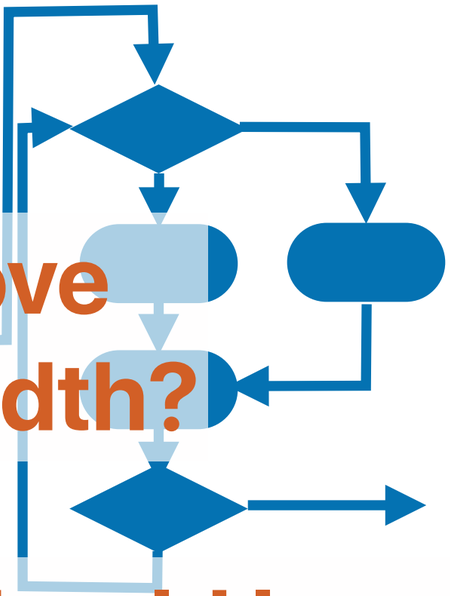
The roofline model and its implications

Hung-Wei Tseng

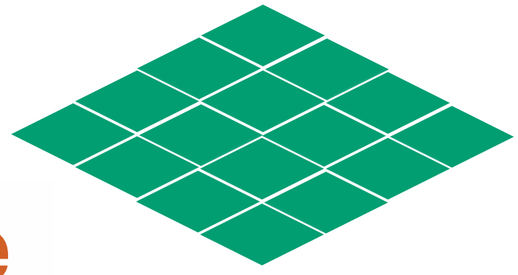
Recap: From the software perspective

Should I improve memory bandwidth?

Should I use NDP?

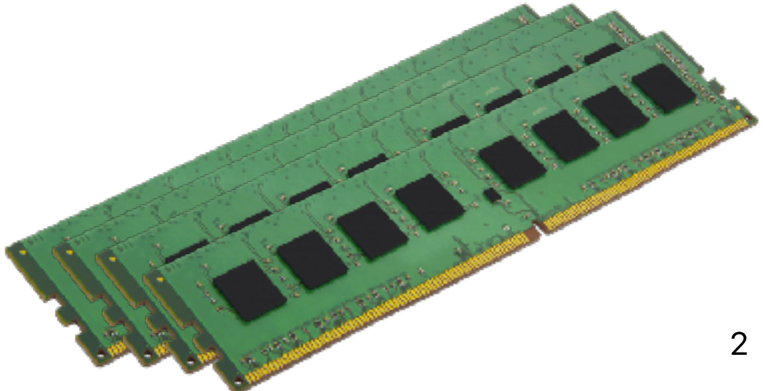


Should I use accelerators?



Source “data"

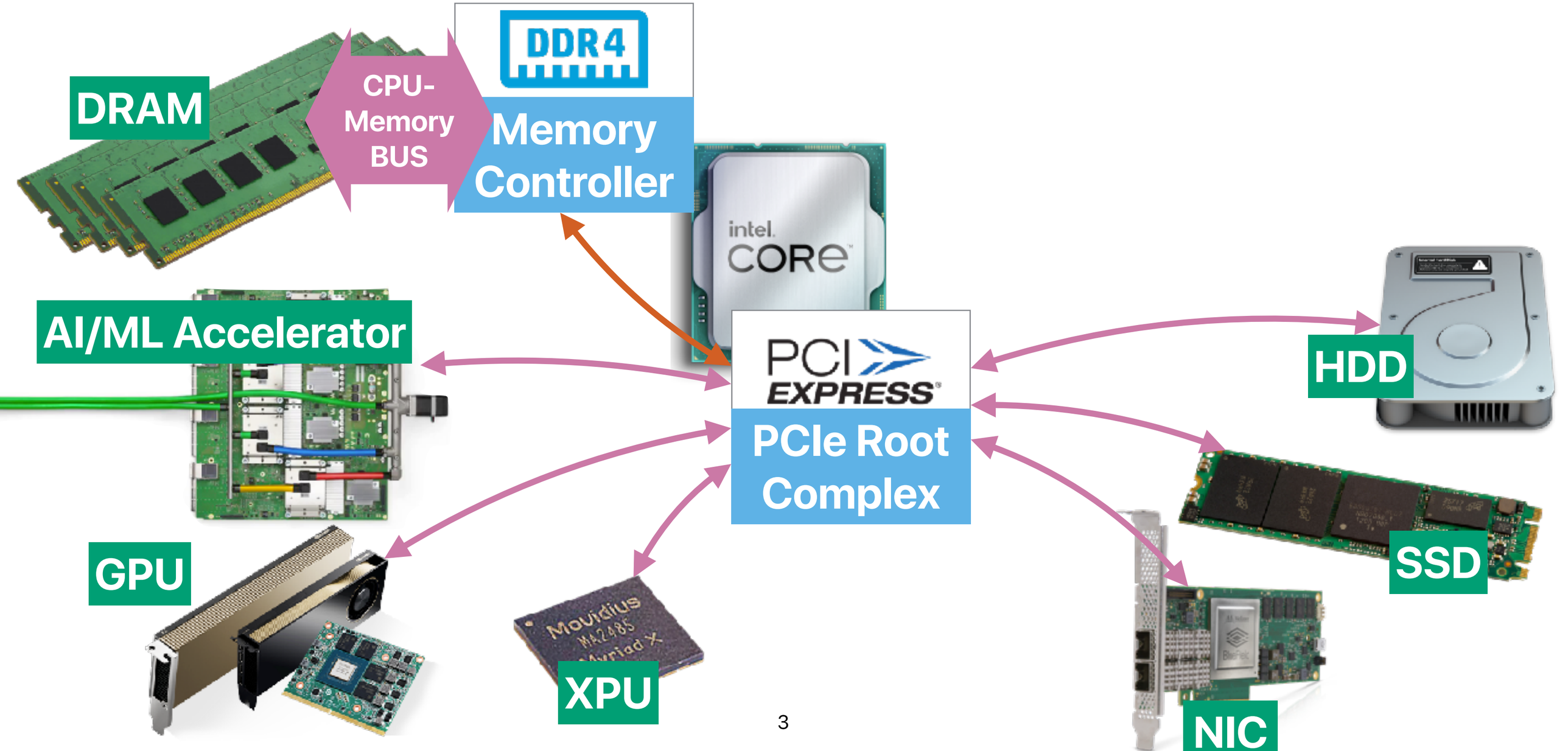
“Data” structures



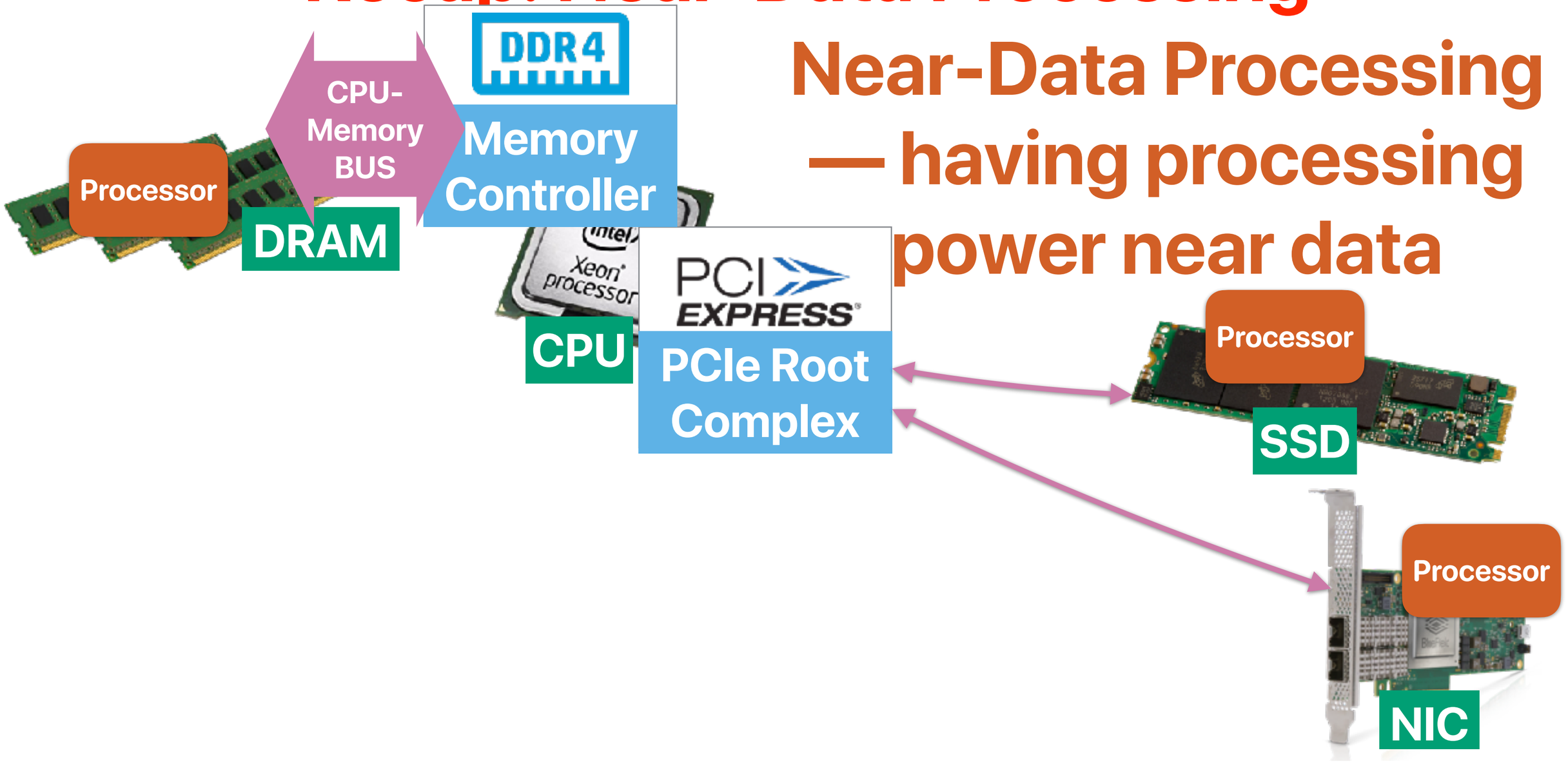
Algorithms

Result

Recap: The “real” datapath



Recap: Near-Data Processing



Near-Data Processing — having processing power near data

Compute-centric? Data-centric?

Outline

- The roofline model
- Why heterogeneous computer architectures
  - GPUs
  - TPUs

3:00

How much can a boba tea shop earn each day?

Ideas?

- How many customers we can attract each day
- How many cups we can make each day
- How much each cup costs

Ideas?

- How many bytes of data we can supply each cycle
- How many OPs we can perform each cycle
- How many OPs each byte of data need

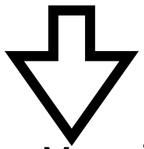
Operational Intensity

- Number of operations each byte of data would need for an algorithm
- Changes with algorithm

Example — 32-bit floating point matrix multiplications

- Performing $M \times N \times P$ matrix multiplications
  - The size of a is $M \times N$
  - The size of b is $N \times P$
  - The size of c is $M \times P$
- Total amount of operations = $2 \times M \times N \times P$ ops
- Total size of data accesses = $4 \times (2 \times N + 1) \times M \times P$ bytes
- Average operations per byte = $$\frac{2 \times M \times N \times P}{4 \times (2 \times N + 1) \times M \times P} = \frac{1}{4} = 0.25$$

```
for(i = 0; i < M; i++) {
  for(j = 0; j < P; j++) {
    for(k = 0; k < N; k++) {
      c[i][j] += a[i][k]*b[k][j];
    }
  }
}
```

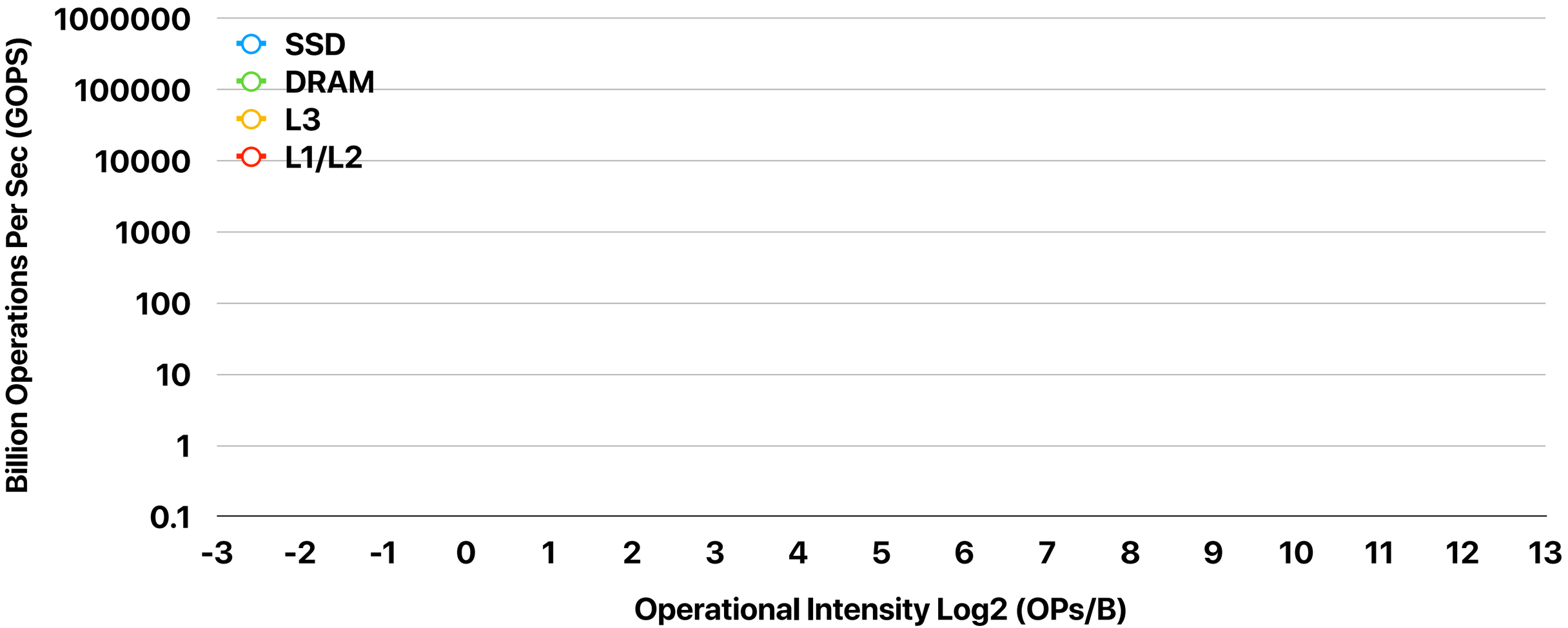


```
for(i = 0; i < M; i++) {
  for(j = 0; j < P; j++) {
    sum = 0.0;
    for(k = 0; k < N; k++) {
      sum += a[i][k]*b[k][j];
    }
    c[i][j] = sum;
  }
}
```

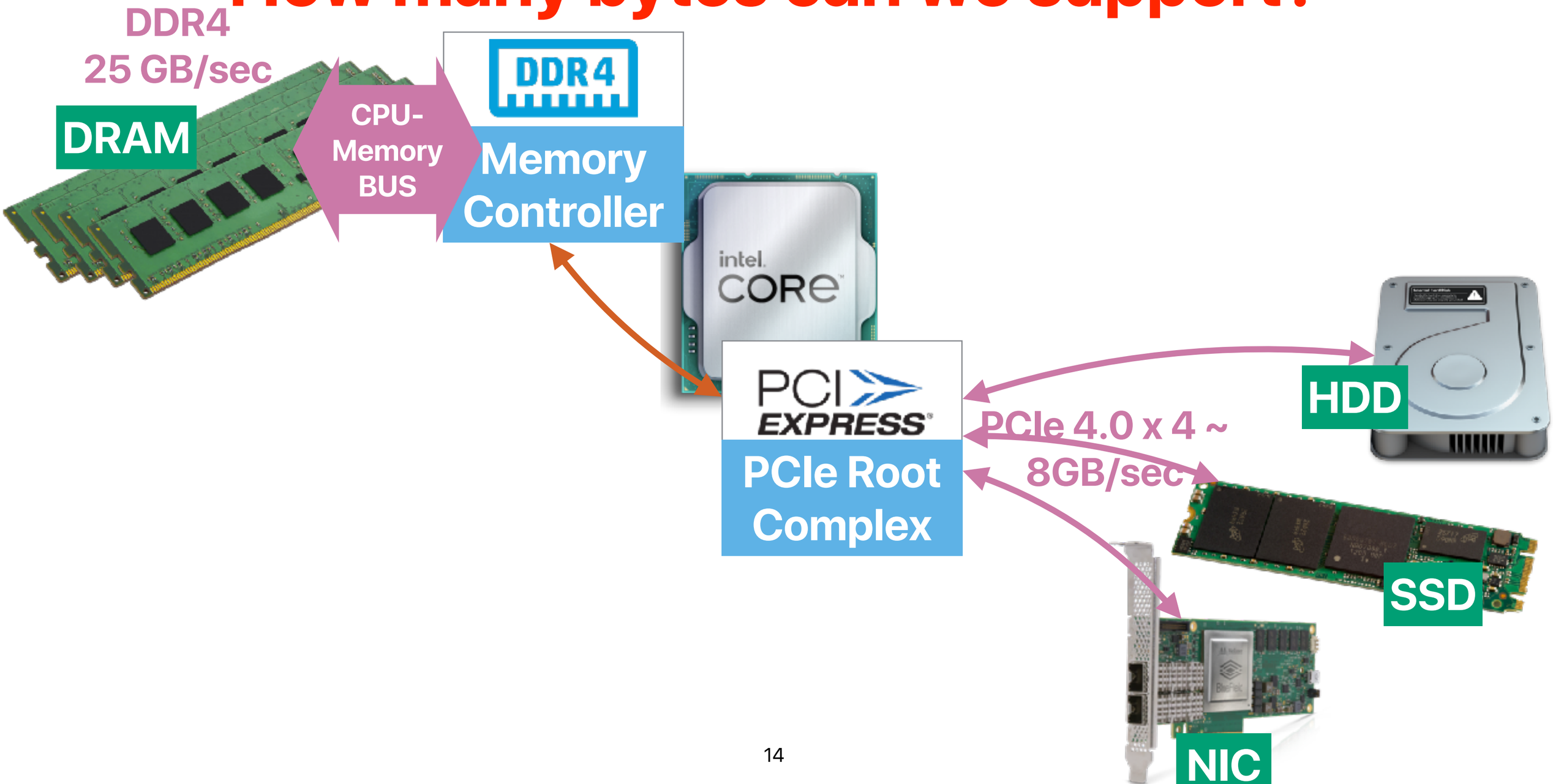
Table 2: Characteristics of four floating-point kernels.

