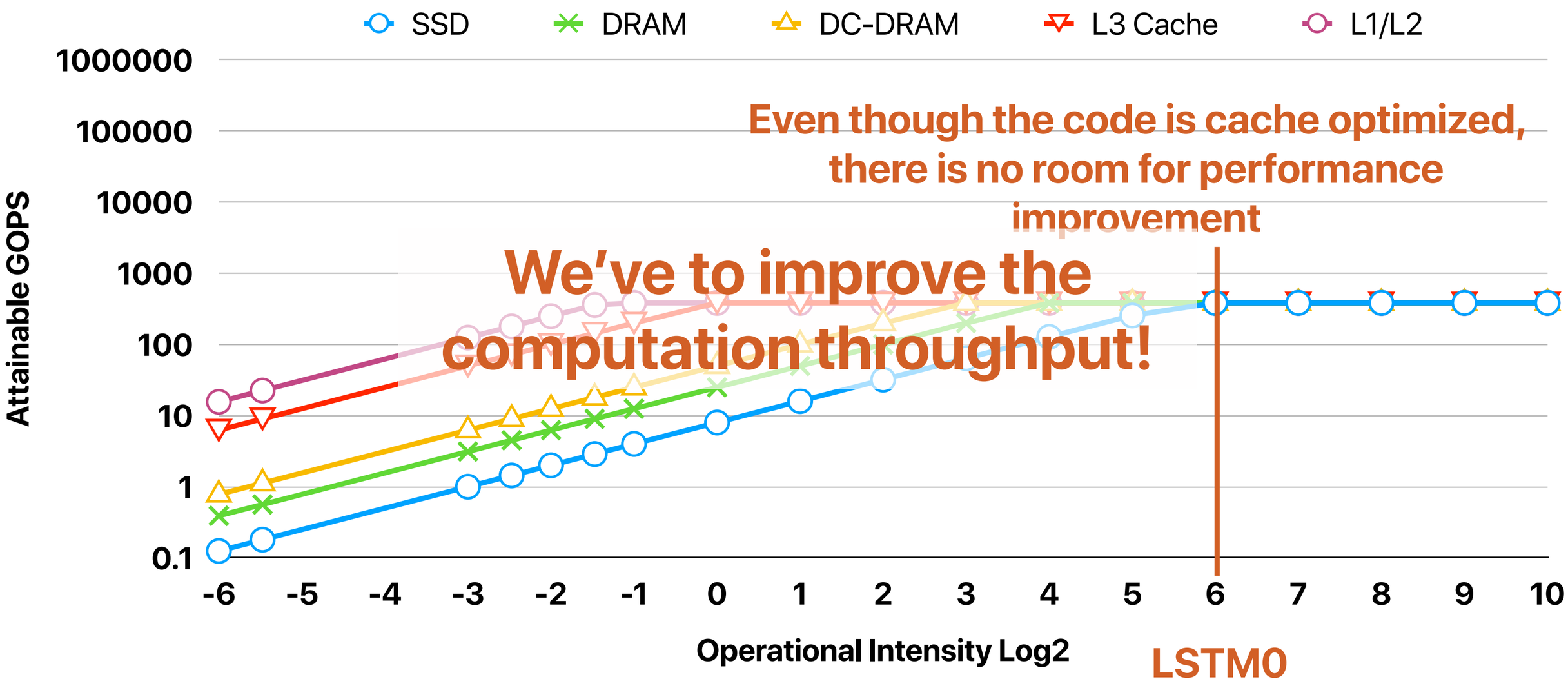
# Recap: the roofline of a CPU-based system



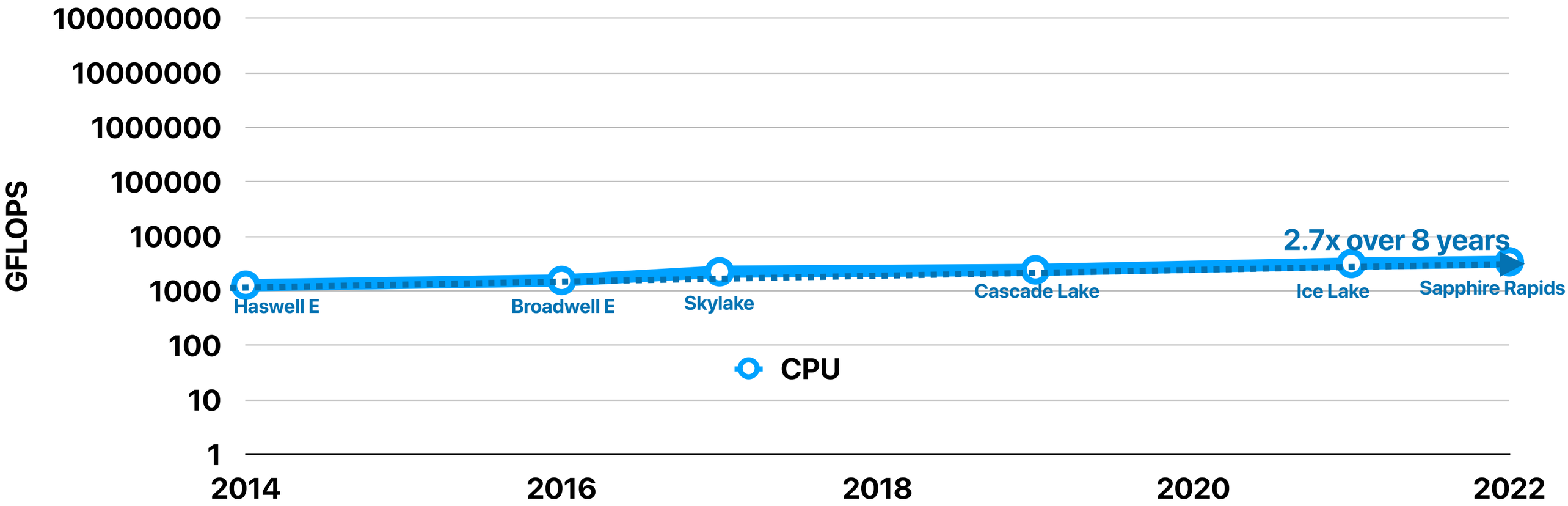
# Recap: the roofline of a CPU-based system



# Recap: the roofline of a CPU-based system

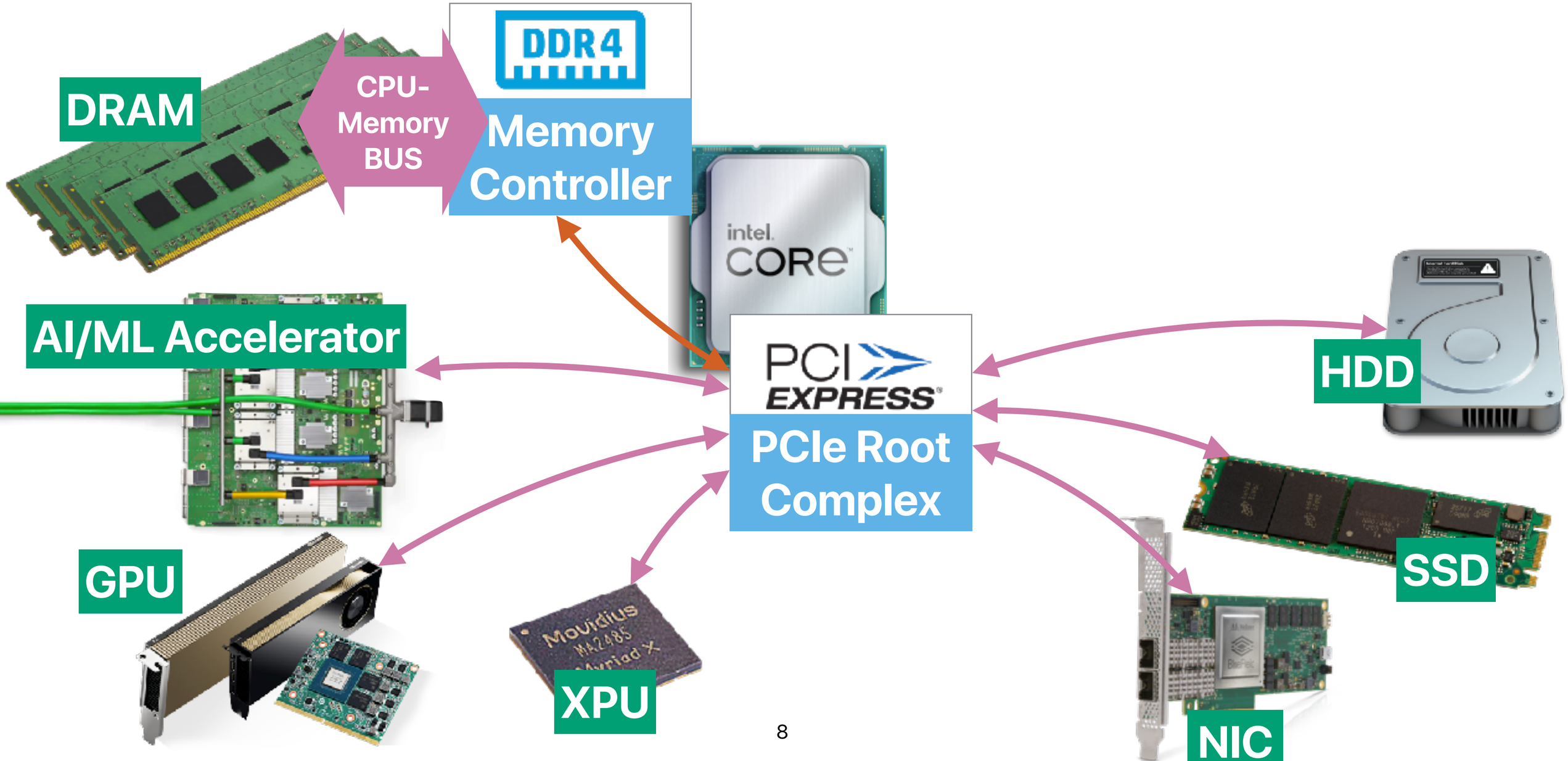


# CPU improvement is slow due to the broken Dennard scaling

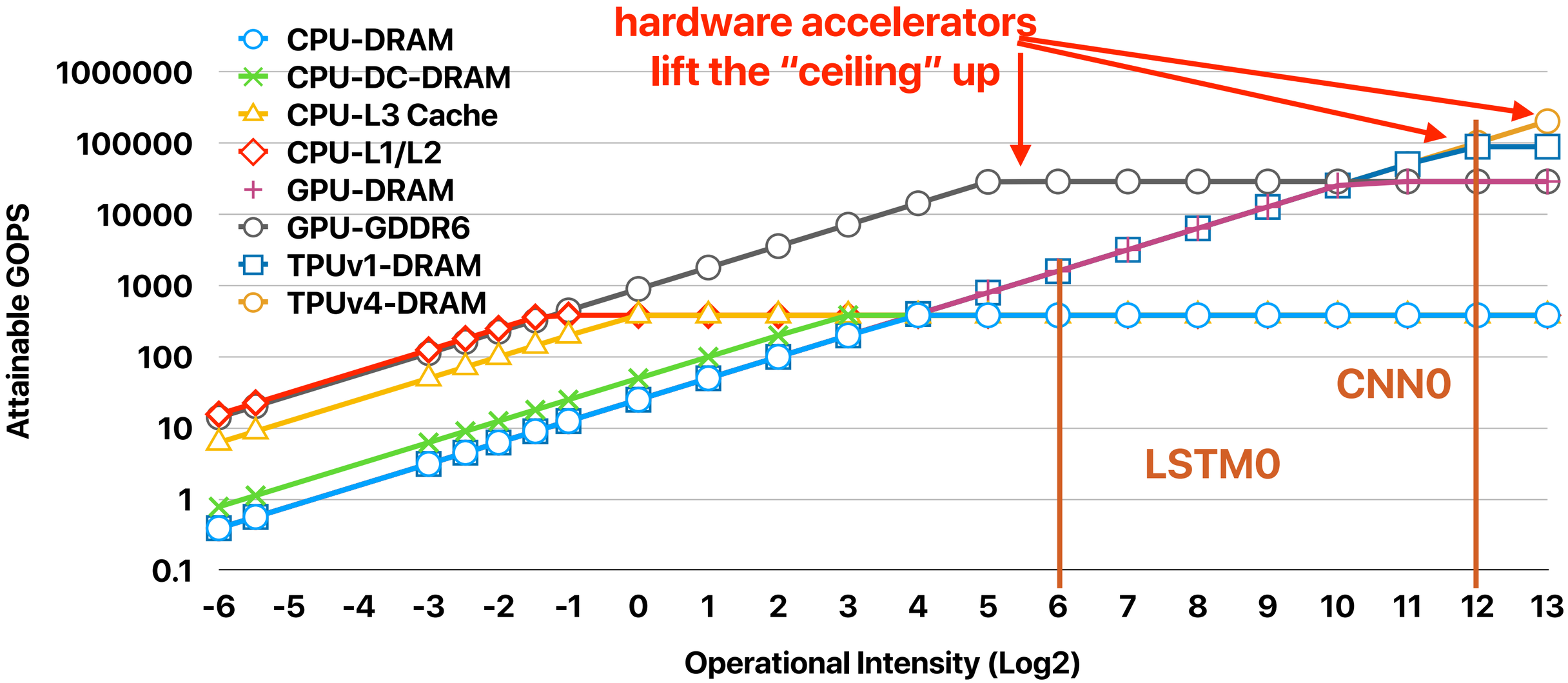



https://ourworldindata.org/grapher/artificial-intelligence-training-computation

# Recap: the landscape of heterogeneous computing



# Recap: the rooflines of modern systems



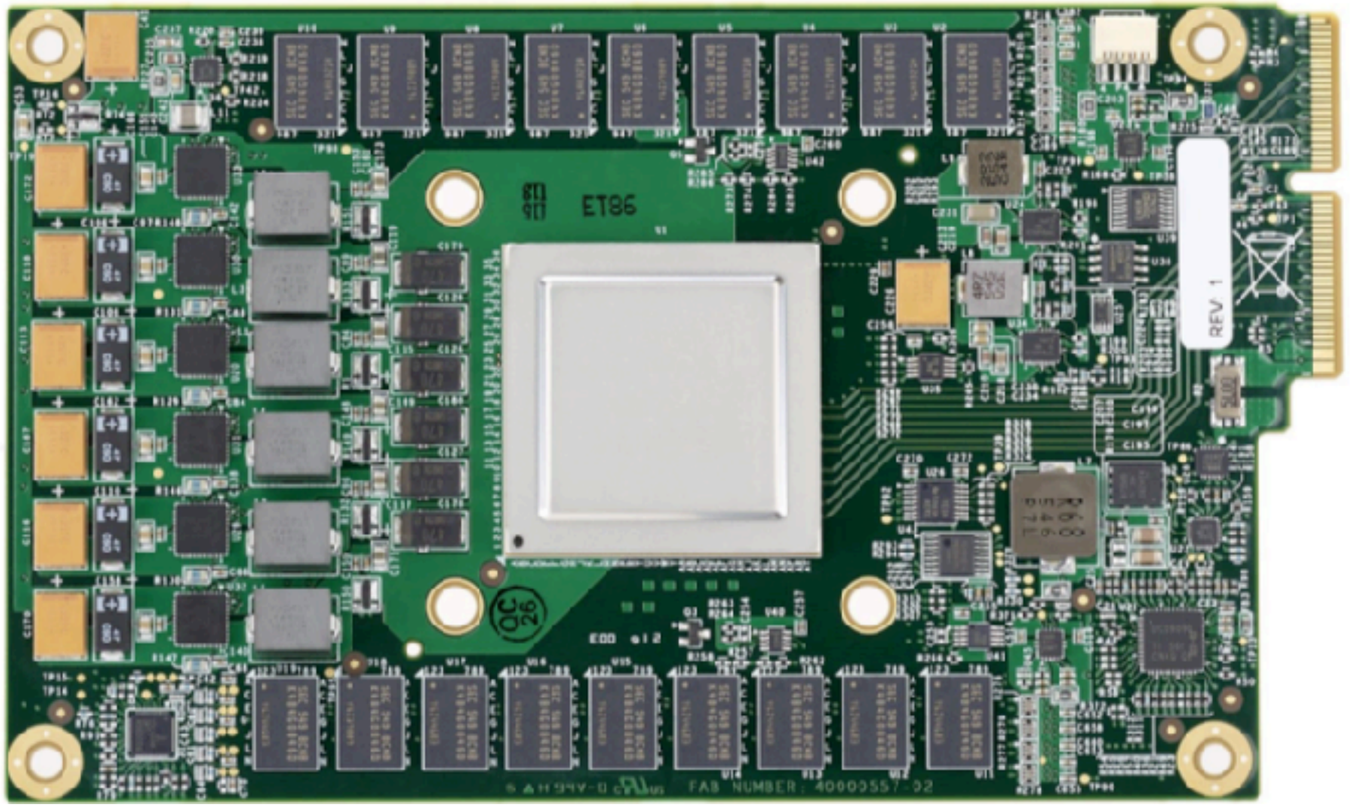
# Outline

- Tensor processing units (cont.)
- Tensor cores and ray tracing units
- FPGAs

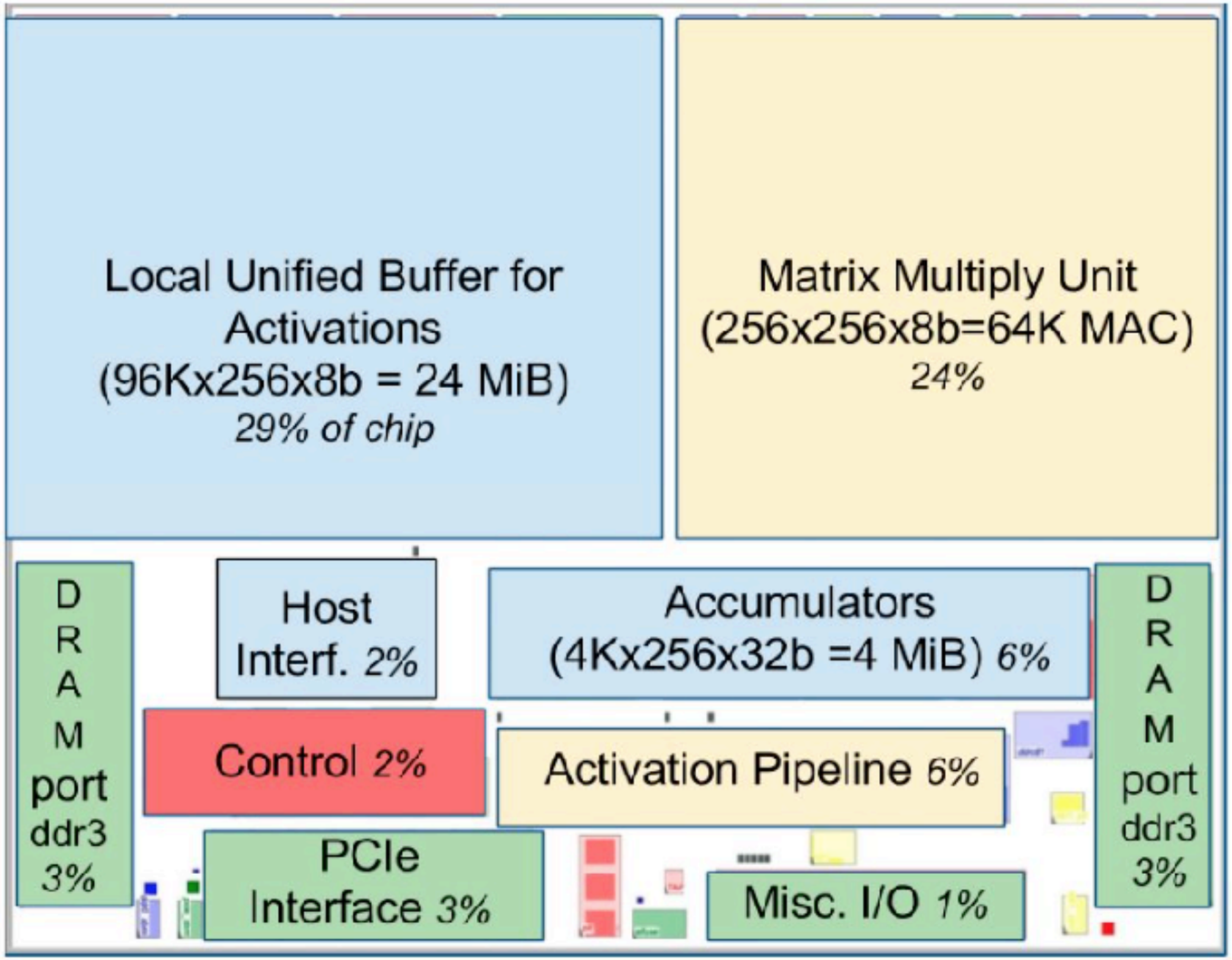
# Example of a hardware accelerator — Tensor Processing Unit


- N. Jouppi, C. Young, N. Patil and D. Patterson. Motivation for and Evaluation of the First Tensor Processing Unit. In IEEE Micro, vol. 38, no. 3, pp. 10-19. 2018
- Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Daniel Killebrew, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon. In-Datacenter Performance Analysis of a Tensor Processing Unit. In ISCA '17. 2017.