Name	Operational Intensity	Description
SpMV[29]	0.17 to 0.25	Sparse Matrix-Vector multiply: y = A*x where A is a sparse matrix and x, y are dense vectors; multiplies and adds equal.
LBMHD[28]	0.70 to 1.07	Lattice-Boltzmann Magnetohydro-dynamics is a structured grid code with a series of time steps.
Stencil[12]	0.33 to 0.50	A multigrid kernel that updates seven nearby points in a 3D stencil for a 256^3 problem.
3D FFT	1.09 to 1.64	3D Fast Fourier Transform (2 sizes: 128^3 and 512^3).

How fast can we supply data?



How many bytes can we support?



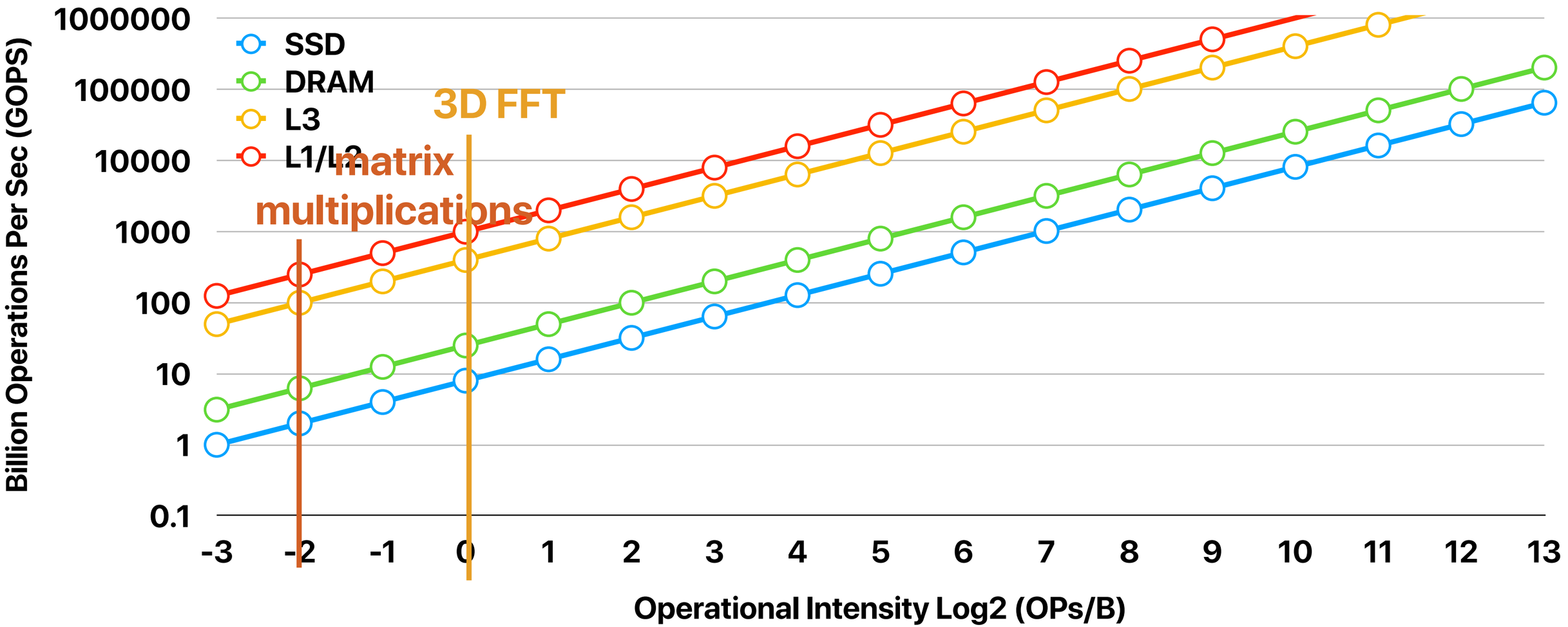
Size, Latency, and Bandwidth of Memory Subsystem Components

Assuming you have a large processor (about 16 cores), the following summarizes, for 2016, approximate data totals present in and moving through the system.

Memory	Size	Latency	Bandwidth
L1 cache	32 KB	1 nanosecond	1 TB/second
L2 cache	256 KB	4 nanoseconds	1 TB/second Sometimes shared by two cores
L3 cache	8 MB or more	10x slower than L2	>400 GB/second
MCDRAM		2x slower than L3	400 GB/second
Main memory on DDR DIMMs	4 GB-1 TB	Similar to MCDRAM	100 GB/second
Main memory on Cornelis* Omni-Path Fabric	Limited only by cost	Depends on distance	Depends on distance and hardware
I/O devices on memory bus	6 TB	100x-1000x slower than memory	25 GB/second
I/O devices on PCIe bus	Limited only by cost	From less than milliseconds to minutes	GB-TB/hour Depends on distance and hardware

https://www.intel.com/content/www/us/en/developer/articles/technical/memory-performance-in-a-nutshell.html

Sustainable OPS if we assume unlimited computing resource



Finding the limitation — the roofline model of CPU processing

How fast is a CPU?

https://download.intel.com/newsroom/2021/client-computing/intel-architecture-day-2021-presentation.pdf

How fast is a CPU?

- Clock rate: 3 GHz — 3B cycles/instructions per “ALU”
- 2 FMA units can perform 2 256-bit vector floating point operations per core (128 pairs of floating point numbers)
- 384B floating point operations per second — or say 384 GFLOPS

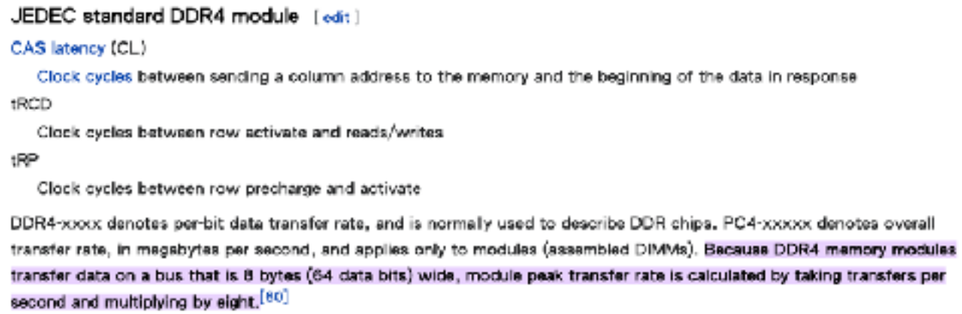
Determining factors of performance

- 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.

Example — matrix multiplications (cont.)

- Remember that we have a CPU core supporting 9G operations per second (OPS)
- If data are currently stored in SSD
  - We can supply data in 4GB/sec
  - The supplied data per second needs
    $$8GB \times \frac{1\ OPS}{4\ B} = 2\ GOPS < 384GFLOPS$$
- If data are currently stored in DRAM
  - We can supply data in 25GB/sec per module
  - The supplied data per second needs per module
    $$25GB \times \frac{1\ OPS}{4\ B} = 6.25\ GOPS < 384GFLOPS$$
  - If we have 2 modules
    $$50GB \times \frac{1\ OPS}{4\ B} = 14.5\ GOPS < 384GFLOPS$$



JEDEC standard DDR4 module [edit]

CAS latency (CL)

Clock cycles between sending a column address to the memory and the beginning of the data in response

tRCD

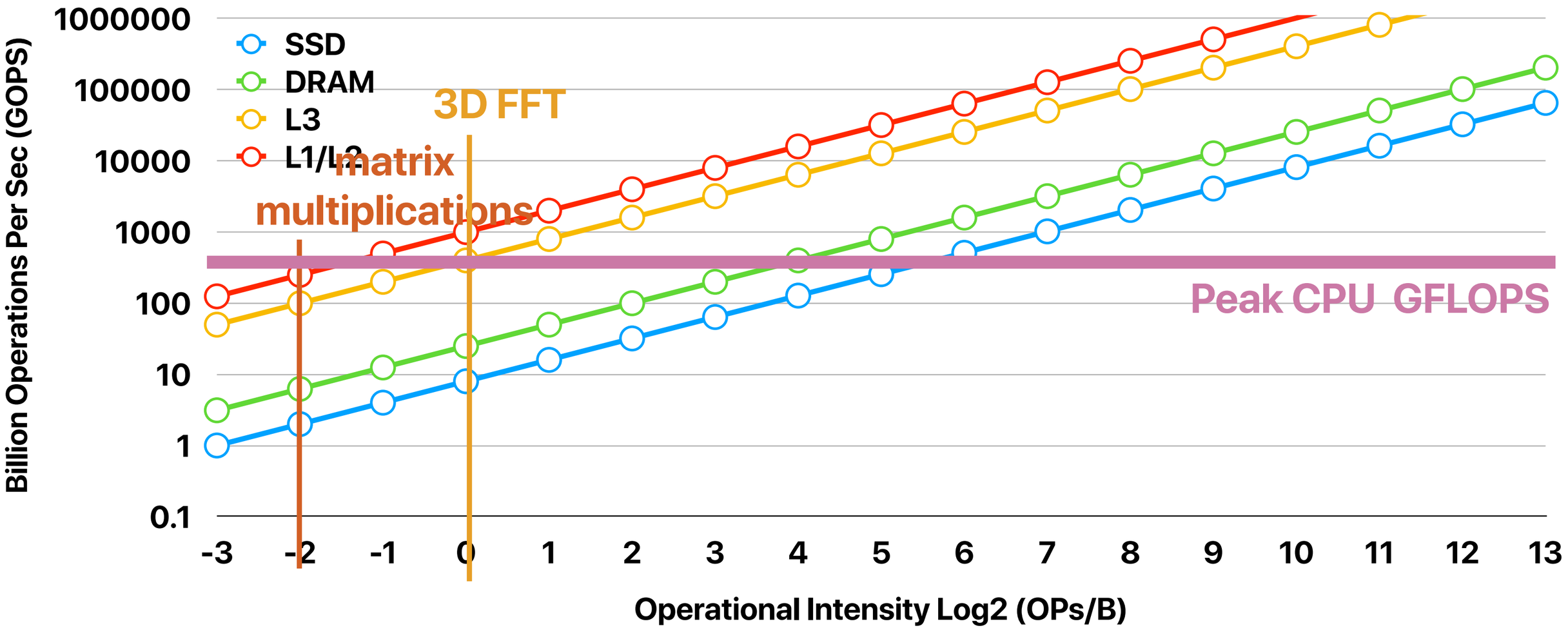
Clock cycles between row activate and reads/writes

tRP

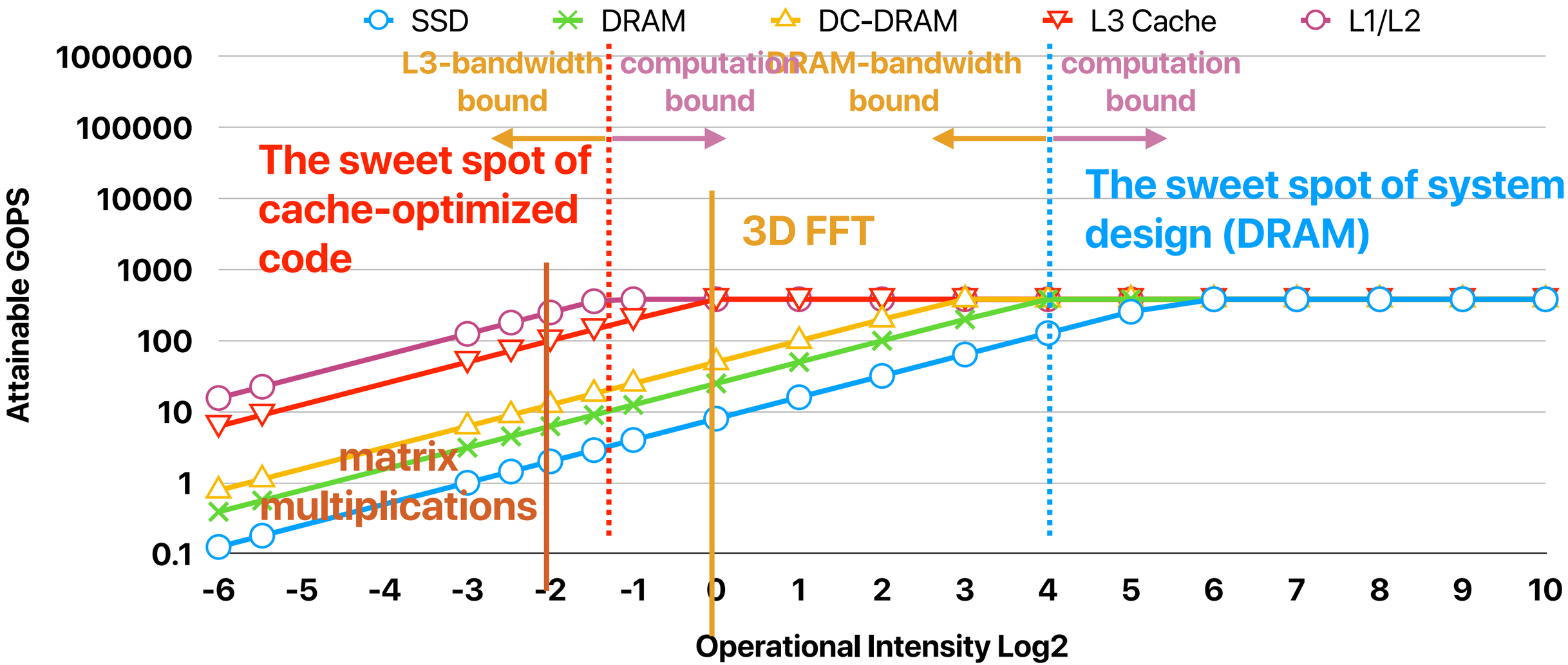
Clock cycles between row precharge and activate

DDR4-xxxx denotes per-bit data transfer rate, and is normally used to describe DDR chips. PC4-xxxxx denotes overall transfer rate, in megabytes per second, and applies only to modules (assembled DIMMs). Because DDR4 memory modules transfer data on a bus that is 8 bytes (64 data bits) wide, module peak transfer rate is calculated by taking transfers per second and multiplying by eight.[80]

Sustainable OPS if we assume unlimited computing resource



The roofline under various configurations



1.6 BFLOAT16 FLOATING-POINT FORMAT

Intel® Deep Learning Boost (Intel® DL Boost) uses bfloat16 format (BF16). Figure 1-6 illustrates BF16 versus FP16 and FP32.

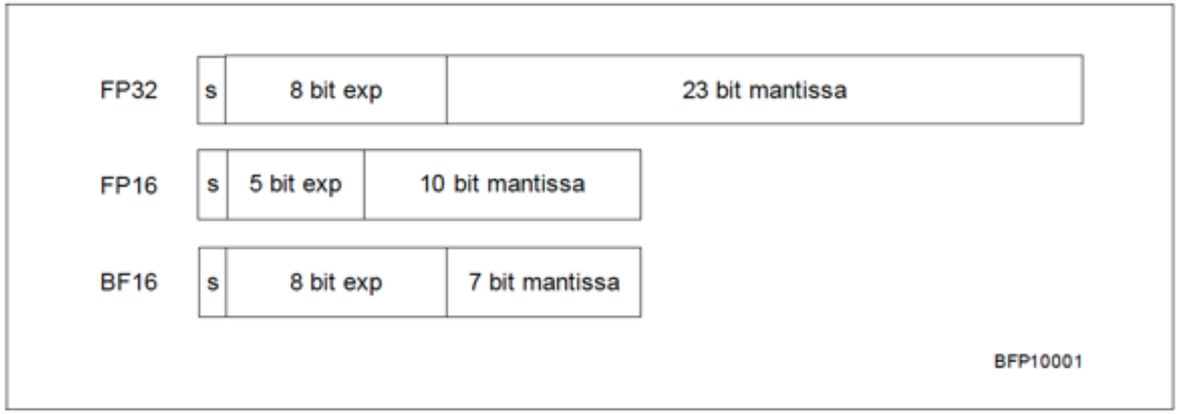


Figure 1-6. Comparison of BF16 to FP16 and FP32

Run in Google Colab | View source on GitHub | Download notebook

Overview

Mixed precision is the use of both 16-bit and 32-bit floating-point types in a model during trainin
and use less memory. By keeping certain parts of the model in the 32-bit types for numeric stab
a lower step time and train equally as well in terms of the evaluation metrics such as accuracy.
to use the Keras mixed precision API to speed up your models. Using this API can improve perfo
times on modern GPUs, 60% on TPUs and more than 2 times on latest Intel CPUs.

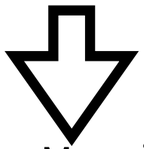
3:00

How does “half-precision” floating point changes the roofline and system design?

16-bit floating point matrix multiplications

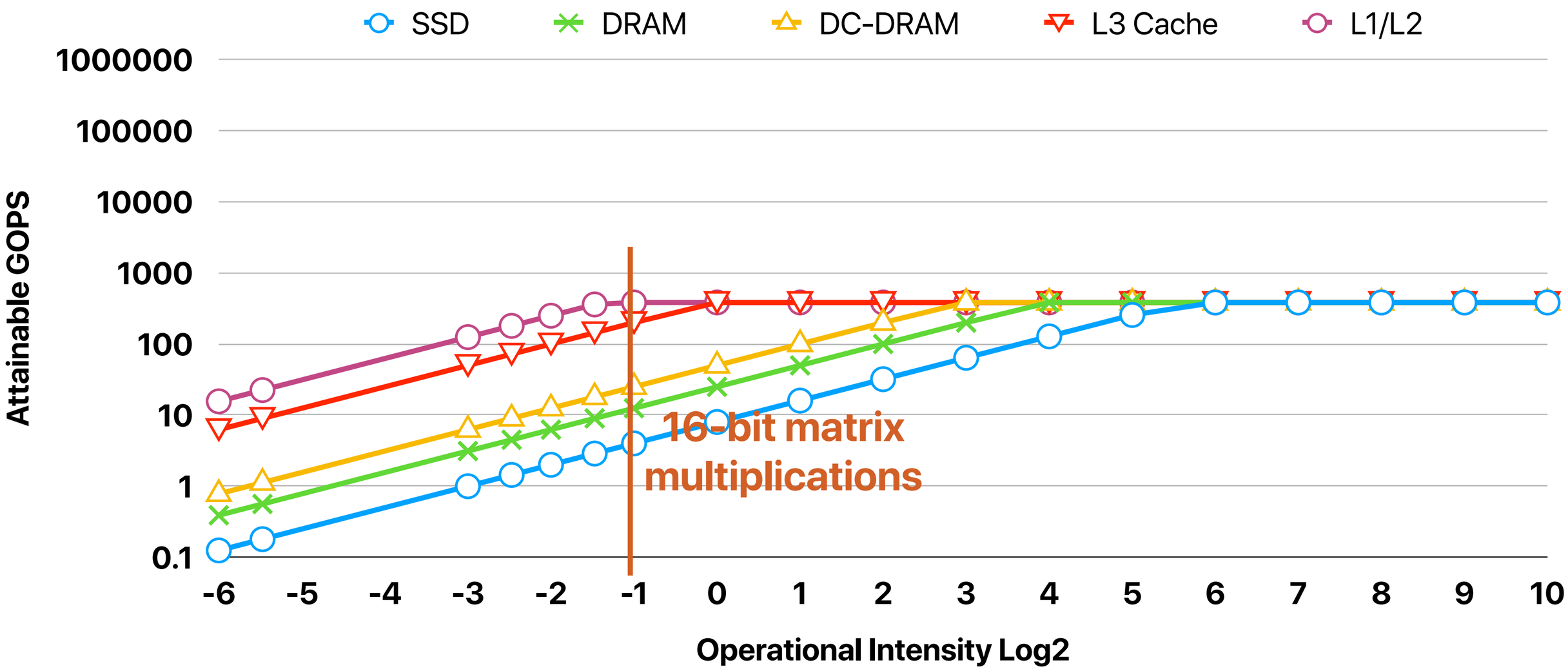
- Performing $M \times N \times P$ matrix multiplications
  - The size of a is $M \times N$
  - The size of b is $N \times P$
  - The size of c is $M \times P$
- Total amount of operations = $2 \times M \times N \times P$ ops
- Total size of data accesses = $2 \times (2 \times N + 1) \times M \times P$ bytes
- Average operations per byte = $\frac{2 \times M \times N \times P}{2 \times (2 \times N + 1) \times M \times P} = \frac{1}{2} = 0.5$

```
for(i = 0; i < M; i++) {
  for(j = 0; j < P; j++) {
    for(k = 0; k < N; k++) {
      c[i][j] += a[i][k]*b[k][j];
    }
  }
}
```



```
for(i = 0; i < M; i++) {
  for(j = 0; j < P; j++) {
    sum = 0.0;
    for(k = 0; k < N; k++) {
      sum += a[i][k]*b[k][j];
    }
    c[i][j] = sum;
  }
}
```

Where does 16-bit sit?

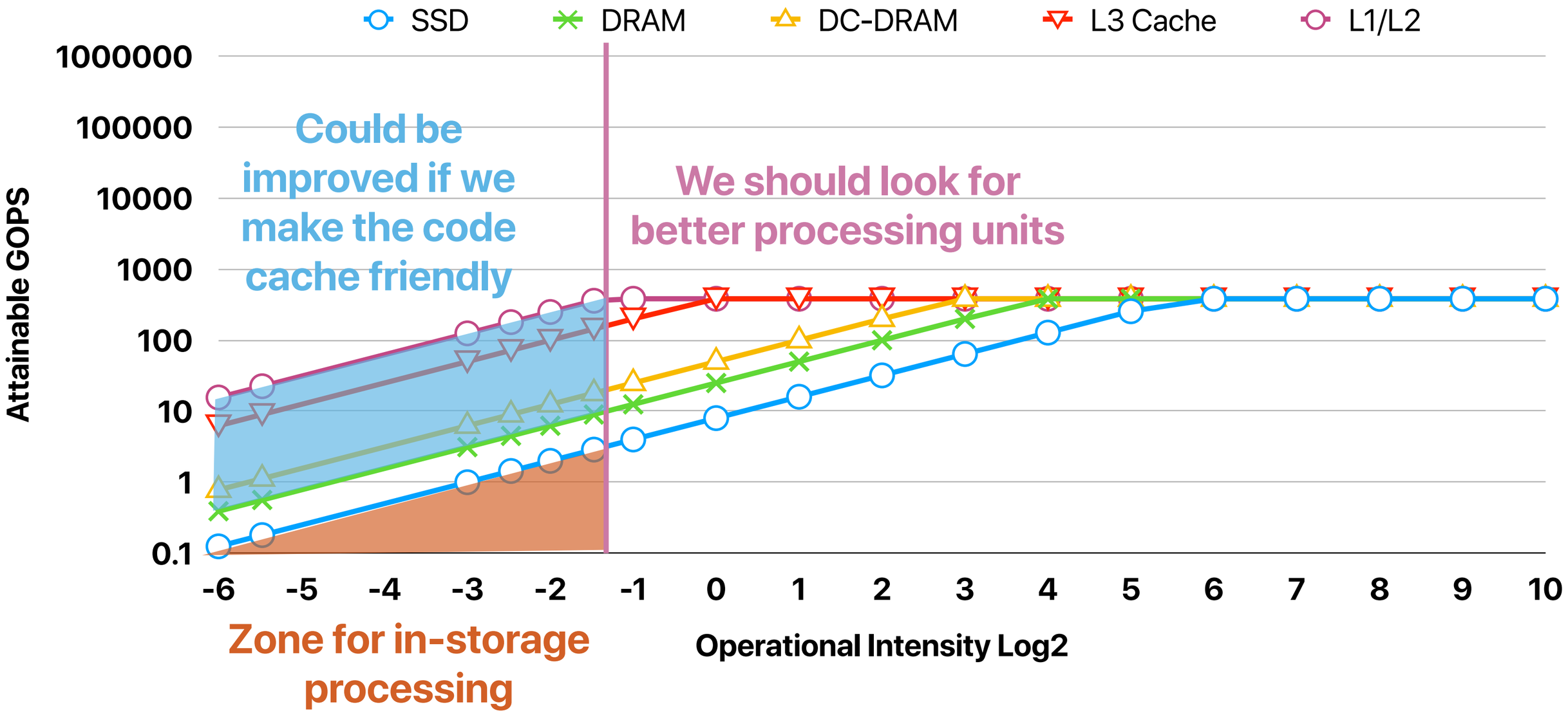


Reduced precision makes a problem more “compute-bound” and less memory bandwidth dependent

3:00

Recap on the roofline model, how do you decide if we want to go for improved processors or alternative architectures (e.g., near-data processing)

Where is my application?



Summary of the roofline model

- The roofline model defines the “best performance” we can achieve under a certain architecture
  - Tells us how good we’re in optimizing our code
  - We can never surpass the roofline without changing the hardware
- The roofline model helps us to identify which direction to go for optimization
  - Memory-bounded
    - Can we more efficiently use cache?
    - Can we reduce the data volume?
  - Compute-bounded
    - Can we make the algorithm more efficient?
    - Can we use more or other processing units?

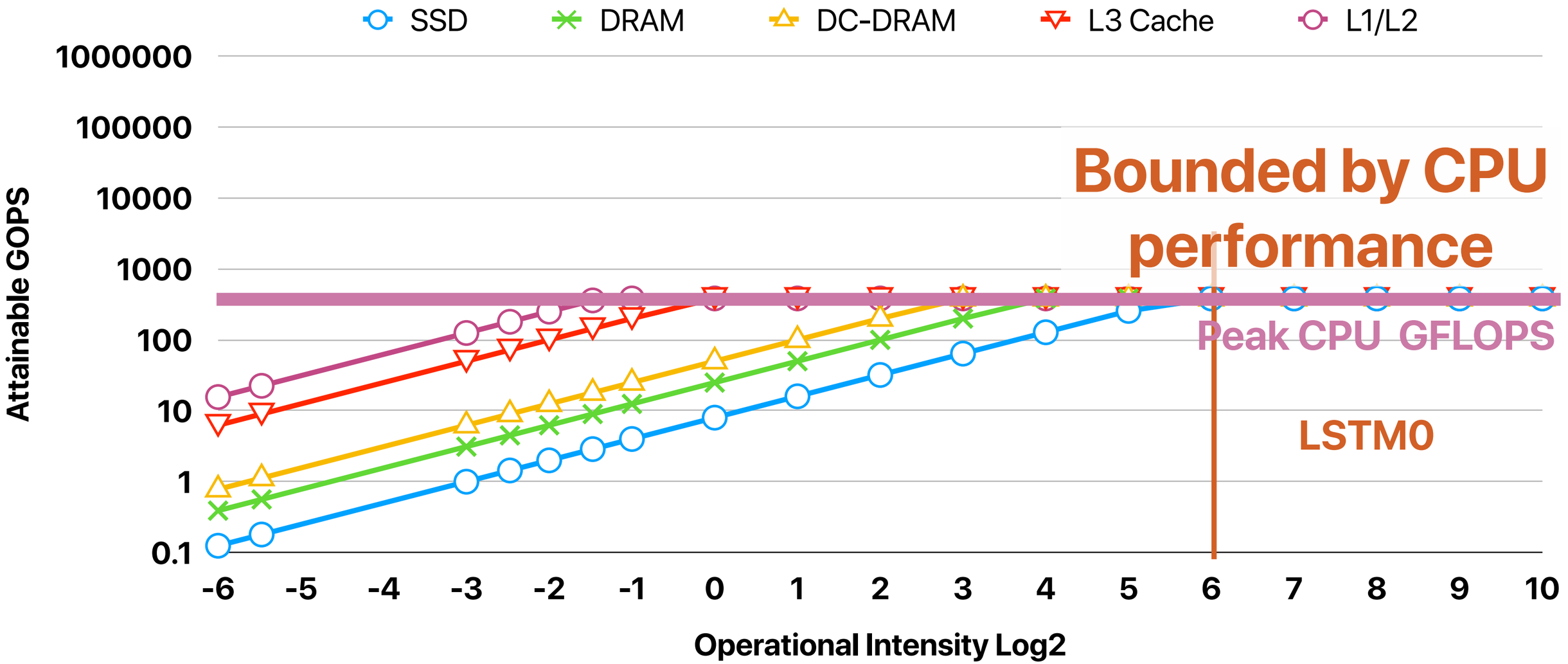
What's the operational intensity of modern workloads?

Table 1. Six DNN applications (two per DNN type). MLP stands for multi-layer perceptron, and LSTM stands for long short-term memory.

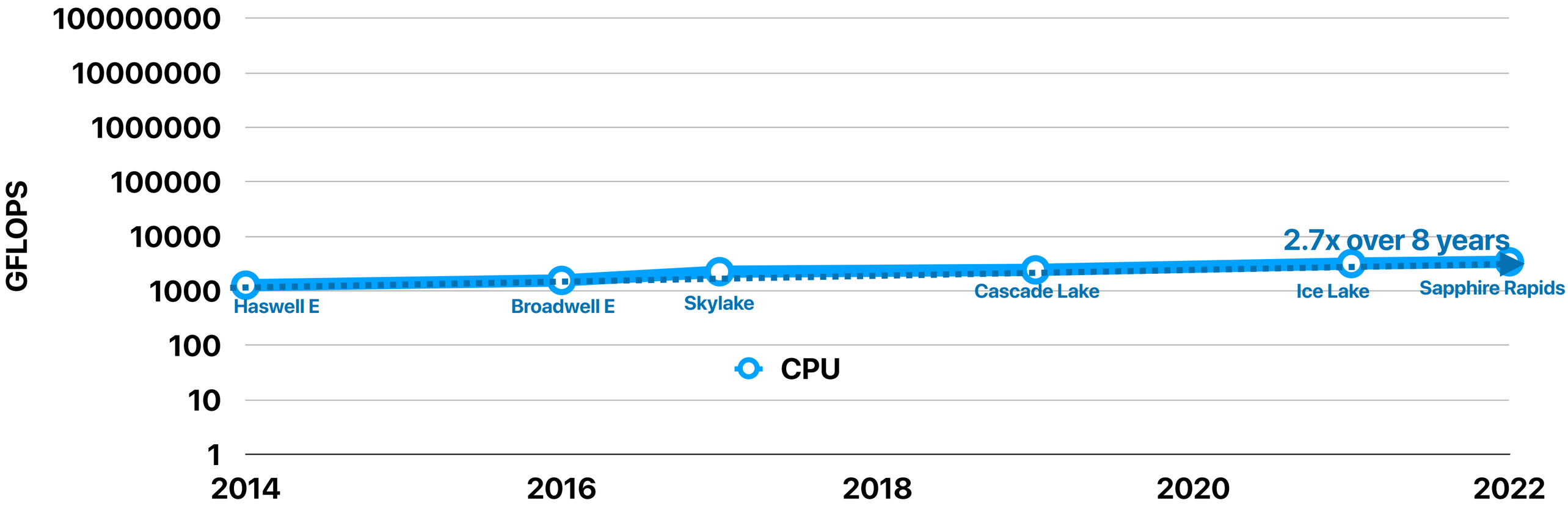
Name	Lines of Code	Weights	TPU Ops / Weight Byte	Operational Intensity Log2
MLP0	100	20 M	200	7.64385618977472
MLP1	1,000	5 M	168	7.39231742277876
LSTM0	1,000	52 M	64	6
LSTM1	1,500	34 M	96	6.58496250072116
CNN0	1,000	8 M	2,888	11.4958550268872
CNN1	1,000	100 M	1,750	10.7731392067197

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, May./Jun. 2018, doi: 10.1109/MM.2018.032271057.

Where is LSTM0 in the roofline?



Performance improvements of CPUs/GPUs in the recent decode



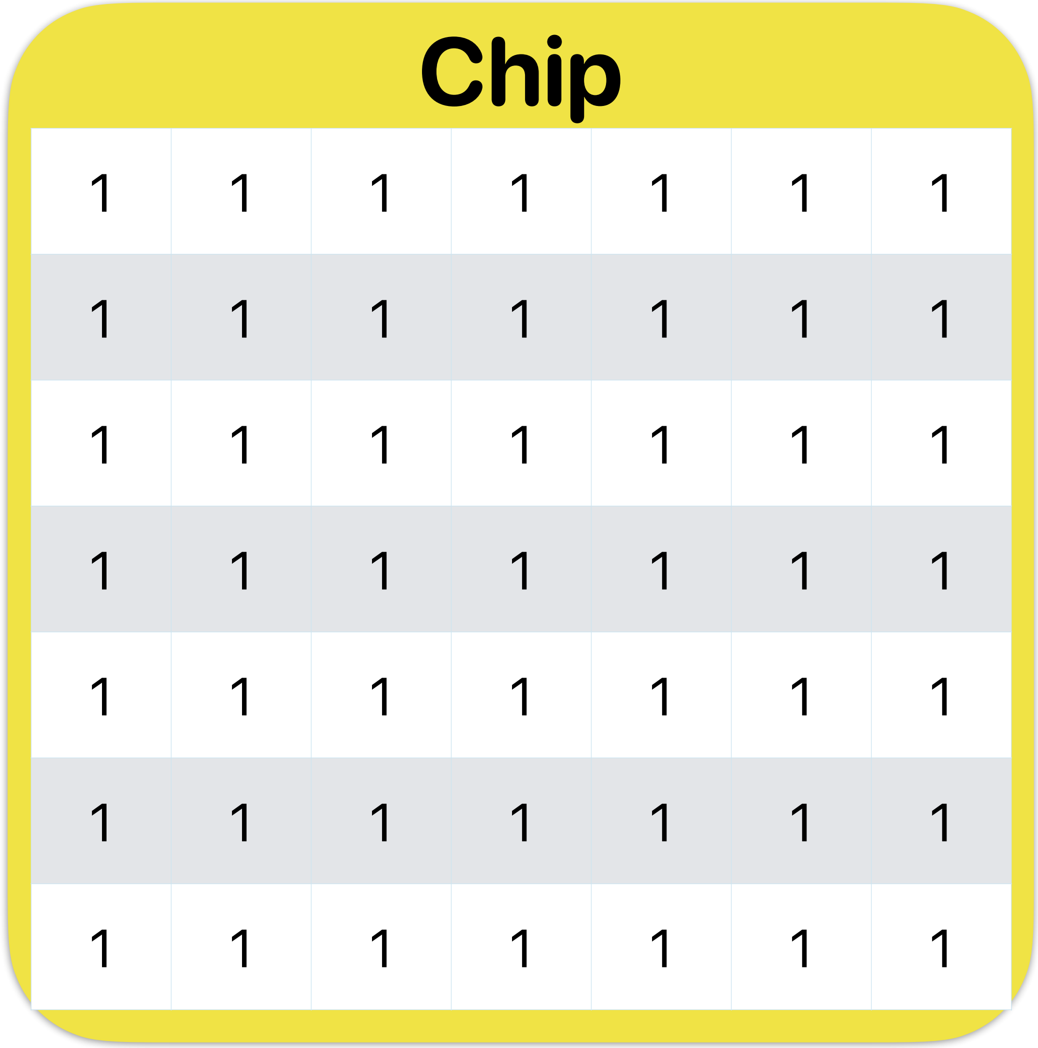
https://ourworldindata.org/grapher/artificial-intelligence-training-computation