# What the “first” generation of TPU looks like



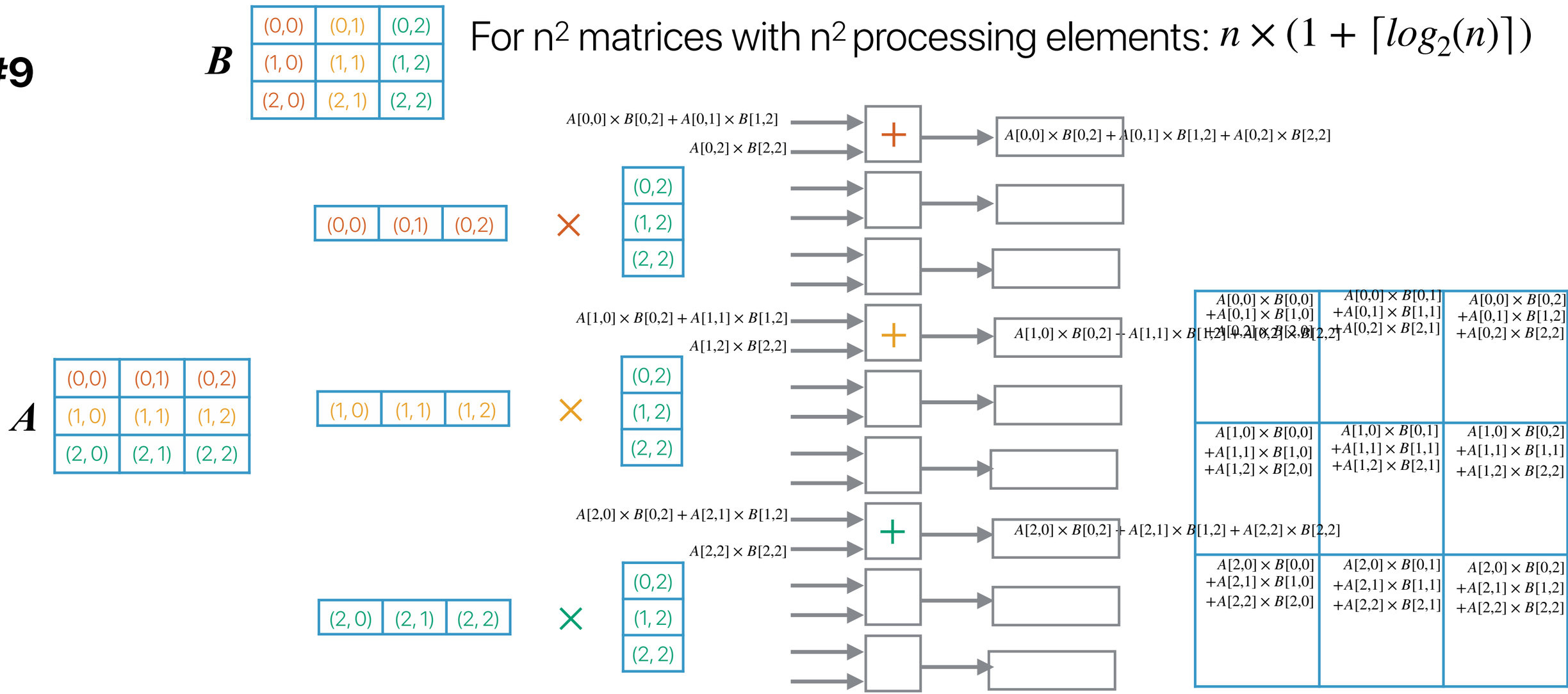
# TPU Floorplan



# Vector processing for MM

#9

For n² matrices with n² processing elements: $n \times (1 + \lceil log_2(n) \rceil)$

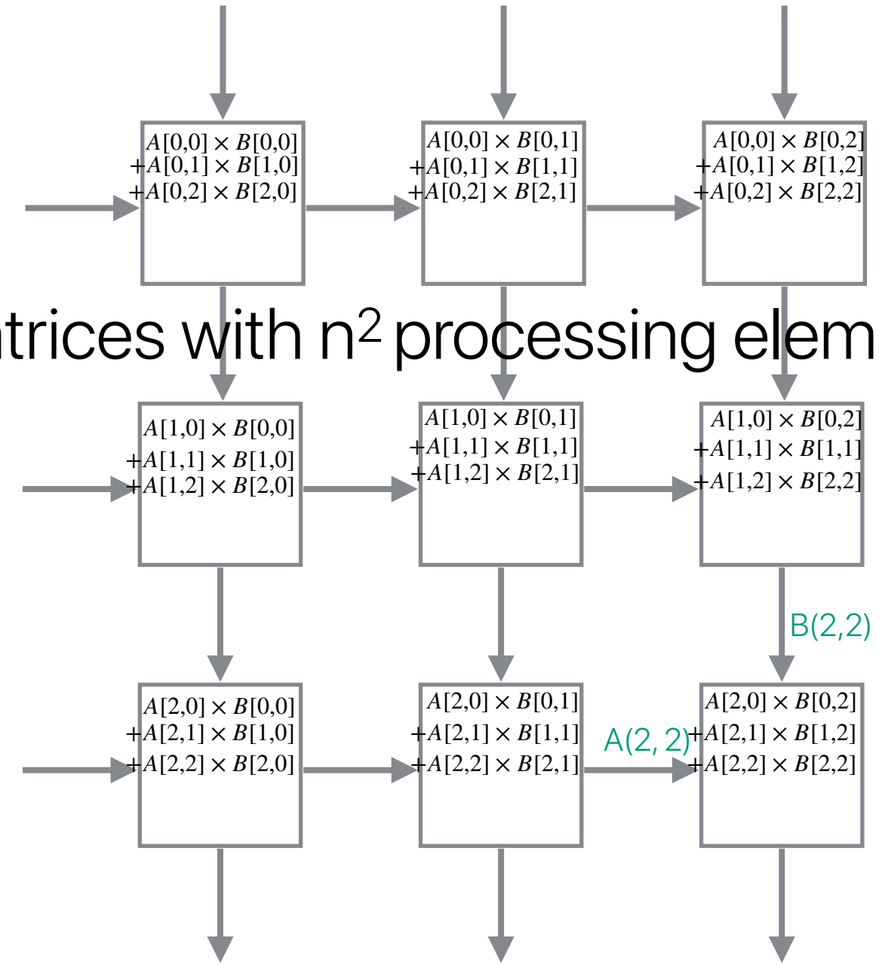


# Systolic Arrays used by AI/ML Accelerators

#7

***B***

| | | |
|---|---|---|
| (0,0) | (0,1) | (0,2) |
| (1, 0) | (1, 1) | (1, 2) |
| (2, 0) | (2, 1) | (2, 2) |




For n² matrices with n² processing elements: $3 \times n - 2$

***A***

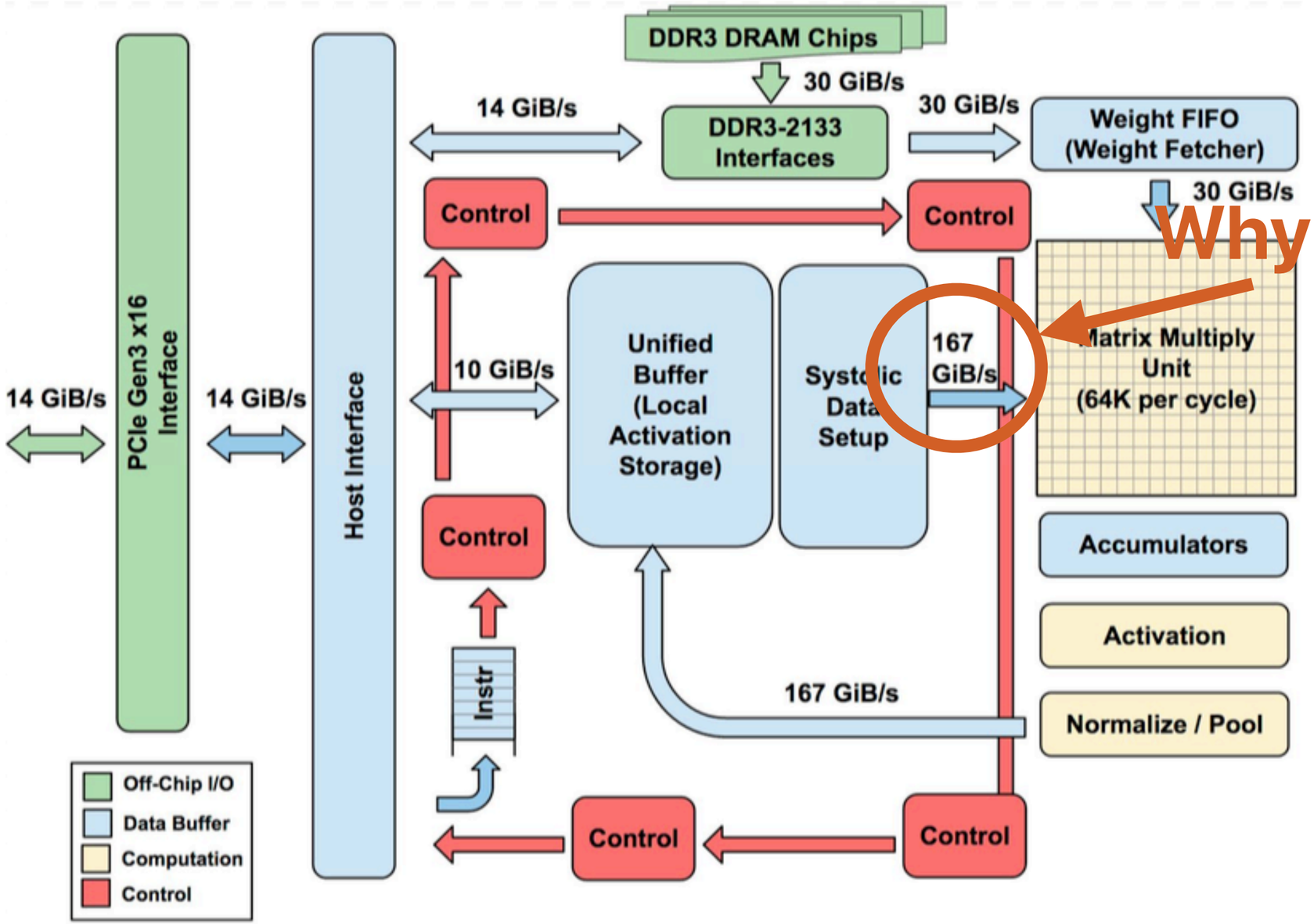
| | | |
|---|---|---|
| (0,0) | (0,1) | (0,2) |
| (1, 0) | (1, 1) | (1, 2) |
| (2, 0) | (2, 1) | (2, 2) |

H.T. Kung, "Why systolic architectures?," in IEEE Computer, vol. 15, no. 1, pp. 37-46, Jan. 1982, doi: 10.1109/MC.1982.1653825.