Dennardian Broken

- Given a scaling factor S

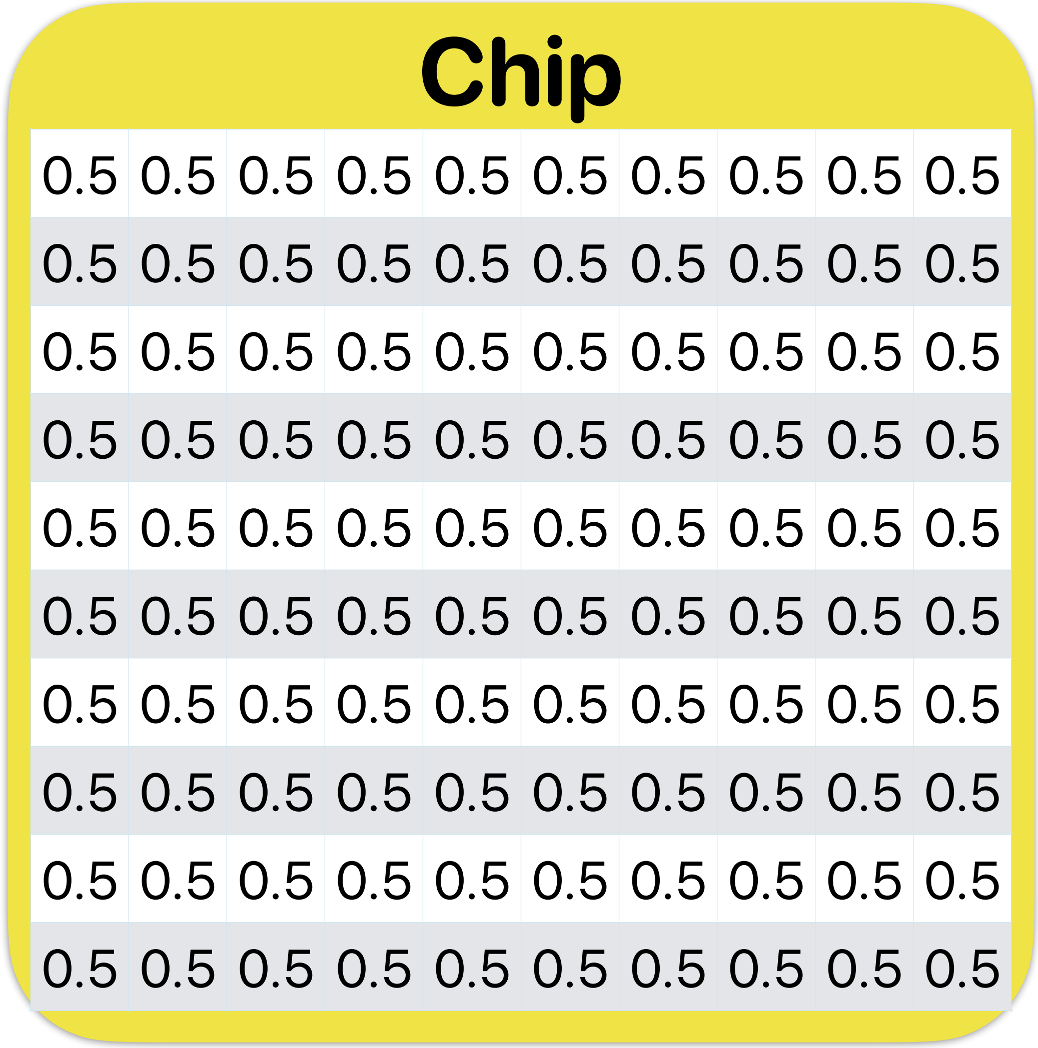
Parameter	Relation	Classical Scaling	Leakage Limited
Power Budget		1	1
Chip Size		1	1
Vdd (Supply Voltage)		1/S	1
Vt (Threshold Voltage)	1/S	1/S	1
tex (oxide thickness)		1/S	1/S
W, L (transistor dimensions)		1/S	1/S
Cgate (gate capacitance)	WL/tox	1/S	1/S
Isat (saturation current)	WVdd/tox	1/S	1
F (device frequency)	Isat/(CgateVdd)	S	S
D (Device/Area)	1/(WL)	S^2	S^2
p (device power)	IsatVdd	$1/S^2$	1
P (chip power)	Dp	1	S^2
U (utilization)	1/P	1	$1/S^2$

Recap: Power consumption to light on all transistors



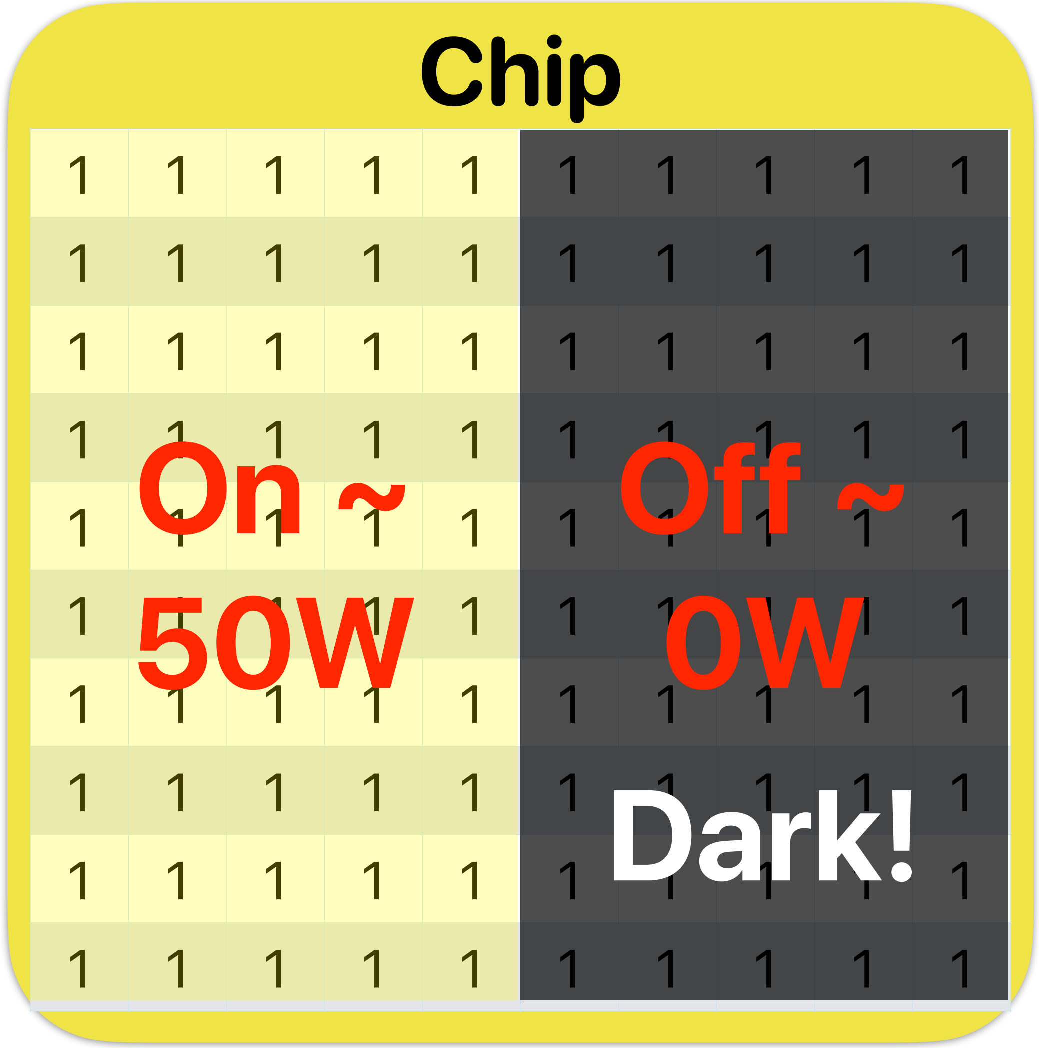
=49W

Dennardian Scaling



=50W

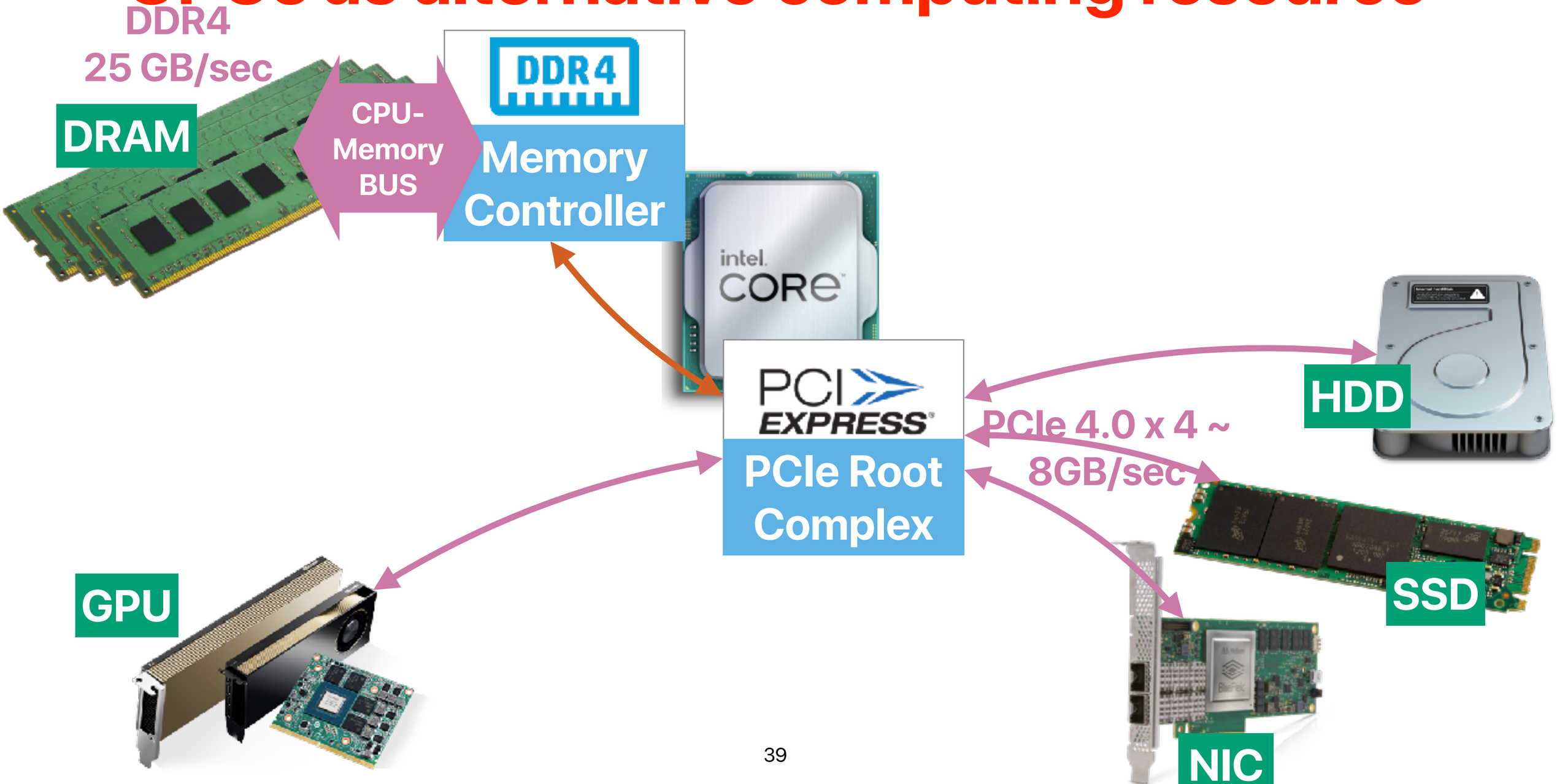
Dennardian Broken



=100W!

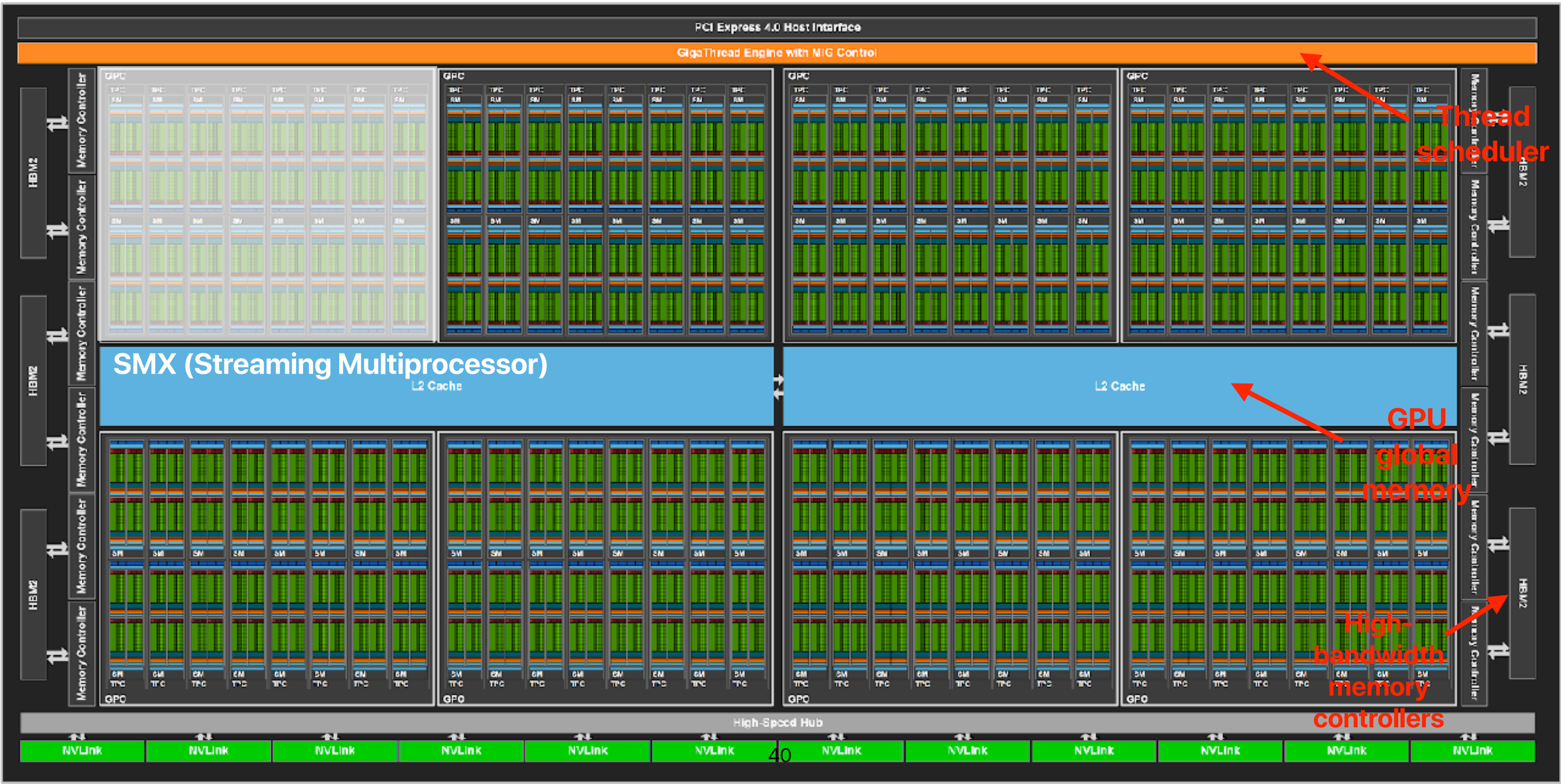
Alternative computing resources

GPUs as alternative computing resource



https://developer.nvidia.com/blog/nvidia-ampere-architecture-in-depth/

GPU Architecture



Inside an SM

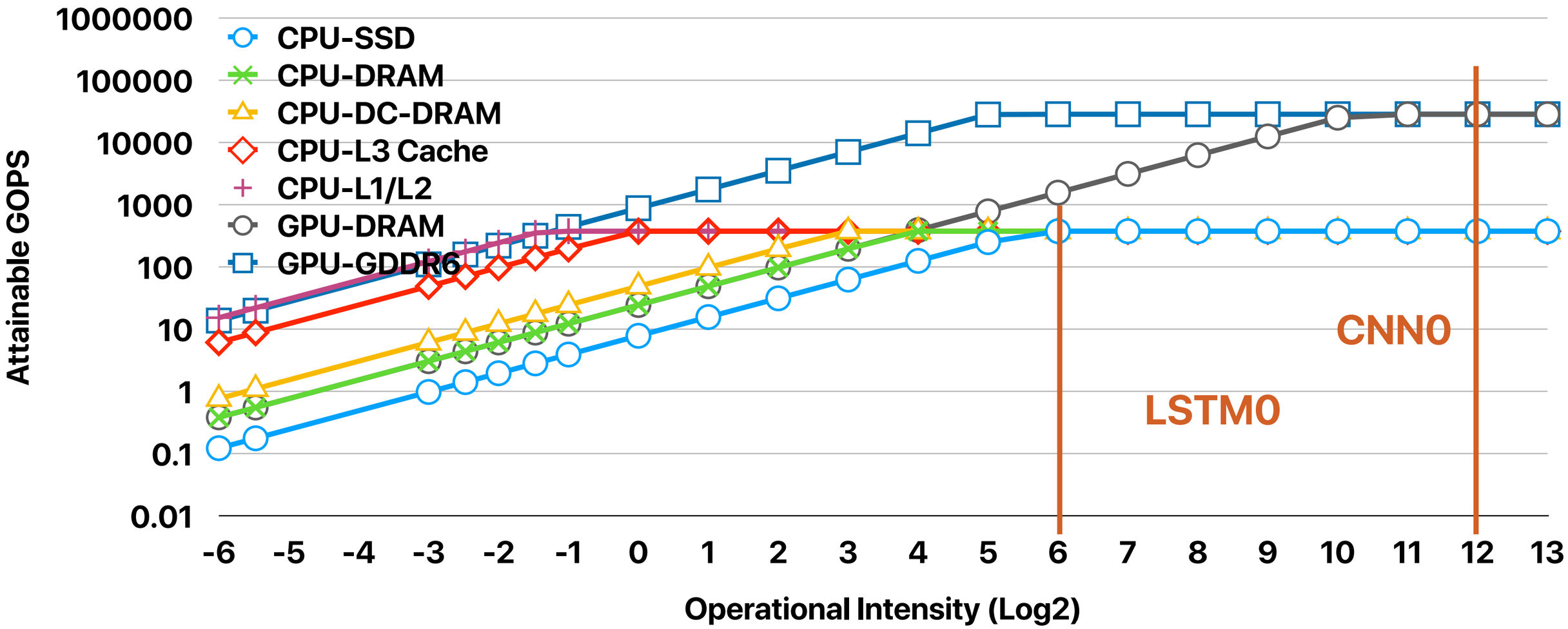
A total of 16*4 = 64 FP32 cores
A total of 16*4 = 64 INT32 cores
A total of 16*4 = 16 FP64 cores

- All of these can only perform the same operation at the same time, but each of these is named as a “thread” in CUDA
- You can only use either FP32, FP64, INT32 and “Tensor Cores” at the same time

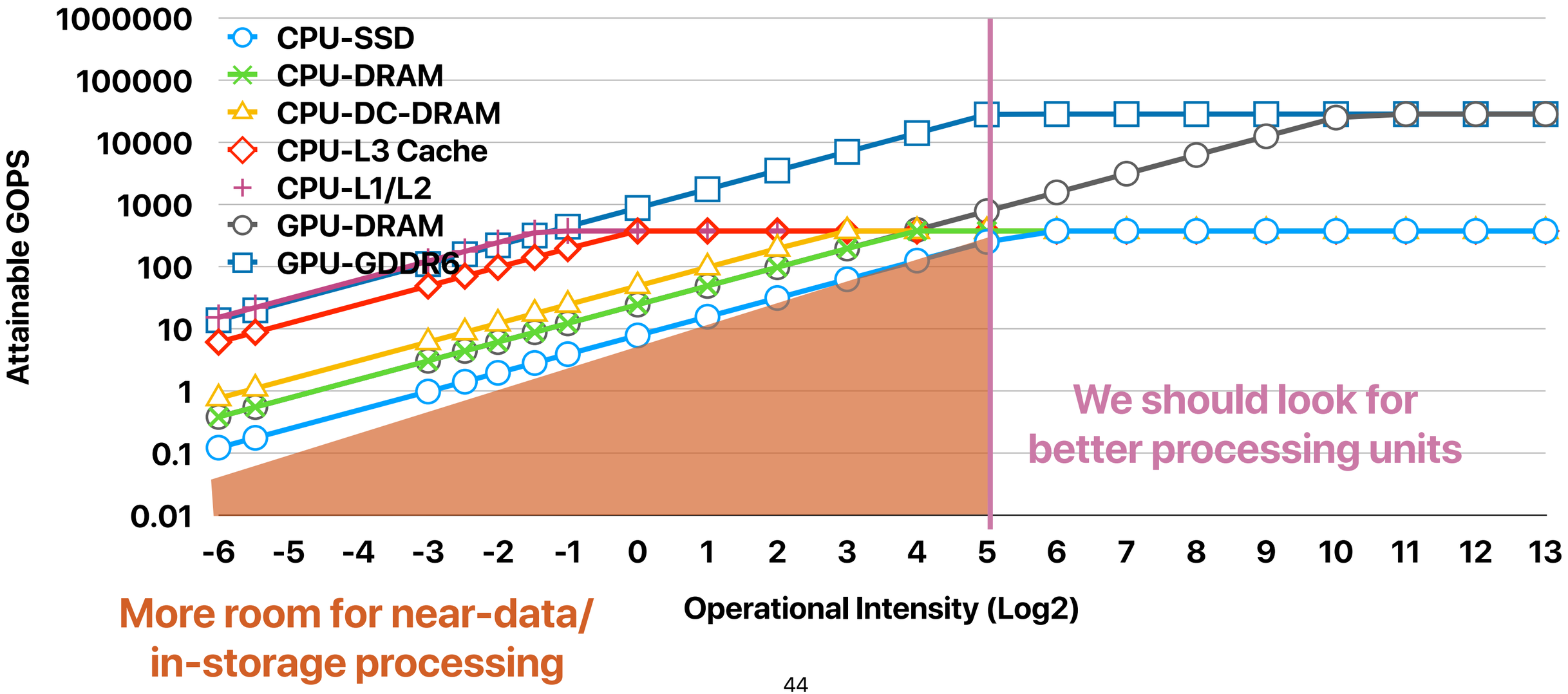
3:00

How do GPUs change the roofline model?

Placing GPUs in the roofline



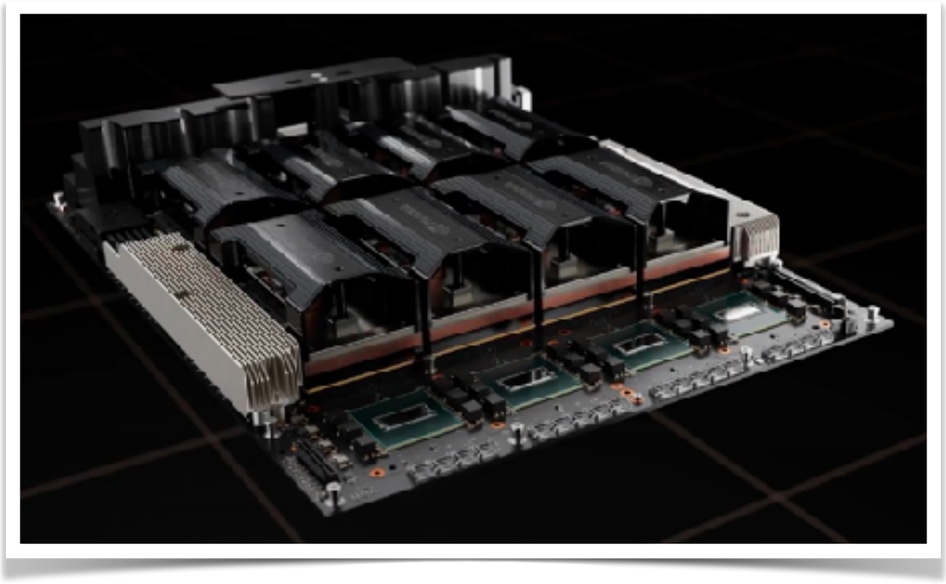
Placing GPUs in the roofline



Power/energy is the main issue

NVIDIA Accelerator Specification Comparison			
	H100	A100 (80GB)	V100
FP32 CUDA Cores	16896	6912	5120
Tensor Cores	528	432	640
Boost Clock	~1.78GHz (Not Finalized)	1.41GHz	1.53GHz
Memory Clock	4.8Gbps HBM3	3.2Gbps HBM2e	1.75Gbps HBM2
Memory Bus Width	5120-bit	5120-bit	4096-bit
Memory Bandwidth	3TB/sec	2TB/sec	900GB/sec
VRAM	80GB	80GB	16GB/32GB
FP32 Vector	60 TFLOPS	19.5 TFLOPS	15.7 TFLOPS
FP64 Vector	30 TFLOPS	9.7 TFLOPS (1/2 FP32 rate)	7.8 TFLOPS (1/2 FP32 rate)
INT8 Tensor	2000 TOPS	624 TOPS	N/A
FP16 Tensor	1000 TFLOPS	312 TFLOPS	125 TFLOPS
TF32 Tensor	500 TFLOPS	156 TFLOPS	N/A
FP64 Tensor	60 TFLOPS	19.5 TFLOPS	N/A
Interconnect	NVLink 4 18 Links (900GB/sec)	NVLink 3 12 Links (600GB/sec)	NVLink 2 6 Links (300GB/sec)
GPU	GH100 (814mm2)	GA100 (826mm2)	GV100 (815mm2)
Transistor Count	80B	54.2B	21.1B
TDP	700W	400W	300W/350W
Manufacturing Process	TSMC 4N	TSMC 7N	TSMC 12nm FFN
Interface	SXM5	SXM4	SXM2/SXM3
Architecture	Hopper	Ampere	Volta

https://www.workstationspecialist.com/product/nvidia-tesla-a100/



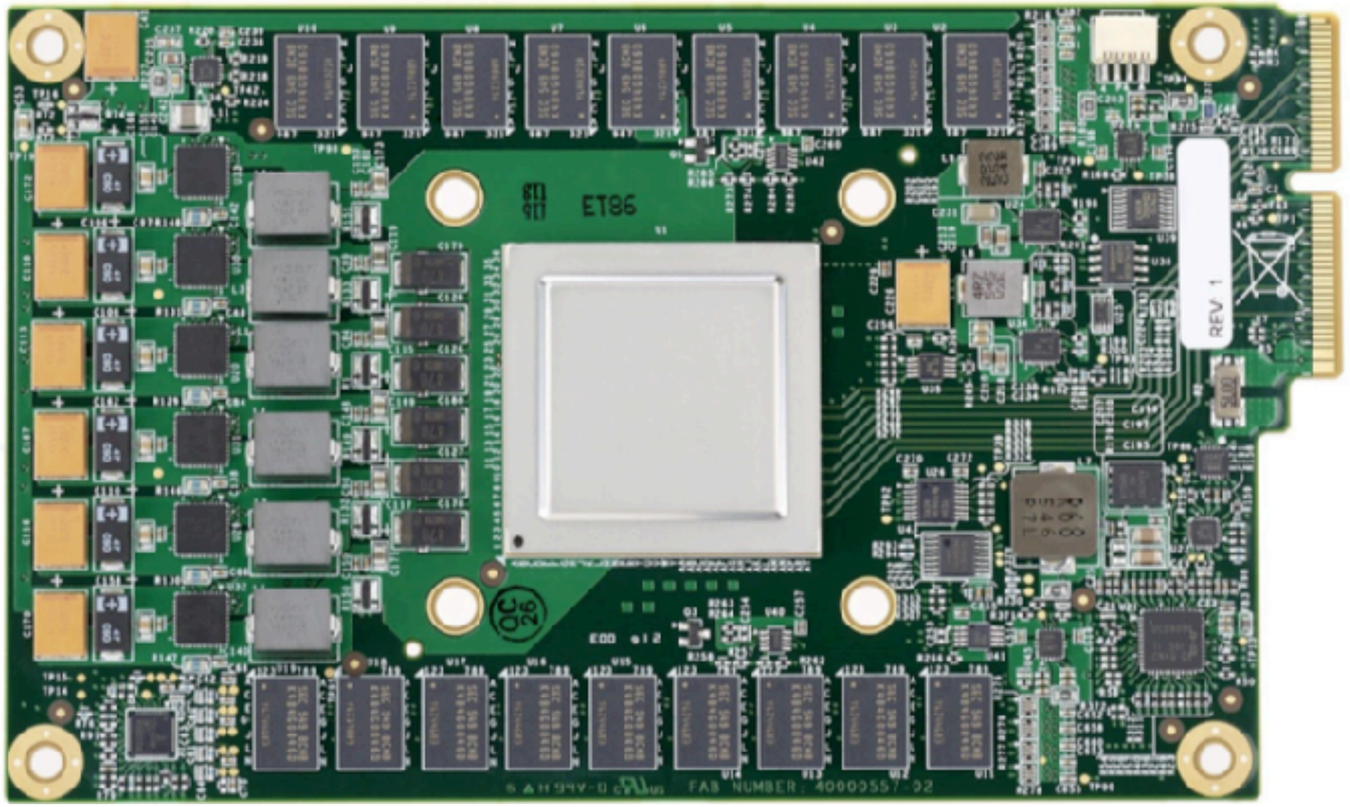
https://www.servethehome.com/wp-content/uploads/2022/03/NVIDIA-GTC-2022-H100-in-HGX-H100.jpg

Can we be fast but also power/ energy efficient?

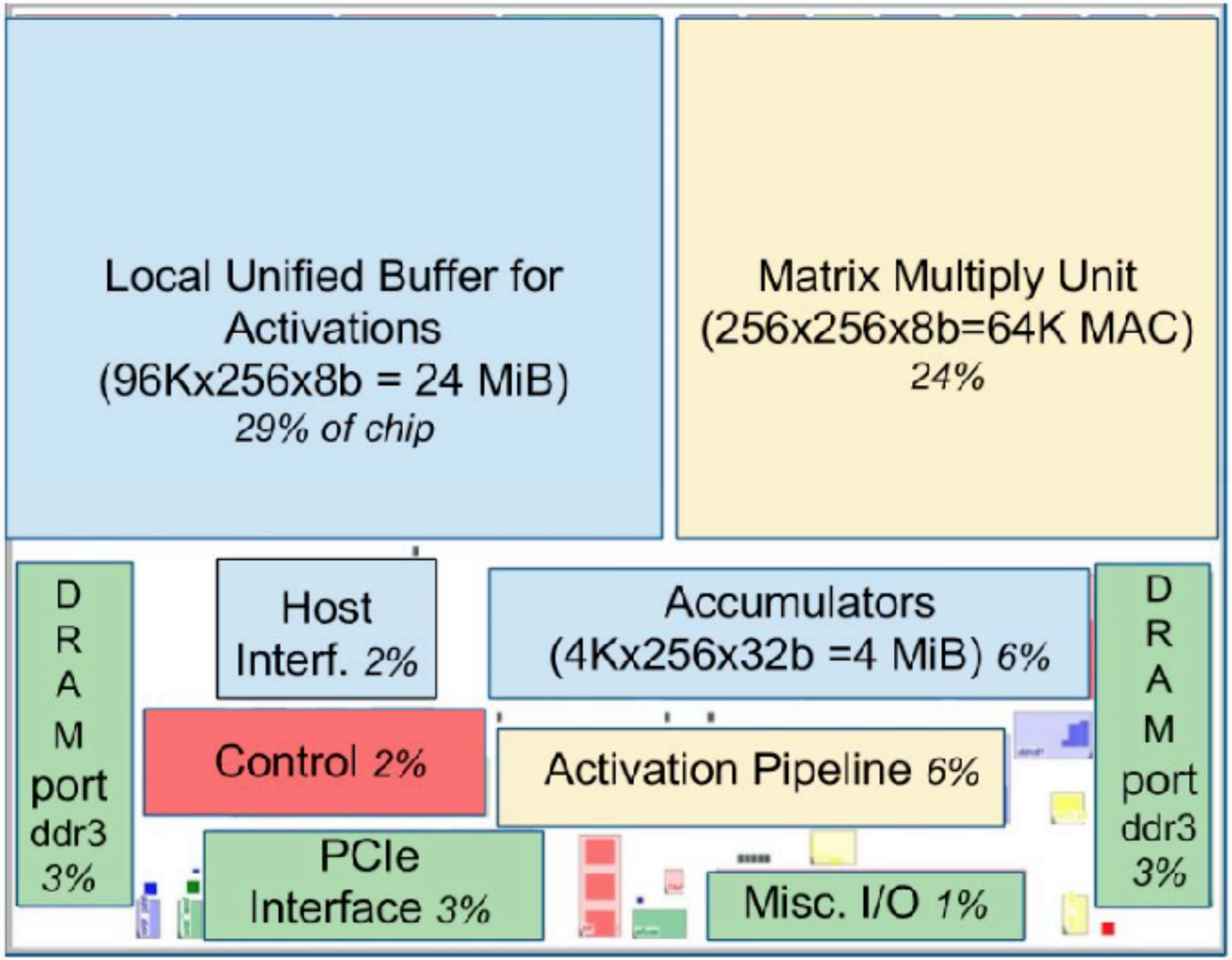
Example of a hardware accelerator — Tensor Processing Unit

- 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.

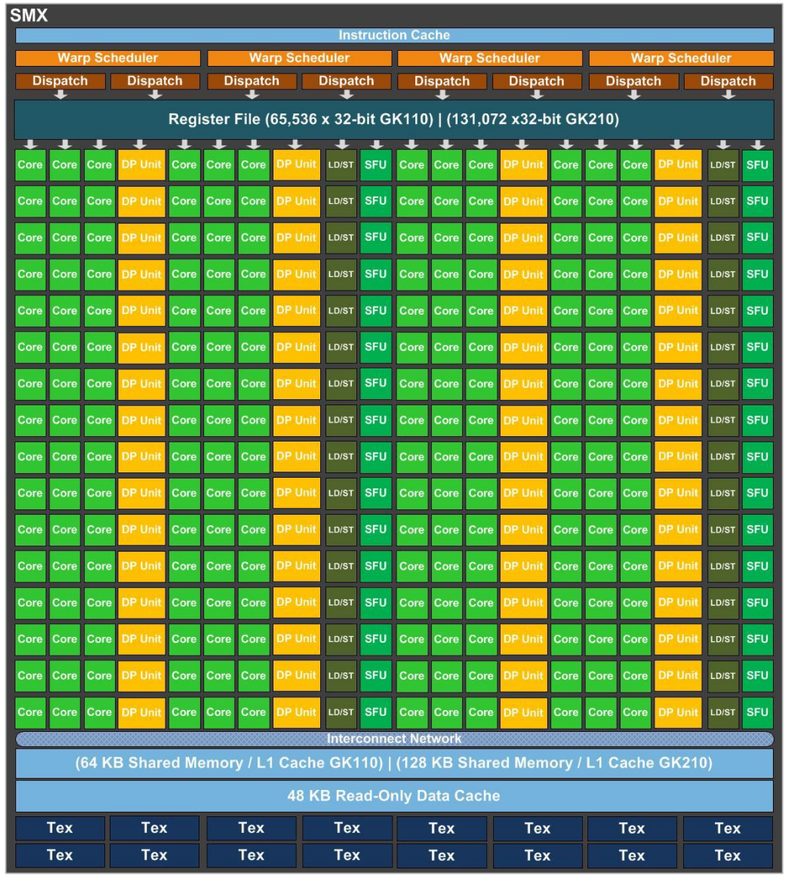
What the “first” generation of TPU looks like