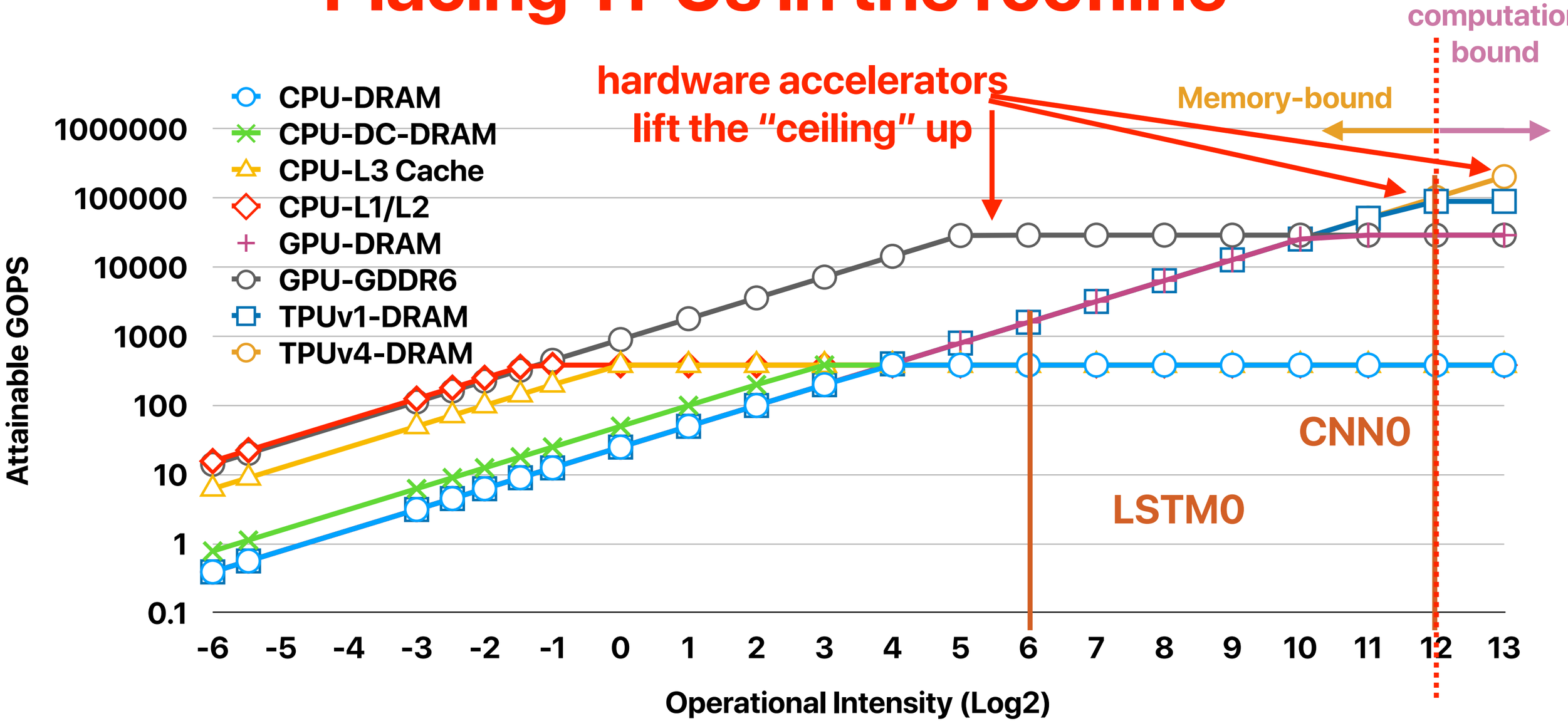
# Why can accelerator help?

- Performance-wise $ET = IC \times CPI \times CT$
  - The accelerator itself reduces the amount of “instructions” used to implement a program
  - The accelerator’s logic is simpler than that of a general-purpose pipeline (potentially higher clock rate)
- Power-wise $P = \alpha C V^2 f$
  - Simpler logic reduces the waste of gates
- Energy-wise $Energy = P \times ET$
  - Energy is power times execution time
  - If you improve both, it’s better anyway

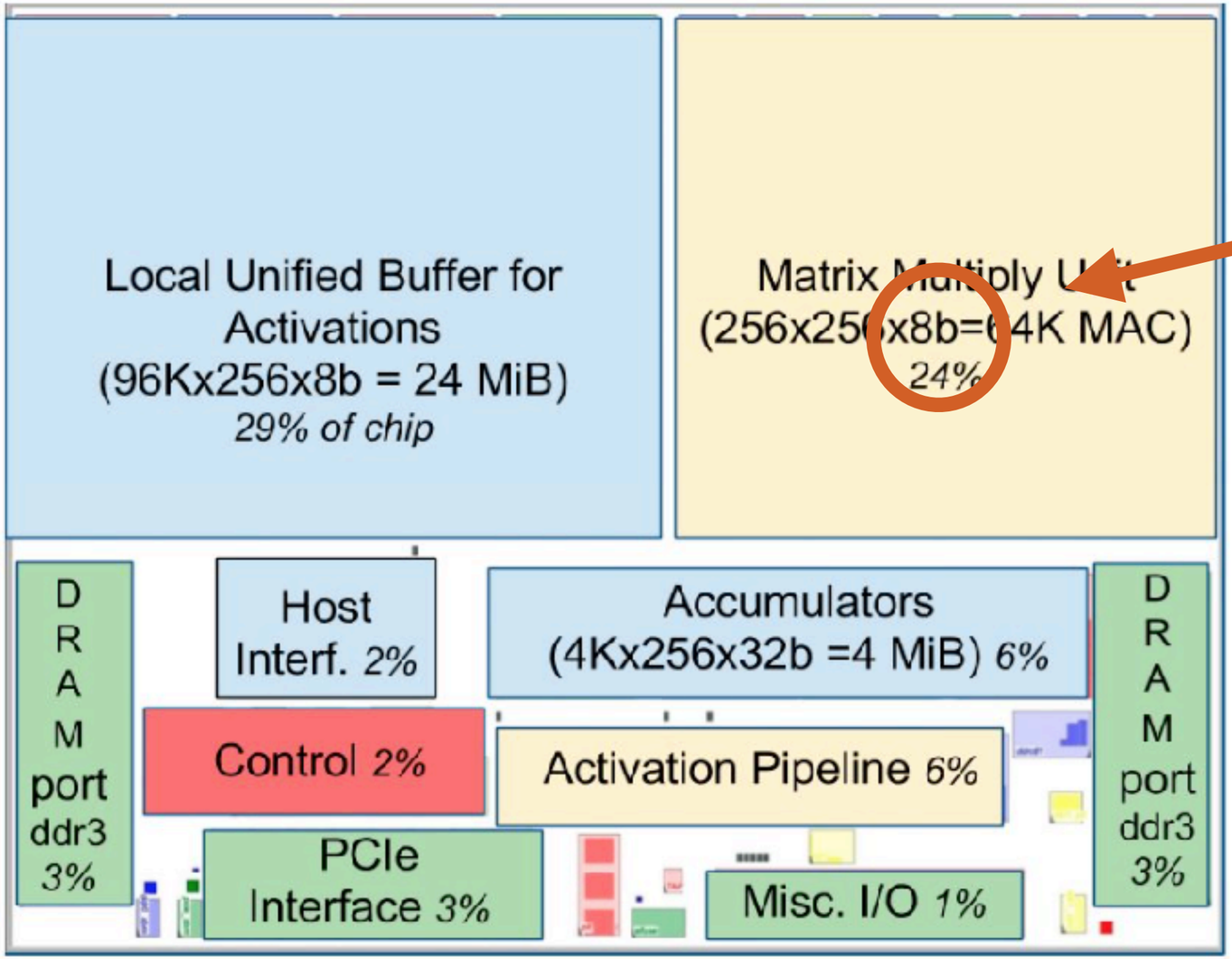
# TPU Block diagram



# Placing TPUs in the roofline



# TPU Floorplan




Why 8-bit?

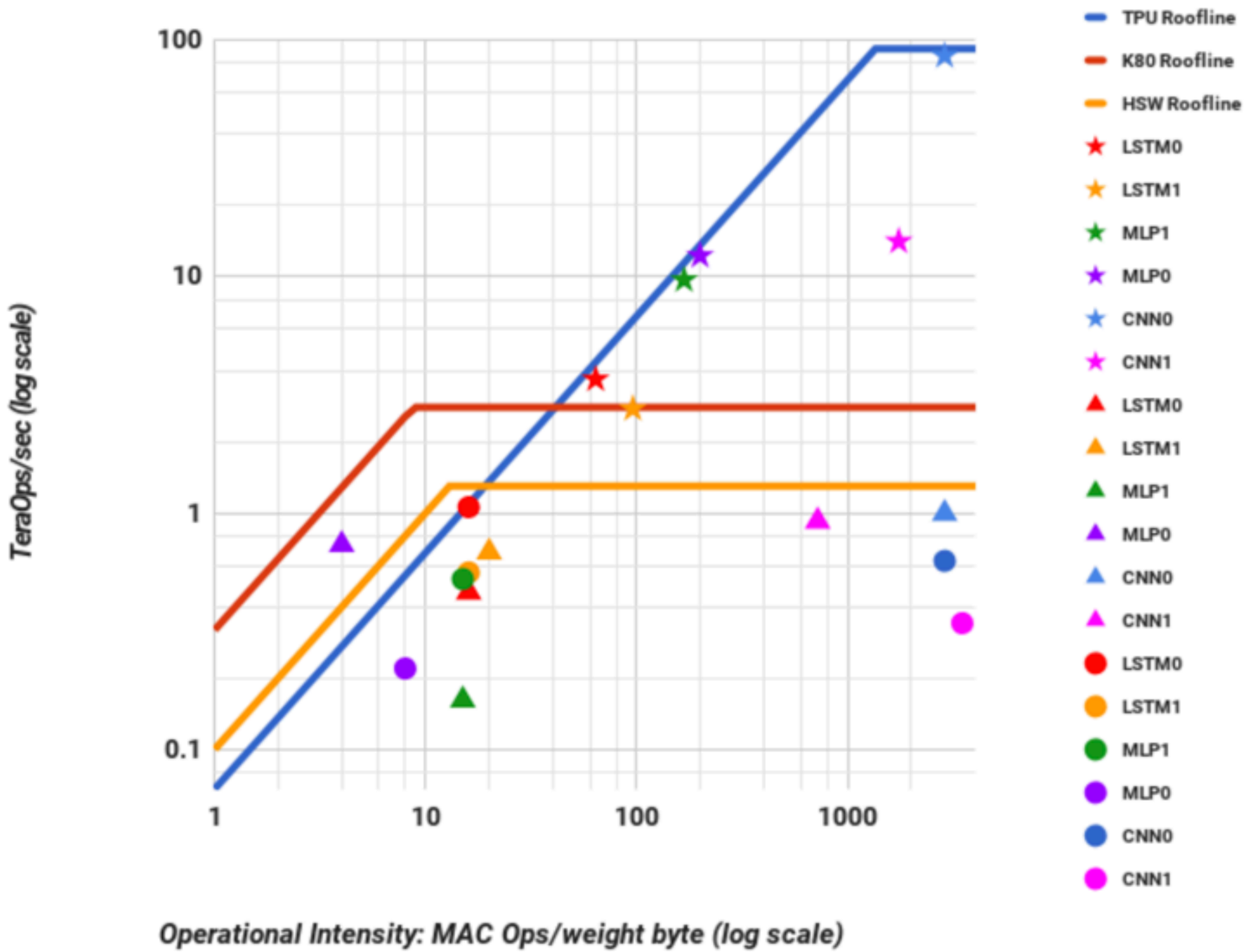
# The benefit of application-specific design

- We only need to provide "just enough" hardware support rather than complete support
  - We can deliver higher throughput within the same area
    - Support 2x bits in precision 4x the area
  - Reduce the total power/energy consumption