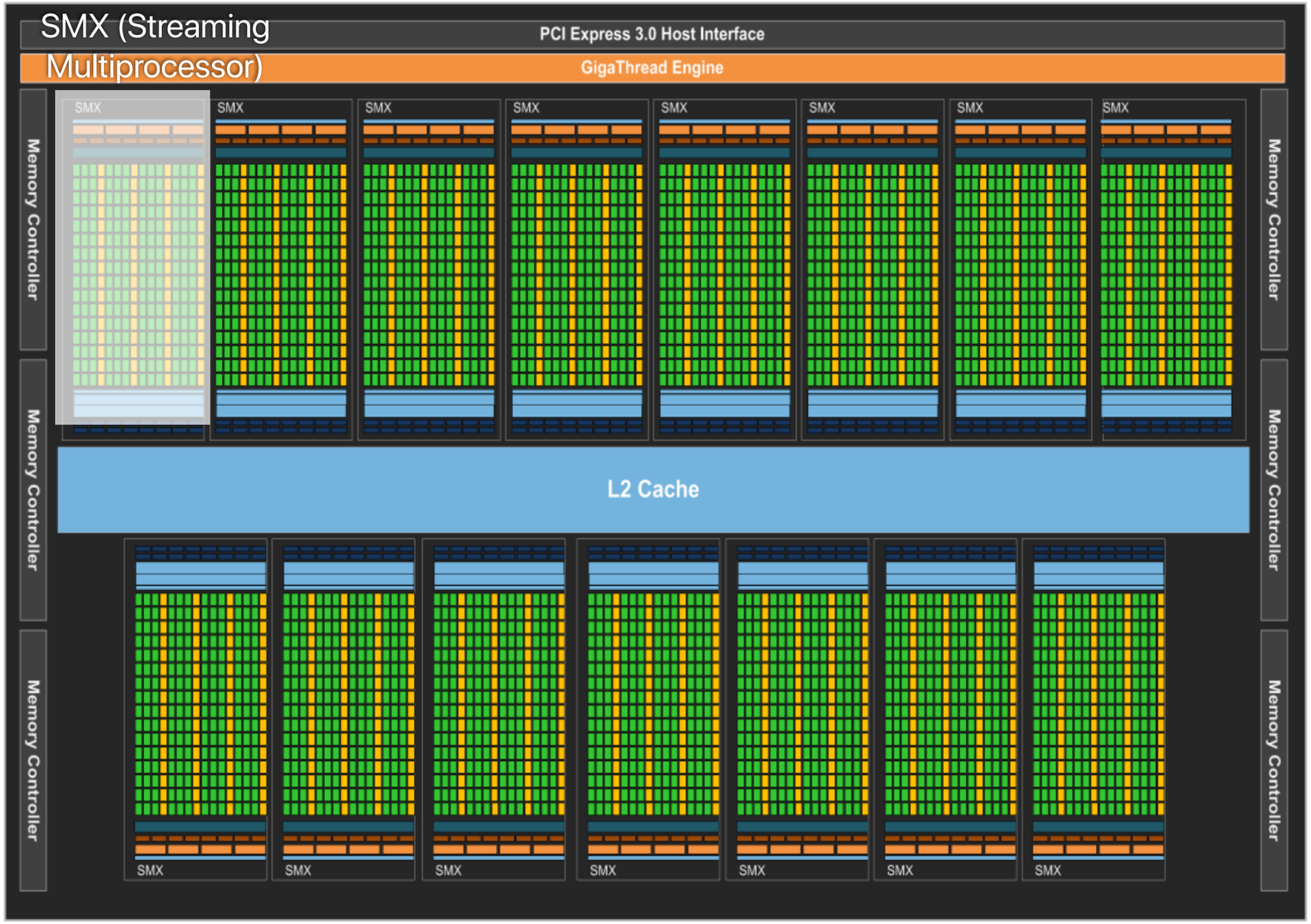


TPU Floorplan



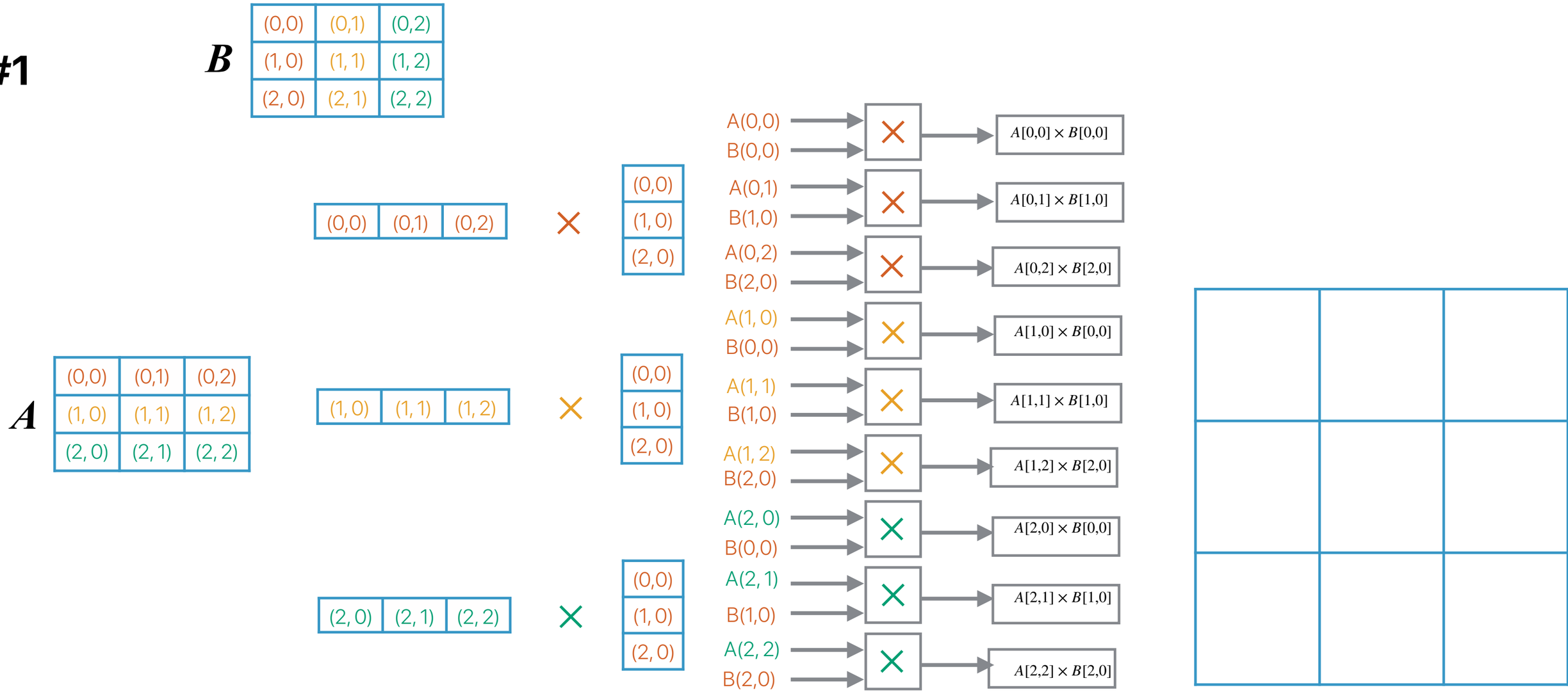
Vector processing model by GPUs





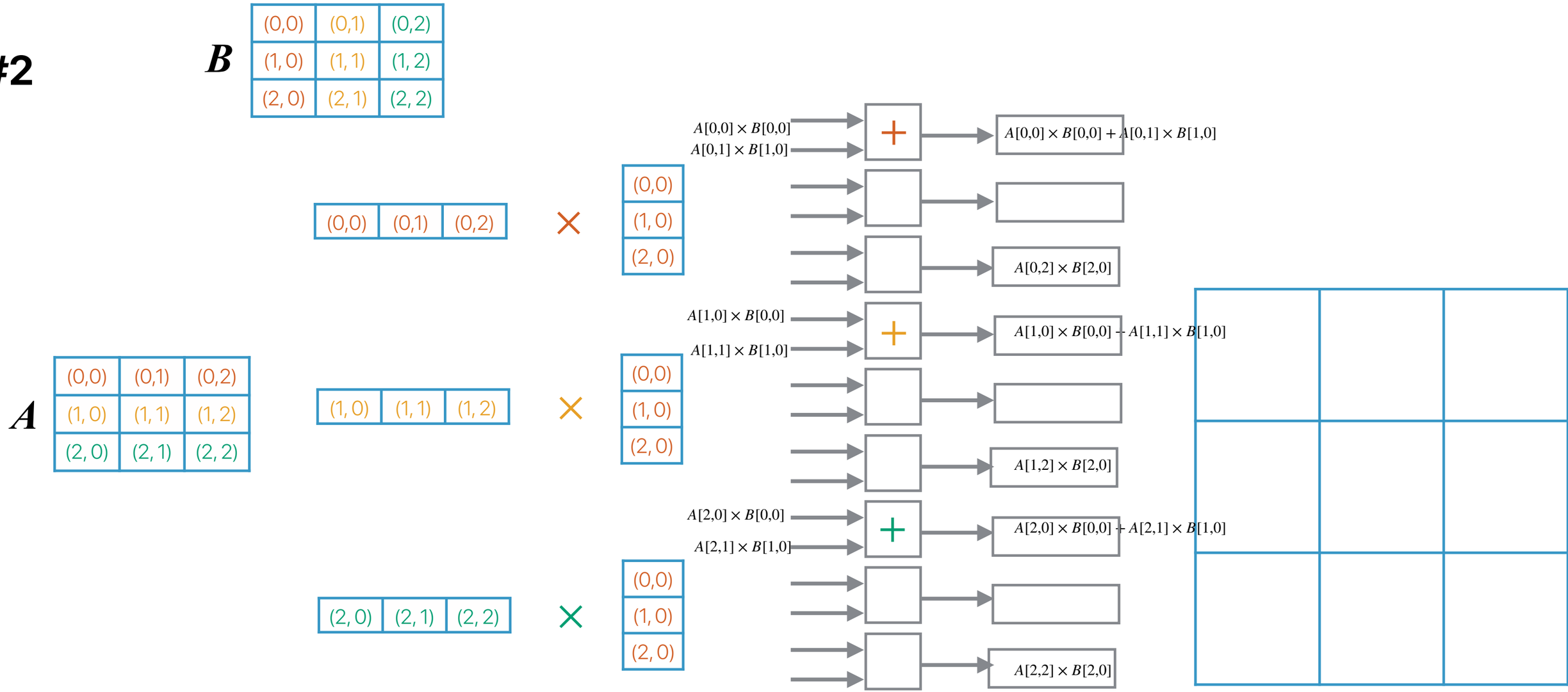
Vector processing for MM

#1



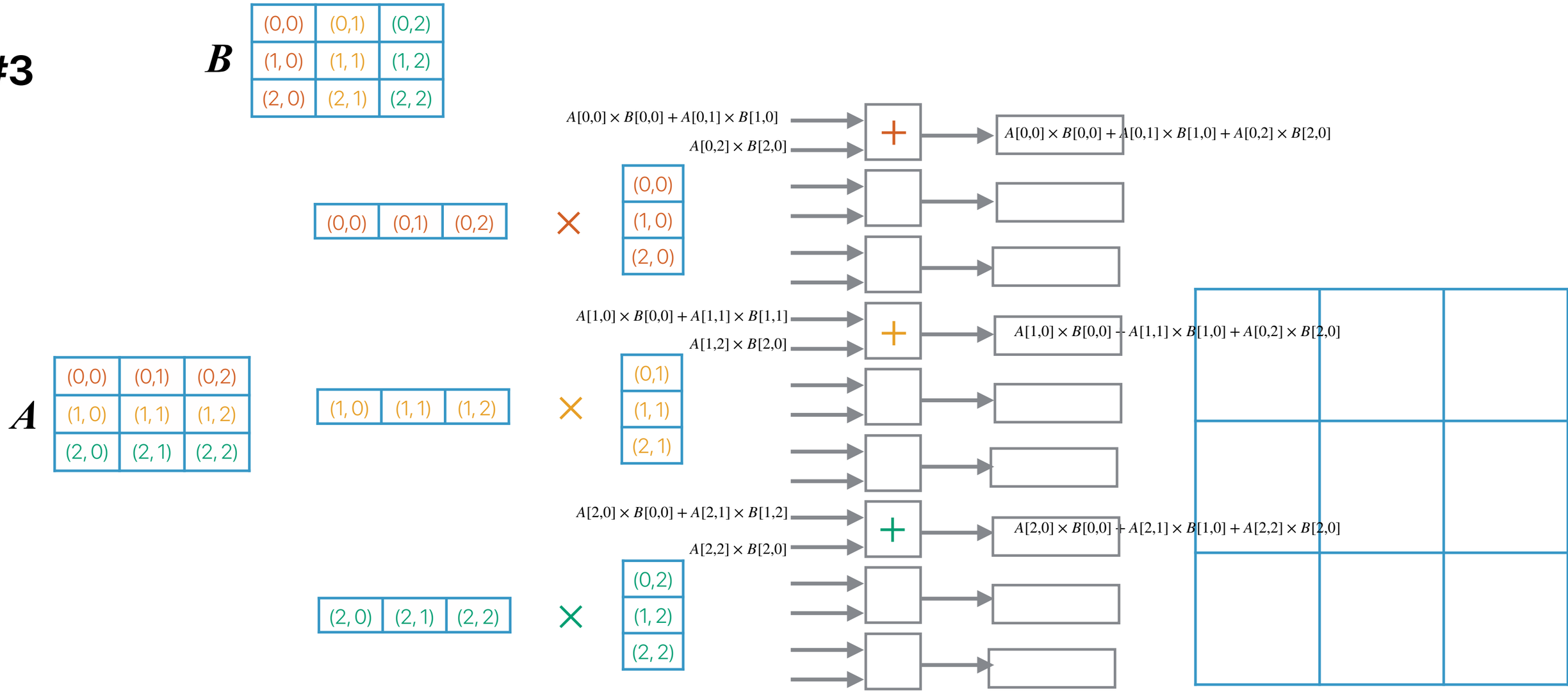
Vector processing for MM

#2



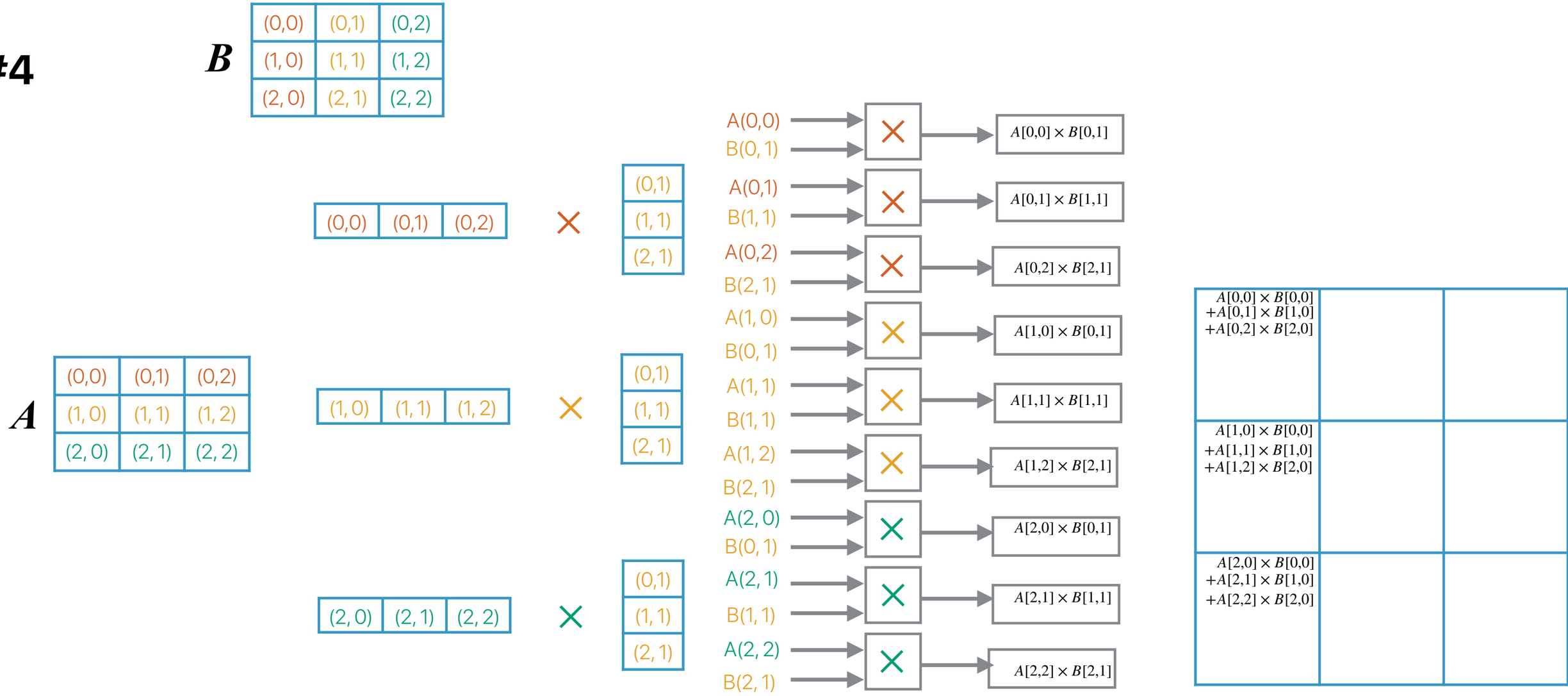
Vector processing for MM

#3



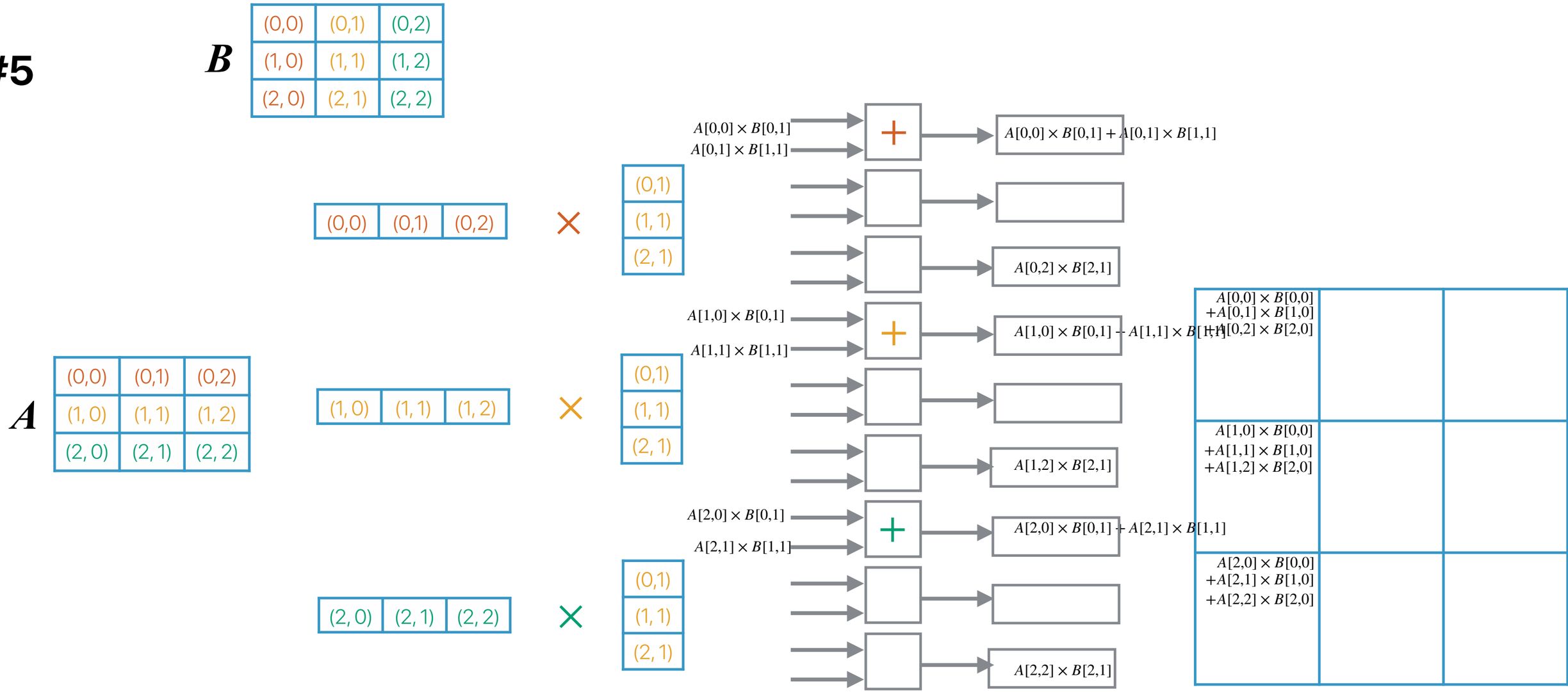
Vector processing for MM

#4



Vector processing for MM

#5



Vector processing for MM

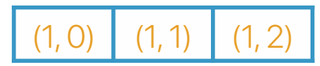
#6

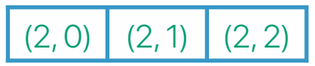
B

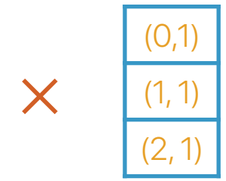
(0,0)	(0,1)	(0,2)
(1, 0)	(1, 1)	(1, 2)
(2, 0)	(2, 1)	(2, 2)