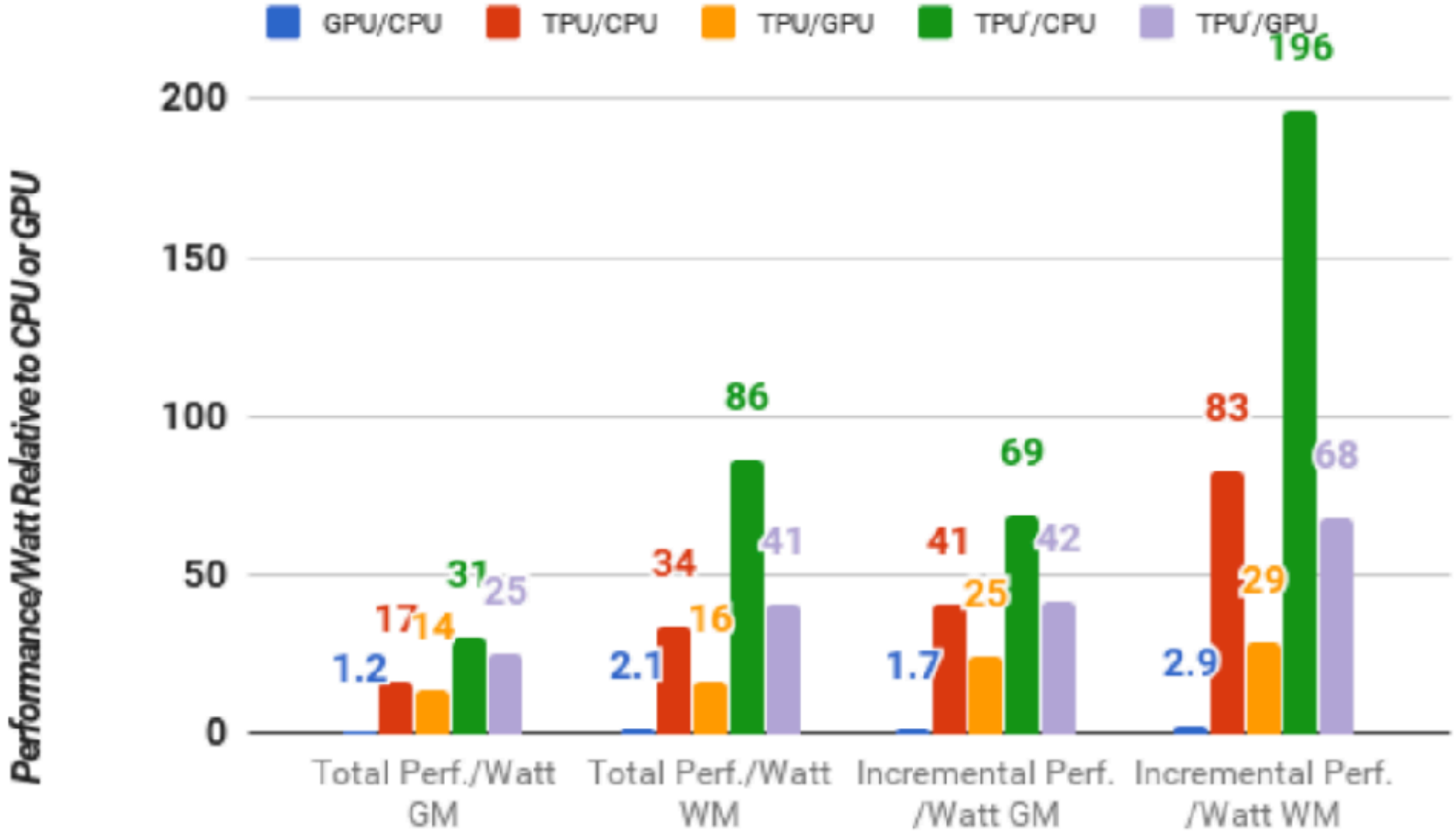
Table 2. Benchmarked servers use Haswell CPUs, K80 GPUs, and TPUs. The GPU and TPU use the Haswell server as host. The TPU die is less than half the Haswell die size.

| Model | $mm^2$ | nm | MHz | Thermal Design Power (TDP) per Chip | Cores | TeraOps/s | | Memory Gbyte/s | Chips/ Server | TDP per Server |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | 8 bit | FP32 | | | |
| Haswell | 662 | 22 | 2,300 | 145 W | 18 | 2.6 | 1.3 | 51 | 2 | 504 W |
| Nvidia K80 | 561 | 28 | 560 | 150 W | 13 | -- | 2.8 | 160 | 8 | 1,838 W |
| TPU | < 331 | 28 | 700 | 75 W | 1 | 92 | -- | 34 | 4 | 861 W |

# Most workloads surpass the roofline of GPUs



# Energy efficiency is better! — reduced cost!

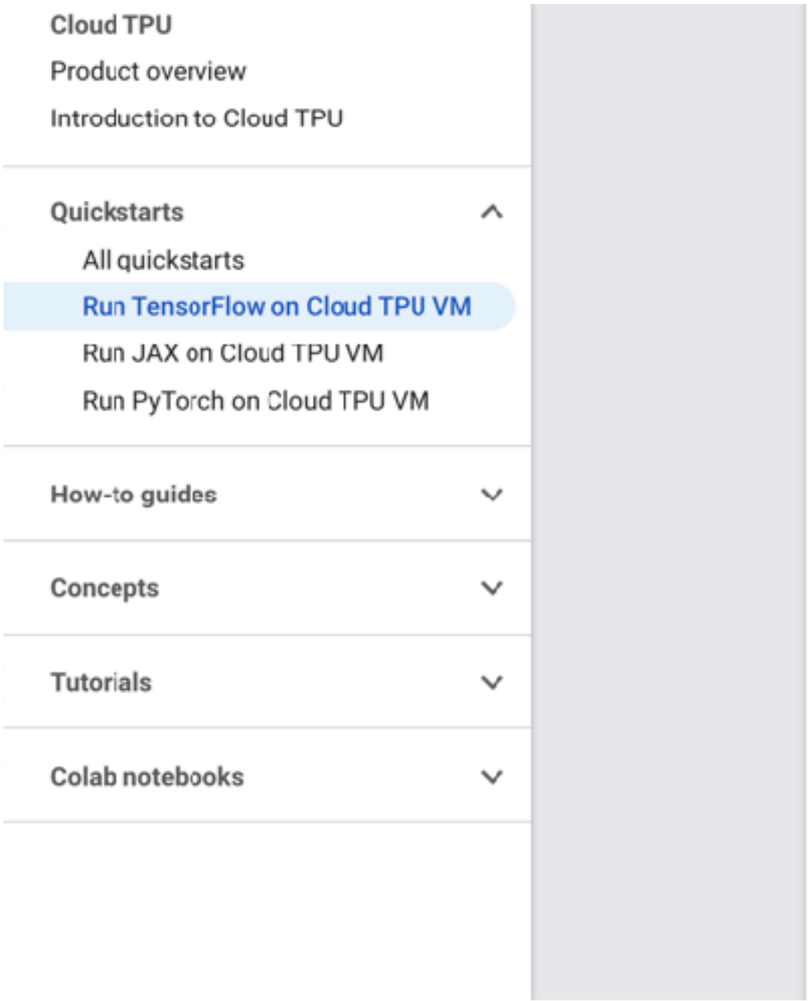


# From the TPU paper

## Pitfall: Being Ignorant of Architecture History when Designing a DSA.

Ideas that didn't fly for general-purpose computing might be ideal for DSAs. For the TPU, three important architectural ideas date from the early 1980s: systolic arrays,[5] decoupled-access/execute,[4] and CISC instructions. The first reduced the area and power of the large matrix multiply unit, the second fetches weights concurrently during operation of the matrix multiply unit, and the third better utilizes the limited bandwidth of the PCIe bus for delivering instructions. History-aware architects could have a competitive edge in this DSA era.

# TPU Programming model — domain specific languages




**Note:** For single TPUs, Slices, and Pods, pass your TPU name to `TPUClusterResolver()`.

```
import tensorflow as tf
print("Tensorflow version " + tf.__version__)

tpu = tf.distribute.cluster_resolver.TPUClusterResolver(tpu='your-tpu-name')  # TPU detection
print('Running on TPU ', tpu.cluster_spec().as_dict()['worker'])

tf.config.experimental_connect_to_cluster(tpu)
tf.tpu.experimental.initialize_tpu_system(tpu)
strategy = tf.distribute.experimental.TPUStrategy(tpu)

@tf.function
def add_fn(x,y):
    z = x + y
    return z

x = tf.constant(1.)
y = tf.constant(1.)
z = strategy.run(add_fn, args=(x,y))
print(z)
```

3:00

# Do you think TPU is a good enough design for AI/ML applications?

# Is TPU a good enough design?

# 10 lessons learned from 3 generations of Google TPUs

- Logic, wires, SRAM & DRAM improve unequally
- Leverage prior compiler optimizations
- Design for performance per TCO vs CapEx
- Backwards ML compatibility
- Inference DSAs need air cooling for global scale
- Some inference apps need floating point arithmetic
- Production inference normally needs multi-tenancy
- DNNs grow ~1.5x/year in memory and compute
- DNN workloads evolve with DNN breakthroughs
- Inference SLO limit is P99 latency, not batch size

N. P. Jouppi *et al.*, "Ten Lessons From Three Generations Shaped Google's TPUv4i : Industrial Product," *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, Valencia, Spain, 2021, pp. 1-14, doi: 10.1109/ISCA52012.2021.00010.

# Recent advancement of TPUs

# TPU v4

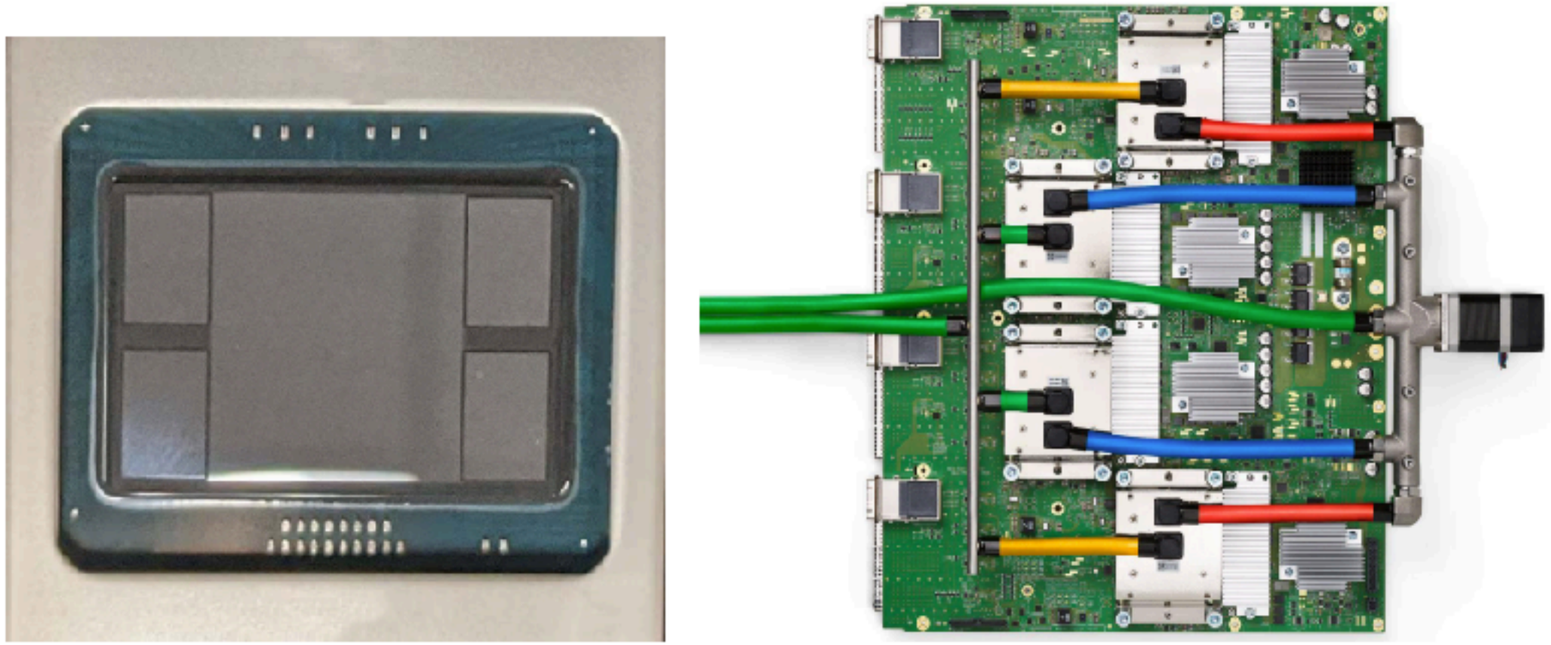


**Figure 2: The TPU v4 package (ASIC in center plus 4 HBM stacks) and printed circuit board with 4 liquid-cooled packages.** The board's front panel has 4 top-side PCIe connectors and 16 bottom-side OSFP connectors for inter-tray ICI links.

# Four generations of TPUs

**Table 1: Workloads by DNN model type (% TPUs used).** Over 90% of training at Google is on TPUs. The parenthesized entries split Transformer models into the subtypes of BERT and LLM. Columns 2 to 4 show workloads for inference [25], training and inference [26], and inference [27]. The last workload is for training on TPU v4s over 30 days in October 2022.

| *DNN Model* | *TPU v1 7/2016 (Inference)* | *TPU v3 4/2019 (Training & Inference)* | *TPU v4 Lite 2/2020 (Inference)* | *TPU v4 10/2022 (Training)* |
|---|---|---|---|---|
| MLP/DLRM | 61% | 27% | 25% | 24% |
| RNN | 29% | 21% | 29% | 2% |
| CNN | 5% | 24% | 18% | 12% |
| Transformer | -- | 21% | 28% | 57% |
| *(BERT)* | -- | -- | *(28%)* | *(26%)* |
| *(LLM)* | -- | -- | -- | *(31%)* |

Norm Jouppi, George Kurian, Sheng Li, Peter Ma, Rahul Nagarajan, Lifeng Nai, Nishant Patil, Suvinay Subramanian, Andy Swing, Brian Towles, Clifford Young, Xiang Zhou, Zongwei Zhou, and David A Patterson. 2023. TPU v4: An Optically Reconfigurable Supercomputer for Machine Learning with Hardware Support for Embeddings. In Proceedings of the 50th Annual International Symposium on Computer Architecture (ISCA '23).

# RTX on—The NVIDIA Turing GPU

3:00

# Why RTX architecture?

# Turing architecture — specialized for key workloads

- CUDA cores for vector processing
- Tensor cores for AI
- RT cores for VR/AR

> While Turing is packed with features and horsepower,[1] we made fundamental advancements in several key areas—streaming multiprocessor (SM) efficiency, a Tensor Core for AI inferencing, and an RTCore for ray-tracing acceleration.

# Tensor Cores

# Tensor cores for deep learning

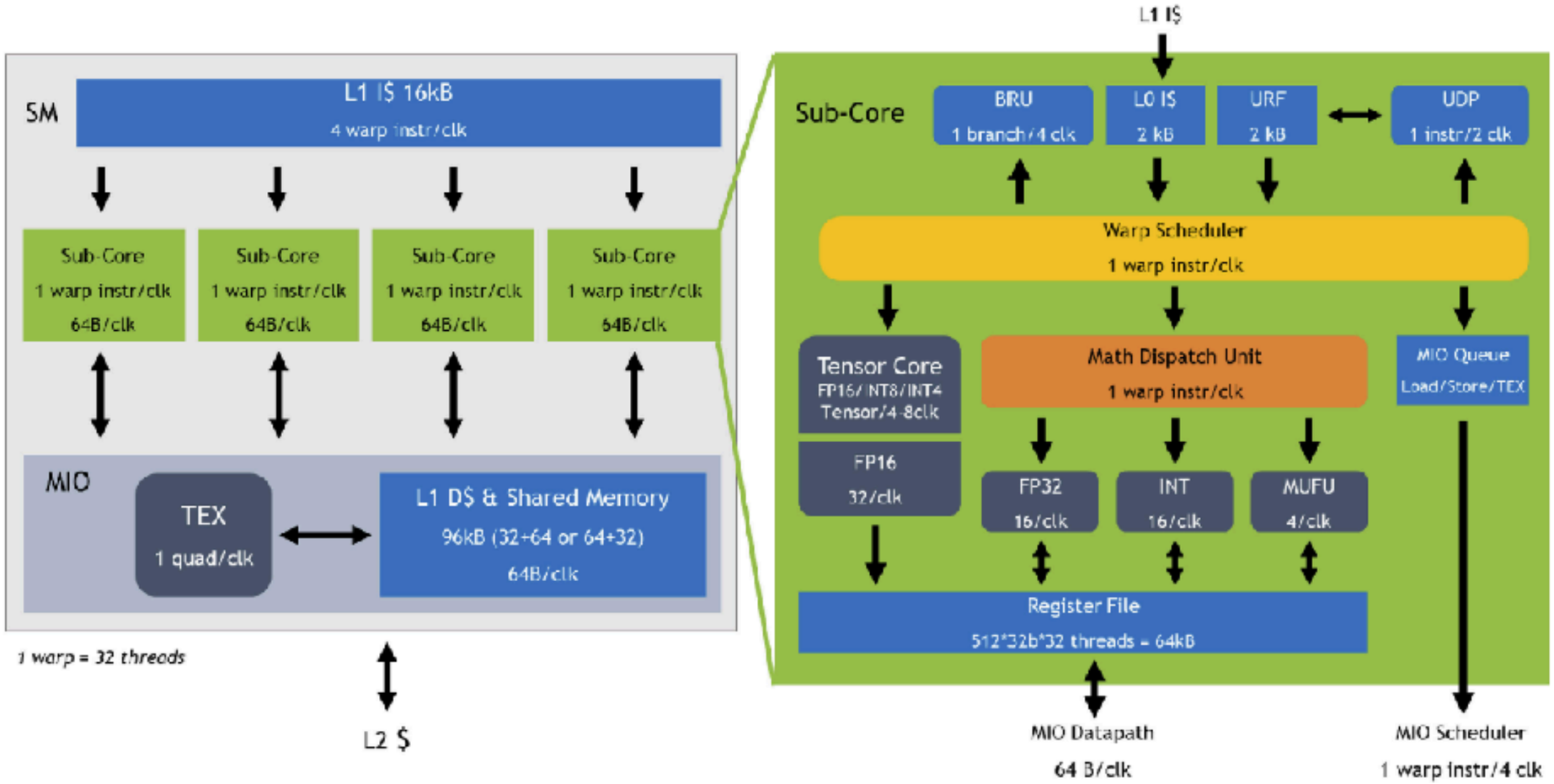



**Figure 1.** Turing GPU SM, comprising four subcores and a memory interface (MIO). Math throughput, memory bandwidth, L1 data cache topology, register file and cache capacity, and a new uniform datapath were all designed or modified to increase processor efficiency over the previous generation.