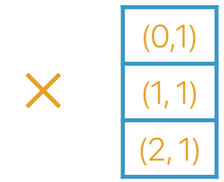
A

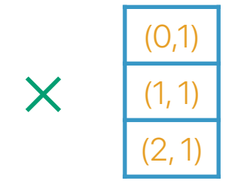
(0,0)	(0,1)	(0,2)
(1, 0)	(1, 1)	(1, 2)
(2, 0)	(2, 1)	(2, 2)







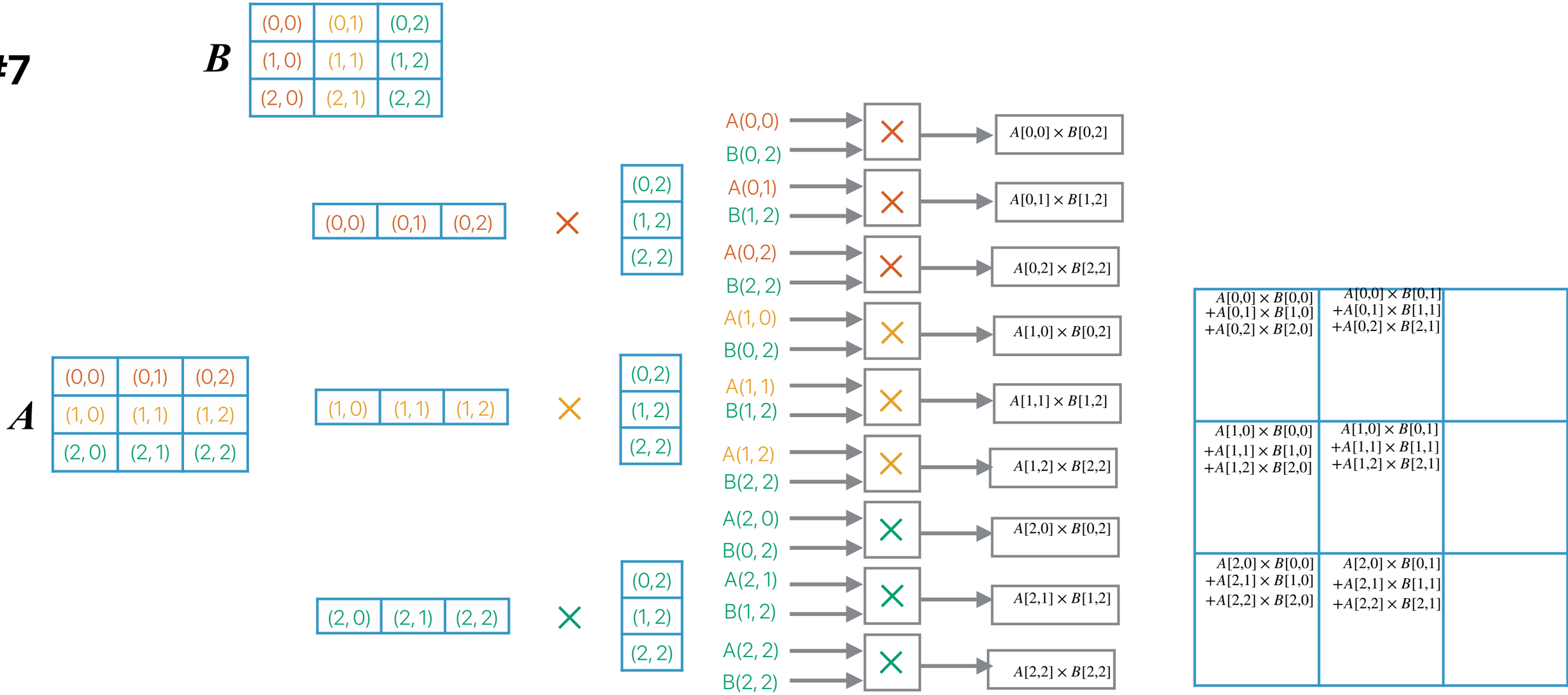




$A[0,0] \times B[0,0]$ $+A[0,1] \times B[1,0]$ $+A[0,2] \times B[2,0]$	$A[0,0] \times B[0,1]$ $+A[0,1] \times B[1,1]$ $+A[0,2] \times B[2,1]$	
$A[1,0] \times B[0,0]$ $+A[1,1] \times B[1,0]$ $+A[1,2] \times B[2,0]$	$A[1,0] \times B[0,1]$ $+A[1,1] \times B[1,1]$ $+A[1,2] \times B[2,1]$	
$A[2,0] \times B[0,0]$ $+A[2,1] \times B[1,0]$ $+A[2,2] \times B[2,0]$	$A[2,0] \times B[0,1]$ $+A[2,1] \times B[1,1]$ $+A[2,2] \times B[2,1]$	

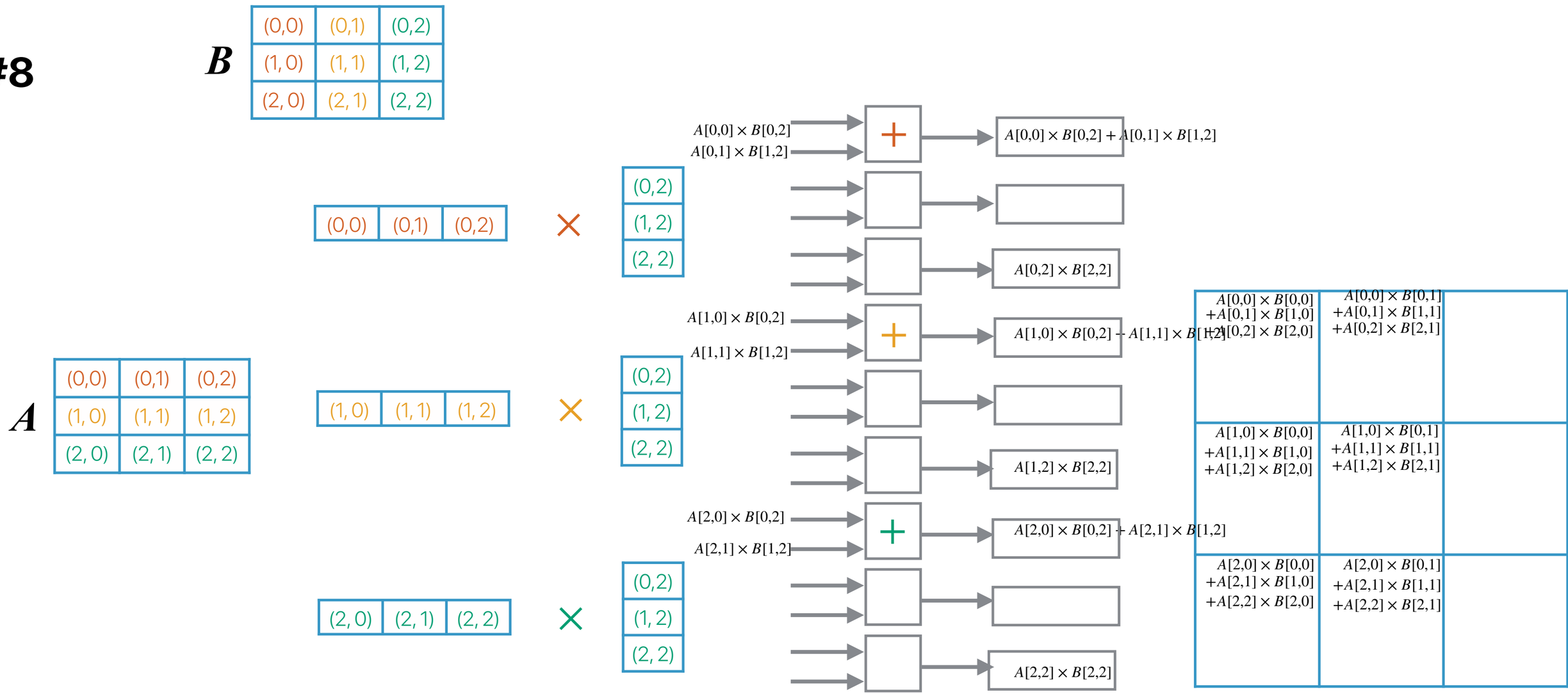
Vector processing for MM

#7



Vector processing for MM

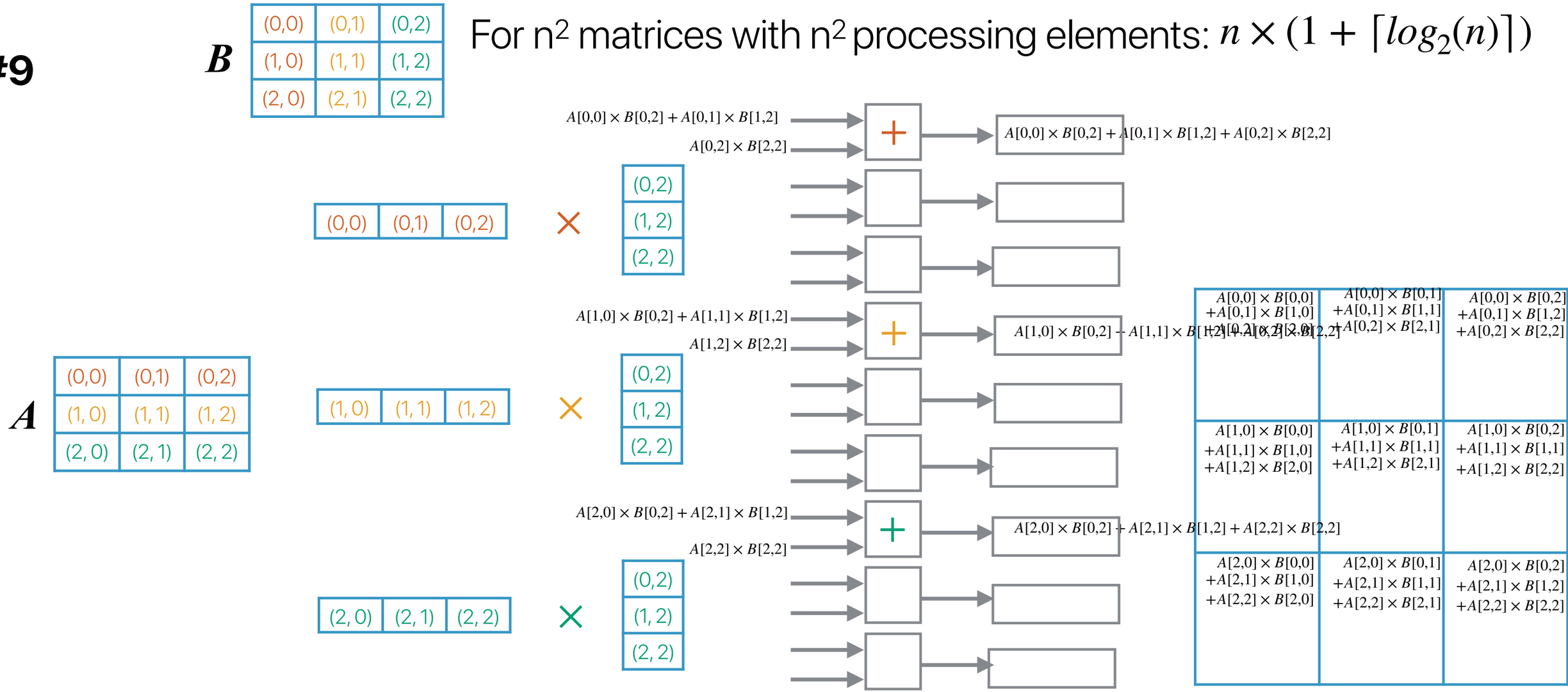
#8



Vector processing for MM

#9

For n^2 matrices with n^2 processing elements: $n \times (1 + \lceil log_2(n) \rceil)$



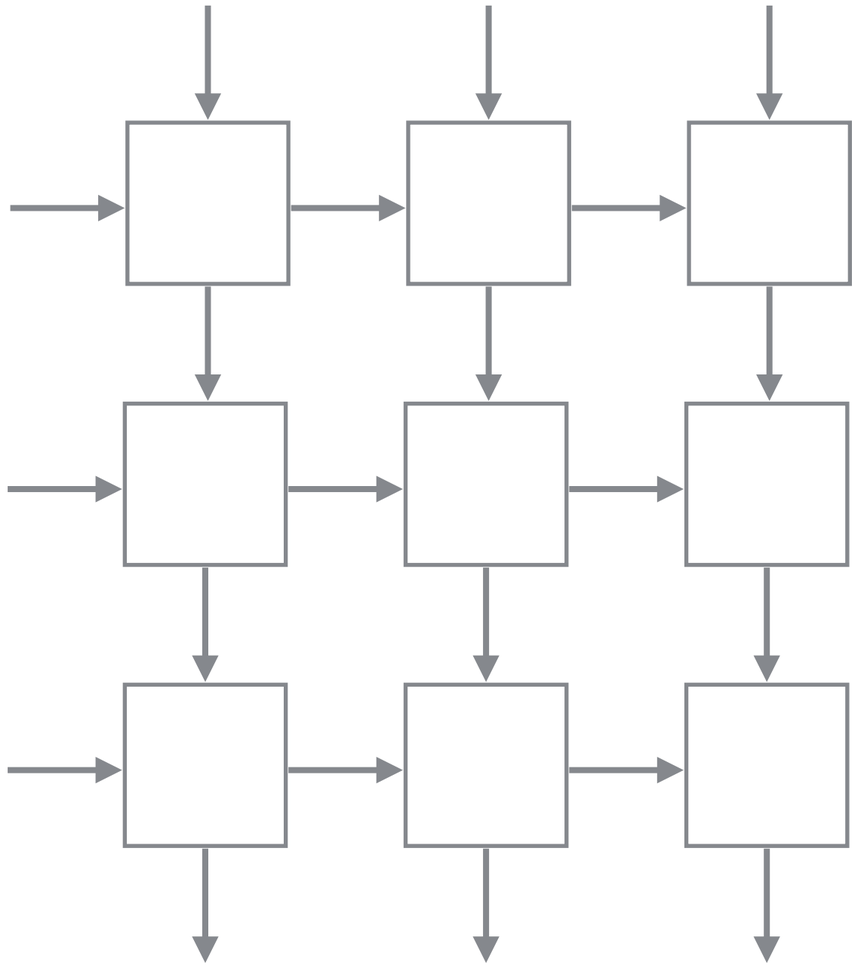
Systolic Arrays used by AI/ML Accelerators

B

(0,0)	(0,1)	(0,2)
(1, 0)	(1, 1)	(1, 2)
(2, 0)	(2, 1)	(2, 2)

A

(0,0)	(0,1)	(0,2)
(1, 0)	(1, 1)	(1, 2)
(2, 0)	(2, 1)	(2, 2)



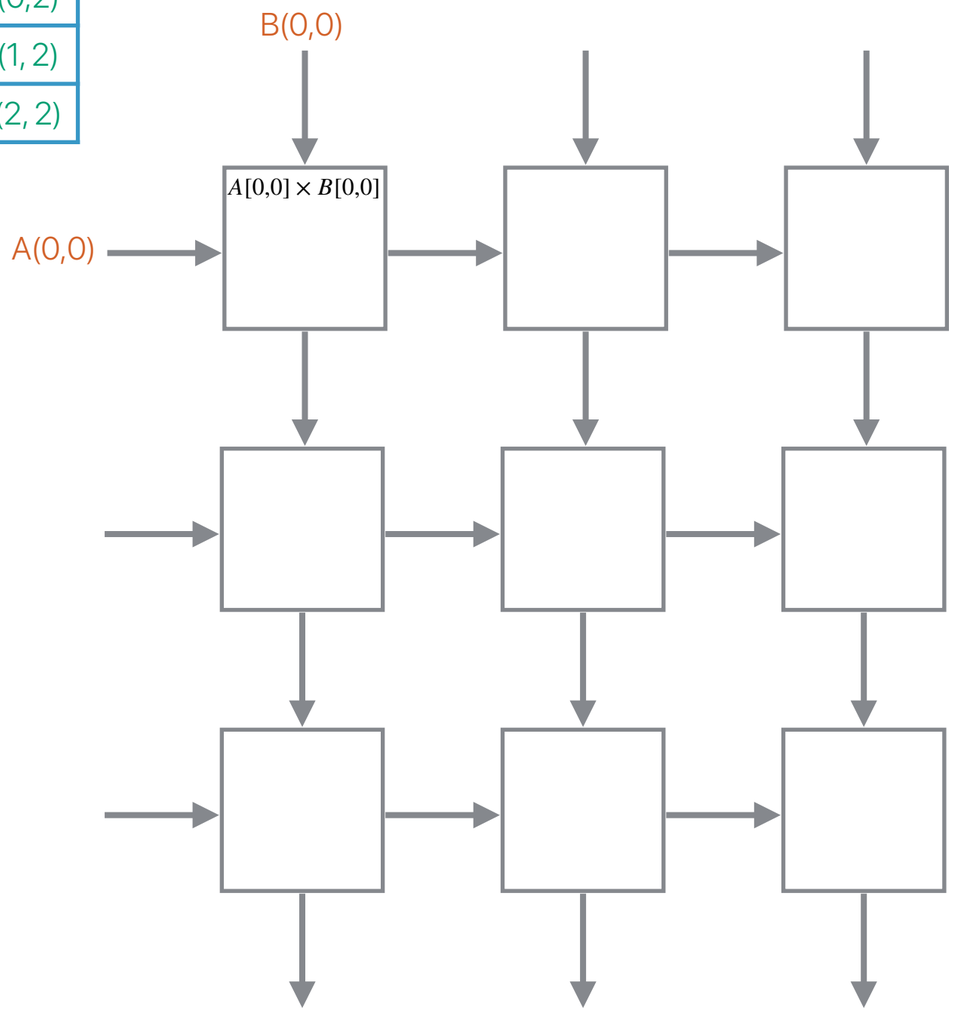
H.T. Kung, "Why systolic architectures?," in IEEE Computer, vol. 15, no. 1, pp. 37-46, Jan. 1982, doi: 10.1109/MC.1982.1653825.