# Turing Architecture

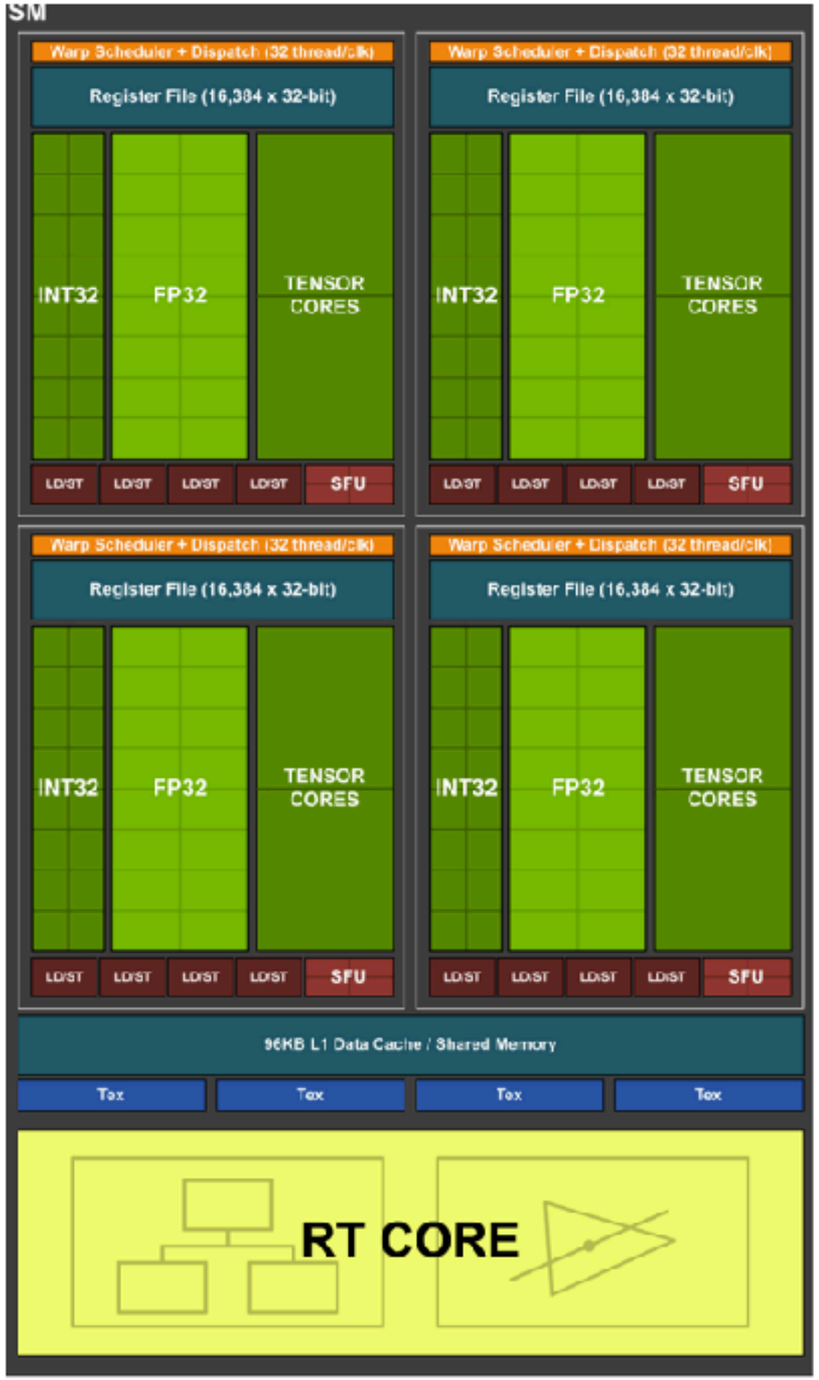


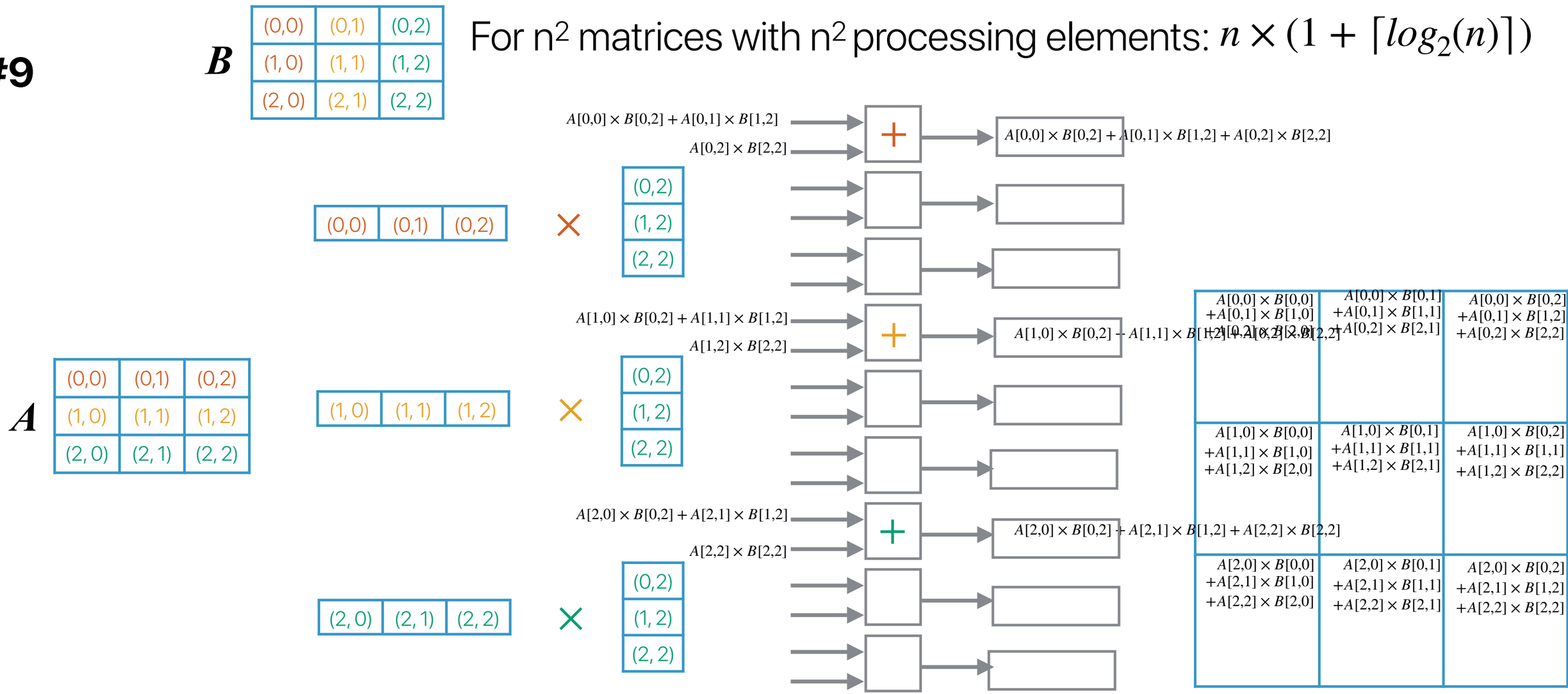

Figure 4. Turing TU102/TU104/TU106 Streaming Multiprocessor (SM)

NVIDIA, "NVIDIA Turing GPU architecture: Graphics reinvented," 2018. [Online]. Available: https://www. nvidia.com/content/dam/en-zz/Solutions/designvisualization/ technologies/ turing-architecture/NVIDIATuring- Architecture-Whitepaper.pdf

# Vector processing for MM

#9

For n² matrices with n² processing elements: $n \times (1 + \lceil log_2(n) \rceil)$

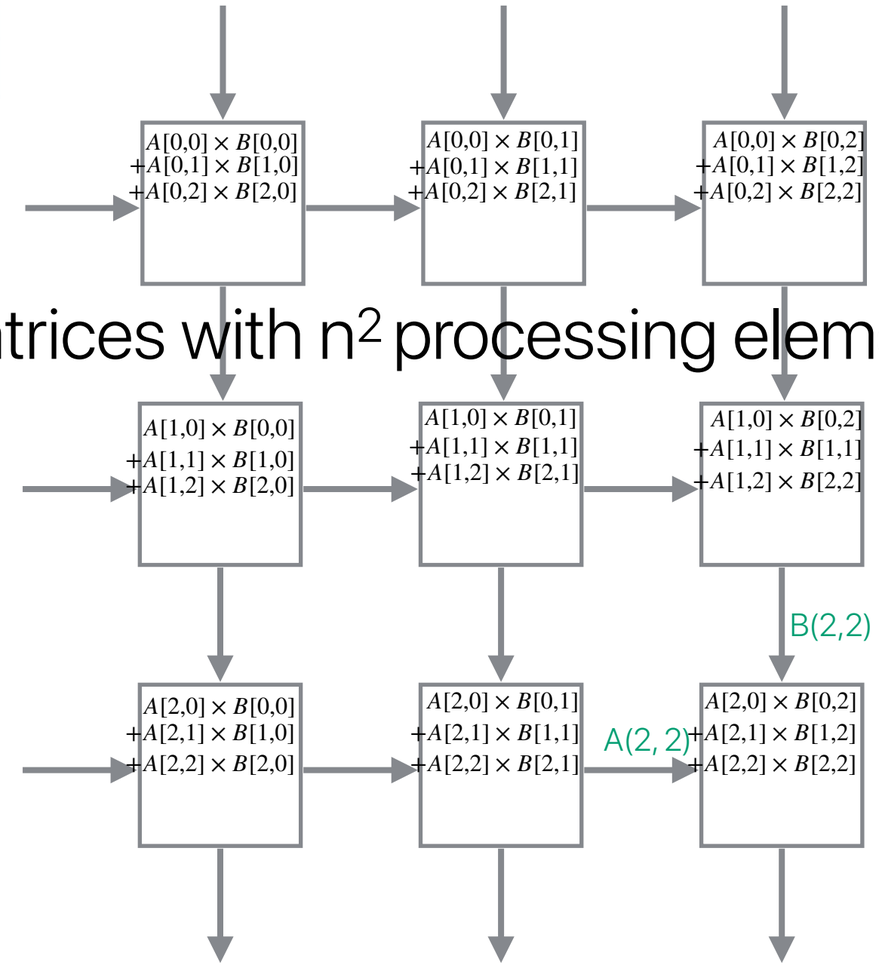


# Systolic Arrays used by AI/ML Accelerators

#7

**B**

| | | |
|---|---|---|
| (0,0) | (0,1) | (0,2) |
| (1, 0) | (1, 1) | (1, 2) |
| (2, 0) | (2, 1) | (2, 2) |

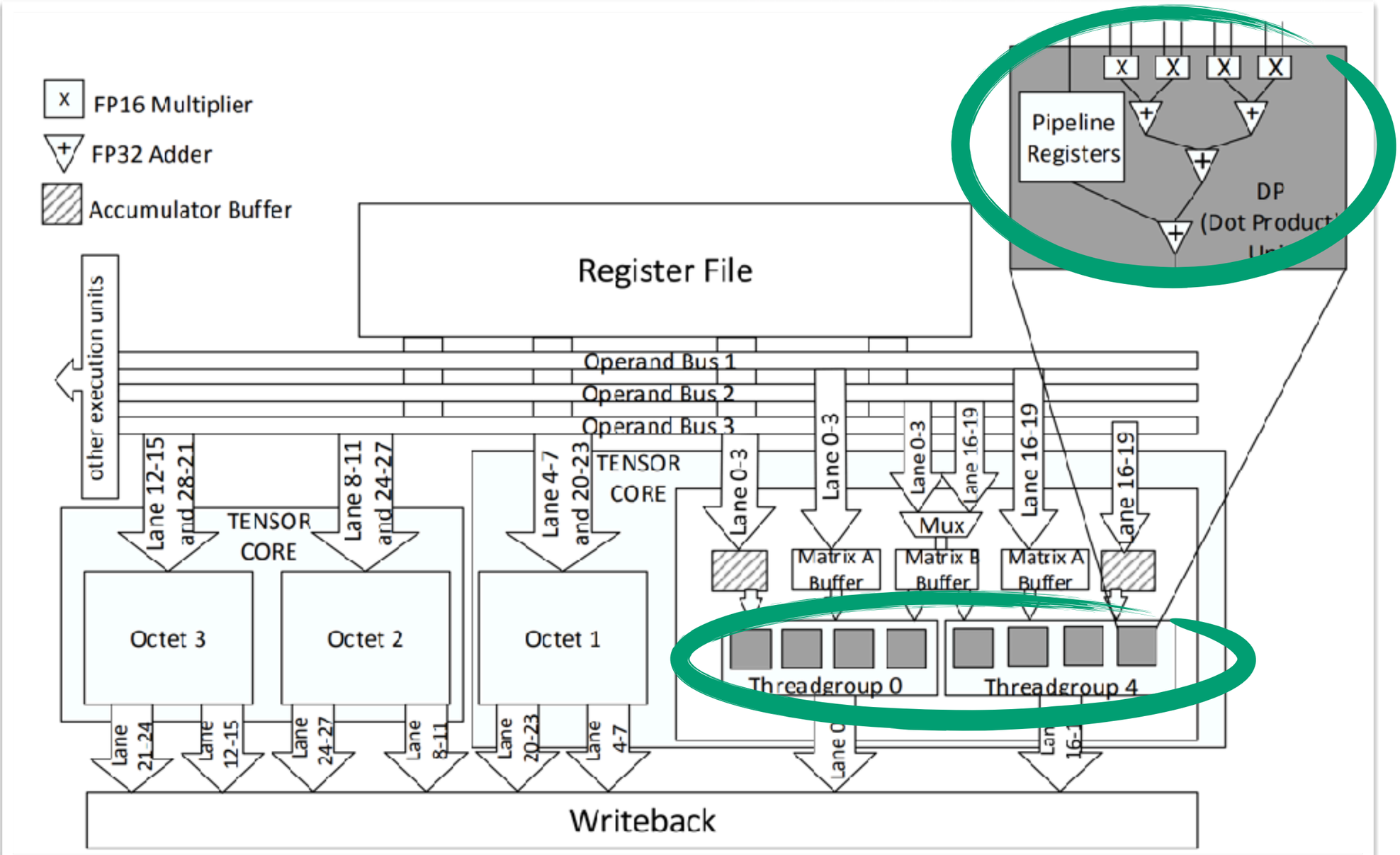



For $n^2$ matrices with $n^2$ processing elements: $3 \times n - 2$

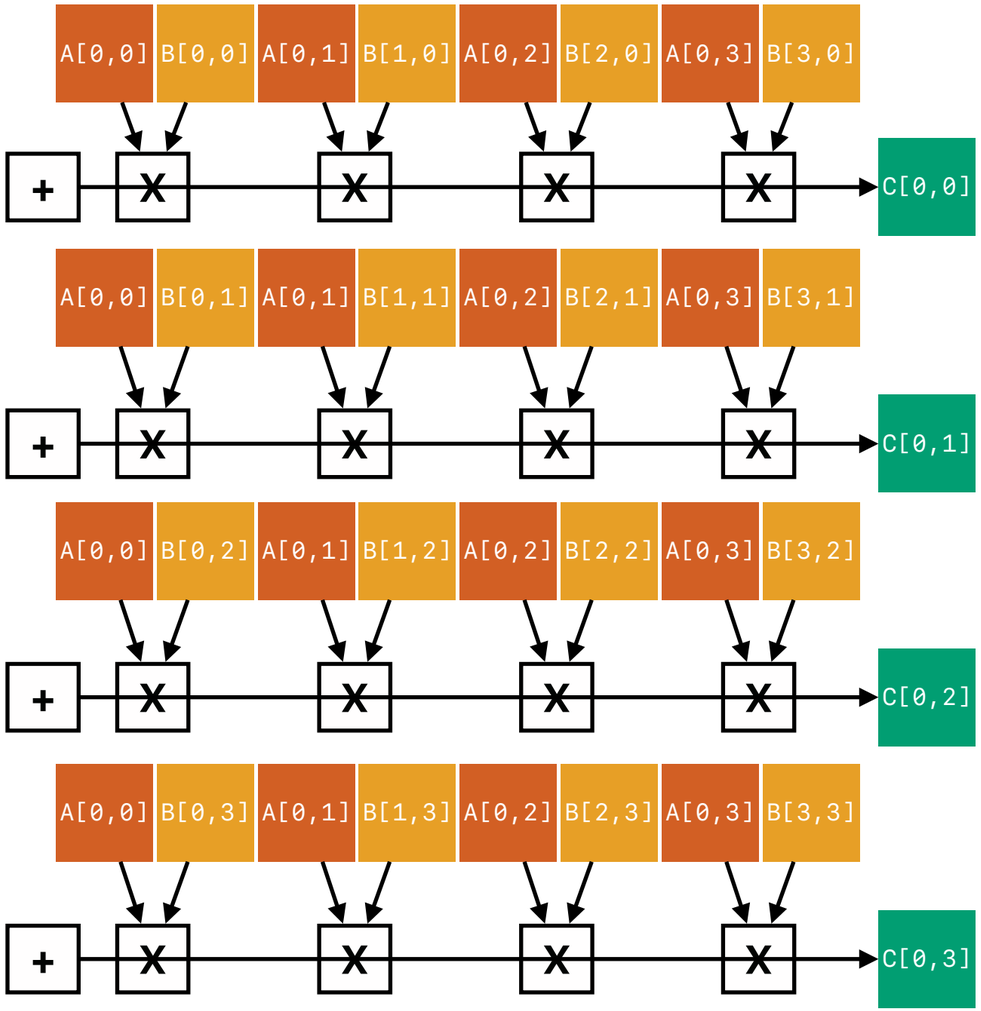
**A**

| | | |
|---|---|---|
| (0,0) | (0,1) | (0,2) |
| (1, 0) | (1, 1) | (1, 2) |
| (2, 0) | (2, 1) | (2, 2) |

H.T. Kung, "Why systolic architectures?," in IEEE Computer, vol. 15, no. 1, pp. 37-46, Jan. 1982, doi: 10.1109/MC.1982.1653825.


X FP16 Multiplier
+ FP32 Adder
Accumulator Buffer
Register File
other execution units
Operand Bus 1
Operand Bus 2
Operand Bus 3
Lane 12-15 and 28-21
Lane 8-11 and 24-27
Lane 4-7 and 20-23
TENSOR CORE
TENSOR CORE
Lane 0-3
Lane 0-3
Lane 16-19
Lane 16-19
Lane 16-19
Mux
Matrix A Buffer
Matrix B Buffer
Matrix A Buffer
Octet 3
Octet 2
Octet 1
Threadgroup 0
Threadgroup 4
Lane 21-24
Lane 12-15
Lane 24-27
Lane 8-11
Lane 20-23
Lane 4-7
Writeback
Pipeline Registers
DP
(Dot Product

# Tensor Cores



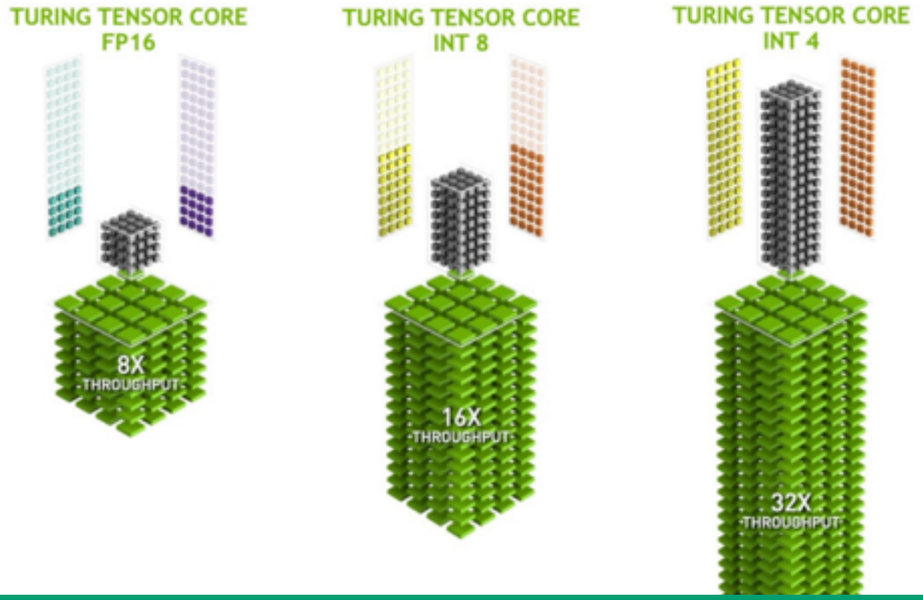

| | | | |
|---|---|---|---|
| A[0,0] | A[0,1] | A[0,2] | A[0,3] |
| A[1,0] | A[1,1] | A[1,2] | A[1,3] |
| A[2,0] | A[2,1] | A[2,2] | A[2,3] |
| A[3,0] | A[3,1] | A[3,2] | A[3,3] |

| | | | |
|---|---|---|---|
| B[0,0] | B[0,1] | B[0,2] | B[0,3] |
| B[1,0] | B[1,1] | B[1,2] | B[1,3] |
| B[2,0] | B[2,1] | B[2,2] | B[2,3] |
| B[3,0] | B[3,1] | B[3,2] | B[3,3] |

| | | | |
|---|---|---|---|
| C[0,0] | C[0,1] | C[0,2] | C[0,3] |
| C[1,0] | C[1,1] | C[1,2] | C[1,3] |
| C[2,0] | C[2,1] | C[2,2] | C[2,3] |
| C[3,0] | C[3,1] | C[3,2] | C[3,3] |

# Efficiency of Tensor Cores

The NVIDIA Tesla T4 GPU includes 2,560 CUDA Cores and 320 Tensor Cores, delivering up to 130 TOPs (Tera Operations per second) of INT8 and up to 260 TOPS of INT4 inferencing performance (see Appendix A, *Turing TU104 GPU* for more Tesla T4 specifications). Compared to CPU-based inferencing, the Tesla T4, powered by the new Turing Tensor Cores, delivers up to 40X higher inference performance[6] (see Figure 9).




**FLOPS of 8K by 8K matrix multiplications =**

$$8192 \times 8192 \times 8192 \times 2$$

$$\frac{8192 \times 8192 \times 8192 \times 2}{2560} = 429496730 \ CUDA \ core \ cycles$$

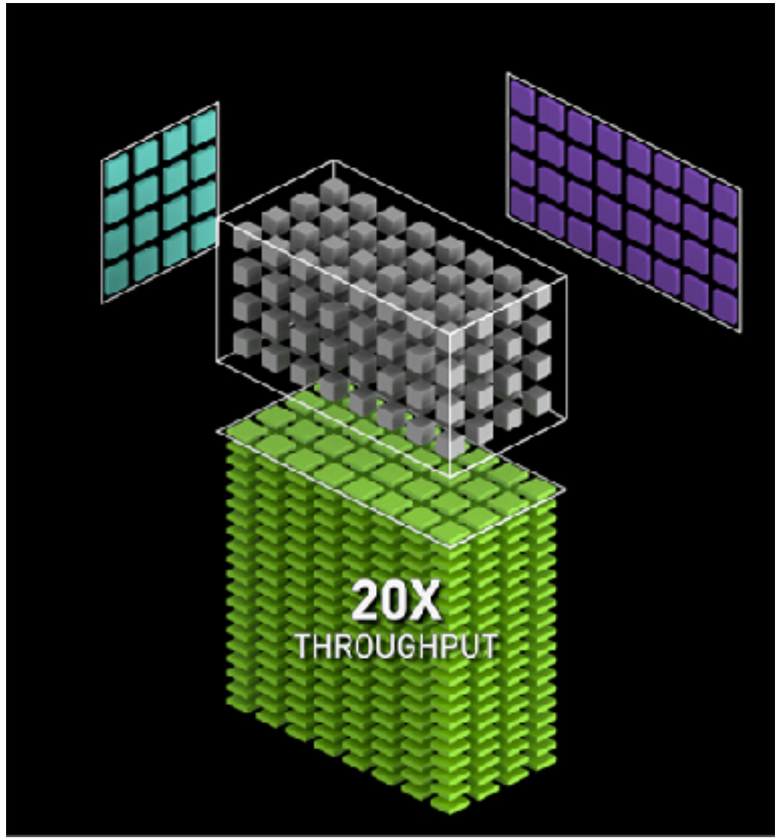
$$\frac{8192 \times 8192 \times 8192 \times 2}{320 \times 128} = 26843546 \ Tensor \ core \ cycles$$

**Tensor cores make matrix processing 16x faster**

**Each tensor core operation performs 4x4x4 MMA in one cycle**
**~ 128 FP operations in conventional scalar models**

NVIDIA, "NVIDIA Turing GPU architecture: Graphics reinvented," 2018. [Online]. Available: https://www. nvidia.com/content/dam/en-zz/Solutions/designvisualization/technologies/turing-architecture/NVIDIATuring- Architecture-Whitepaper.pdf

# Efficiency of Tensor Cores (RTX 4090)




| | RTX 4090 |
|---|---|
| CUDA Cores | 16384 |
| Tensor Cores | 512 |

**FLOPS of 8K by 8K matrix multiplications =**

$$8192 \times 8192 \times 8192 \times 2$$

$$\frac{8192 \times 8192 \times 8192 \times 2}{16384} = 67{,}108{,}864 \ \textit{CUDA core cycles}$$

$$\frac{8192 \times 8192 \times 8192 \times 2}{512 \times 256} = 8{,}388{,}608 \ \textit{Tensor core cycles}$$

**Each tensor core operation performs 8x4x4 MMA in one cycle**
**~ 256 FP operations in conventional scalar models**

**Tensor cores make matrix processing 8x faster**

# Roogle Project Meetings

- Make an appointment through the Google Calendar
- We're trying to make a company project theme
- Available resources
  - SmartSSD, Edge TPUs, CUDA GPUs, Tensor Cores, or something else
- Project ideas
  - Accelerating applications through AI/ML accelerators
  - Accelerating applications through intelligent storage devices
  - Accelerating applications through innovative parallel programming models that hardware accelerators enable
  - Anything related to what we discussed in this class!

# Don’t forget…

- Form your group and discuss the project ideas
- Check the schedule/Google Spaces for up-to-date reading list and submit the summary!

Electrical
Computer Science
Engineering

277

つづく