Systolic Arrays used by AI/ML Accelerators

#1

B

(0,0)	(0,1)	(0,2)
(1, 0)	(1, 1)	(1, 2)
(2, 0)	(2, 1)	(2, 2)



A

(0,0)	(0,1)	(0,2)
(1, 0)	(1, 1)	(1, 2)
(2, 0)	(2, 1)	(2, 2)

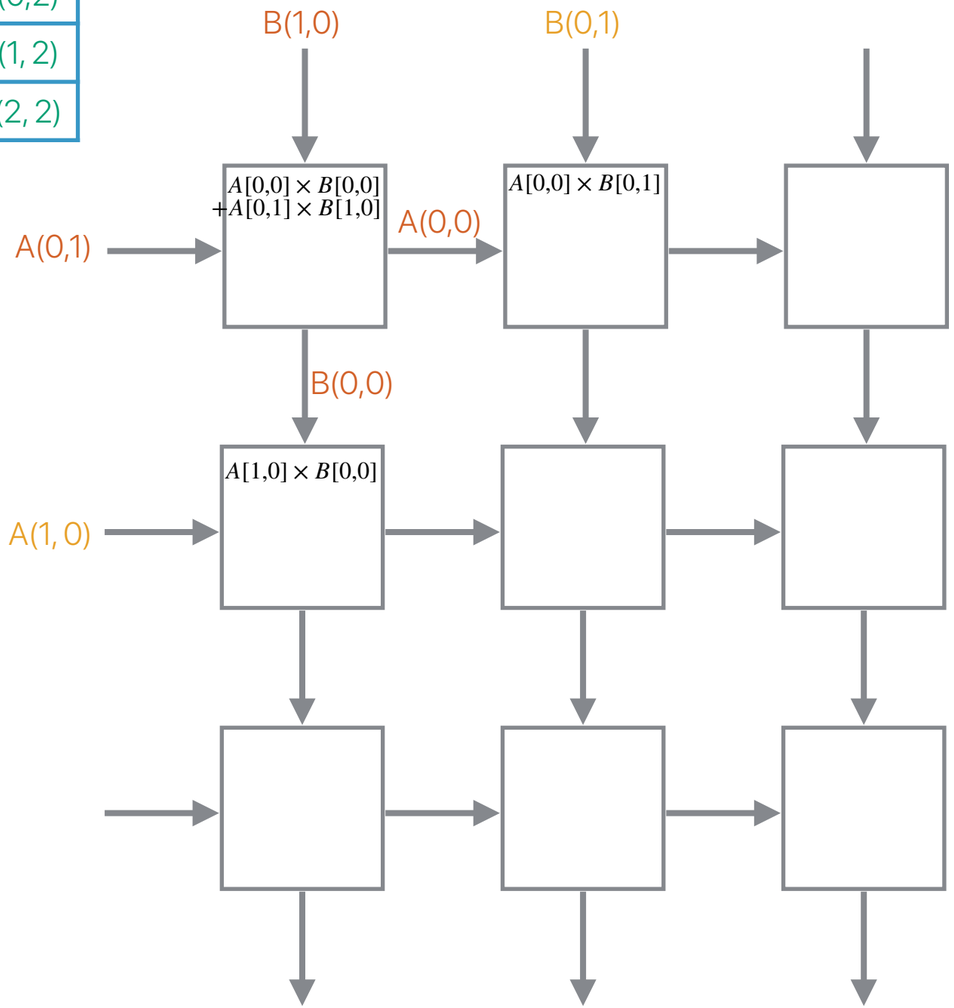
H.T. Kung, "Why systolic architectures?," in IEEE Computer, vol. 15, no. 1, pp. 37-46, Jan. 1982, doi: 10.1109/MC.1982.1653825.

Systolic Arrays used by AI/ML Accelerators

#2

B

(0,0)	(0,1)	(0,2)
(1, 0)	(1, 1)	(1, 2)
(2, 0)	(2, 1)	(2, 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.

Systolic Arrays used by AI/ML Accelerators

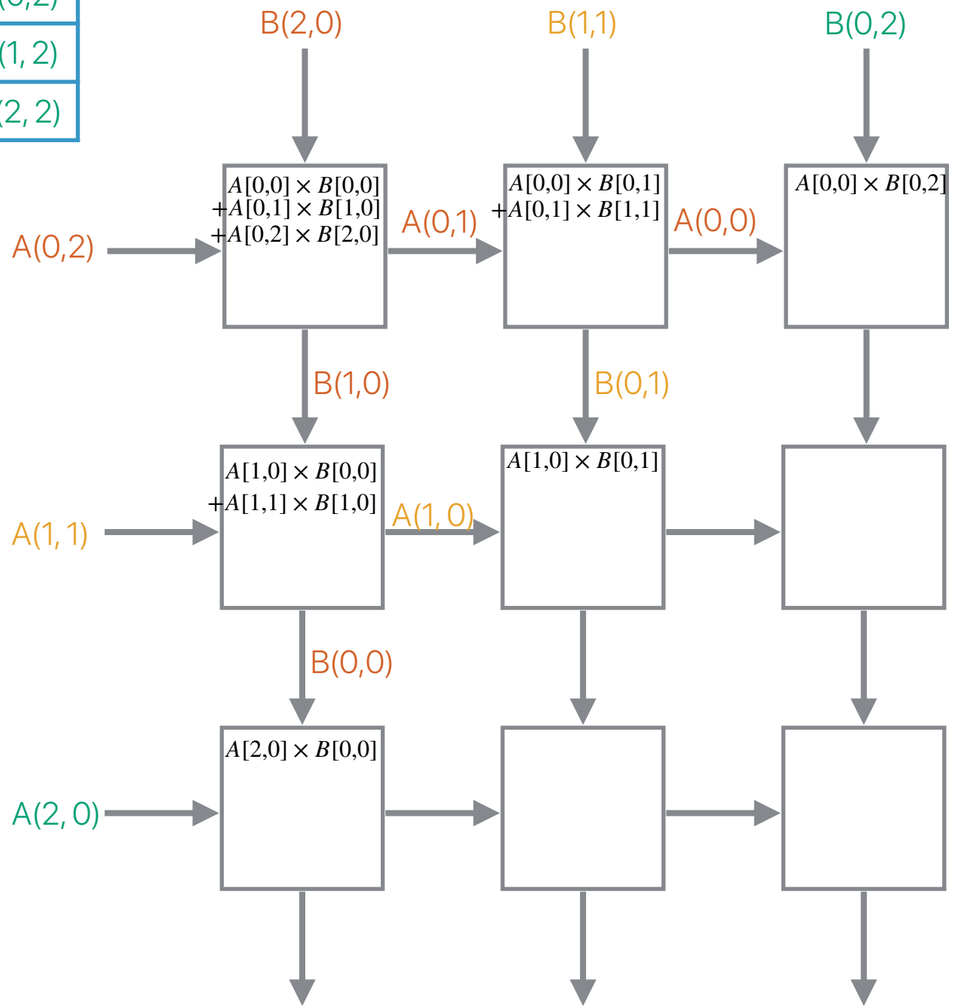
#3

B

(0,0)	(0,1)	(0,2)
(1, 0)	(1, 1)	(1, 2)
(2, 0)	(2, 1)	(2, 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.

Systolic Arrays used by AI/ML Accelerators

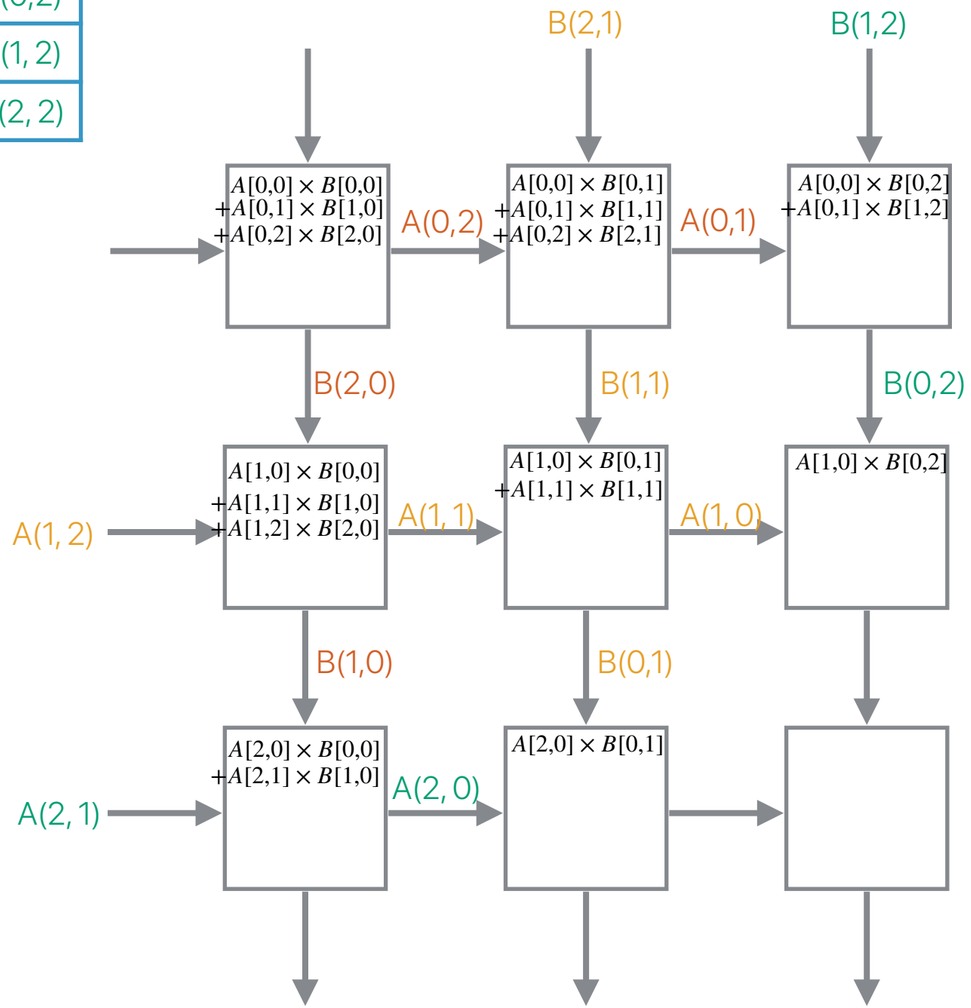
#4

B

(0,0)	(0,1)	(0,2)
(1, 0)	(1, 1)	(1, 2)
(2, 0)	(2, 1)	(2, 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.

Systolic Arrays used by AI/ML Accelerators

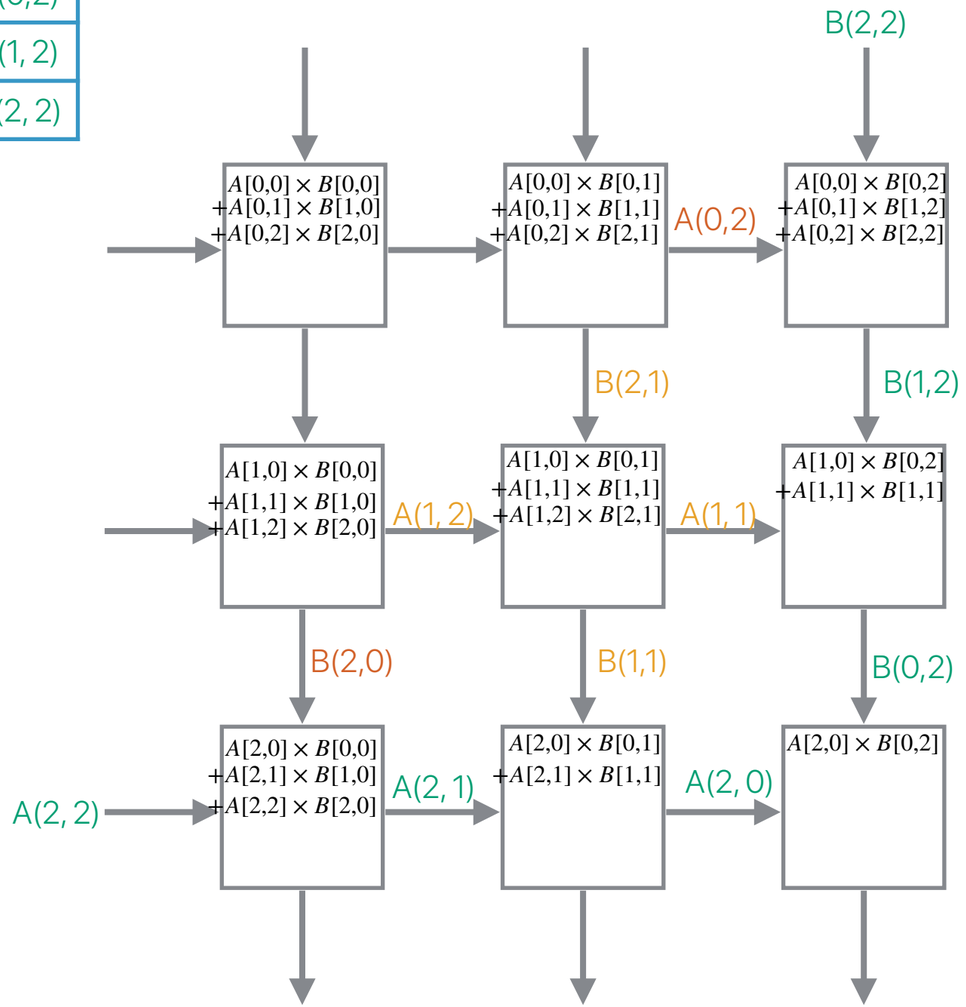
#5

B

(0,0)	(0,1)	(0,2)
(1, 0)	(1, 1)	(1, 2)
(2, 0)	(2, 1)	(2, 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.

Systolic Arrays used by AI/ML Accelerators

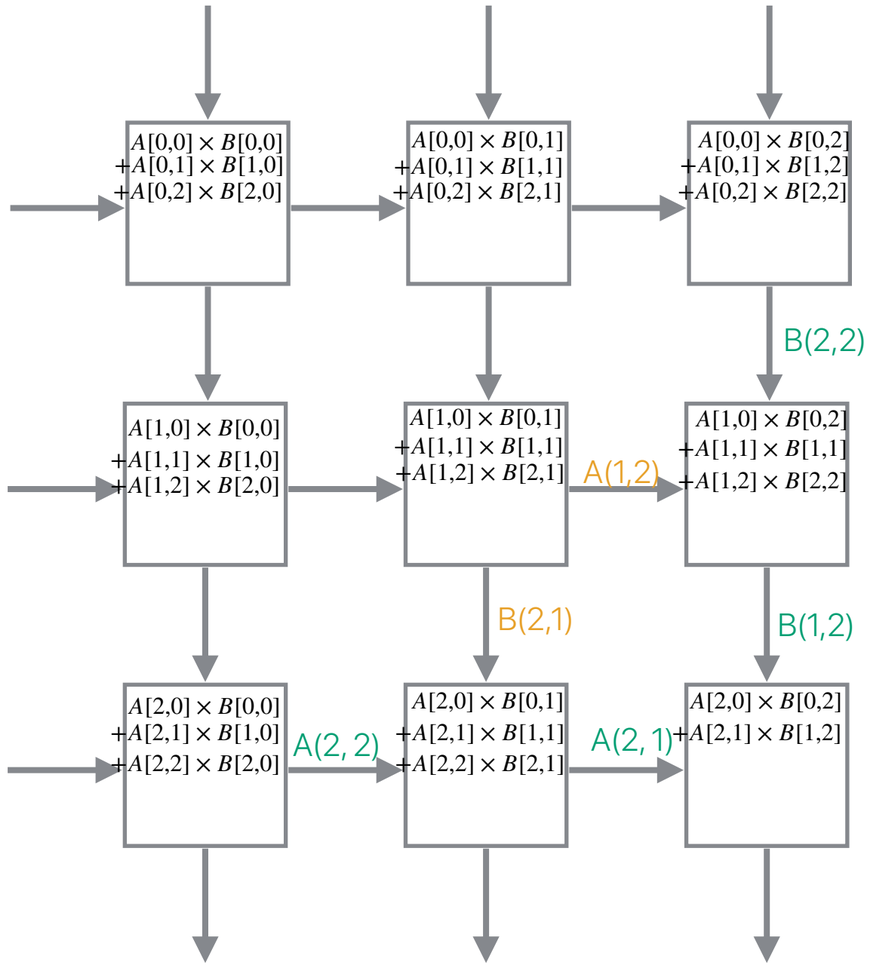
#6

B

(0,0)	(0,1)	(0,2)
(1, 0)	(1, 1)	(1, 2)
(2, 0)	(2, 1)	(2, 2)

A

(0,0)	(0,1)	(0,2)
(1, 0)	(1, 1)	(1, 2)
(2, 0)	(2, 1)	(2, 2)



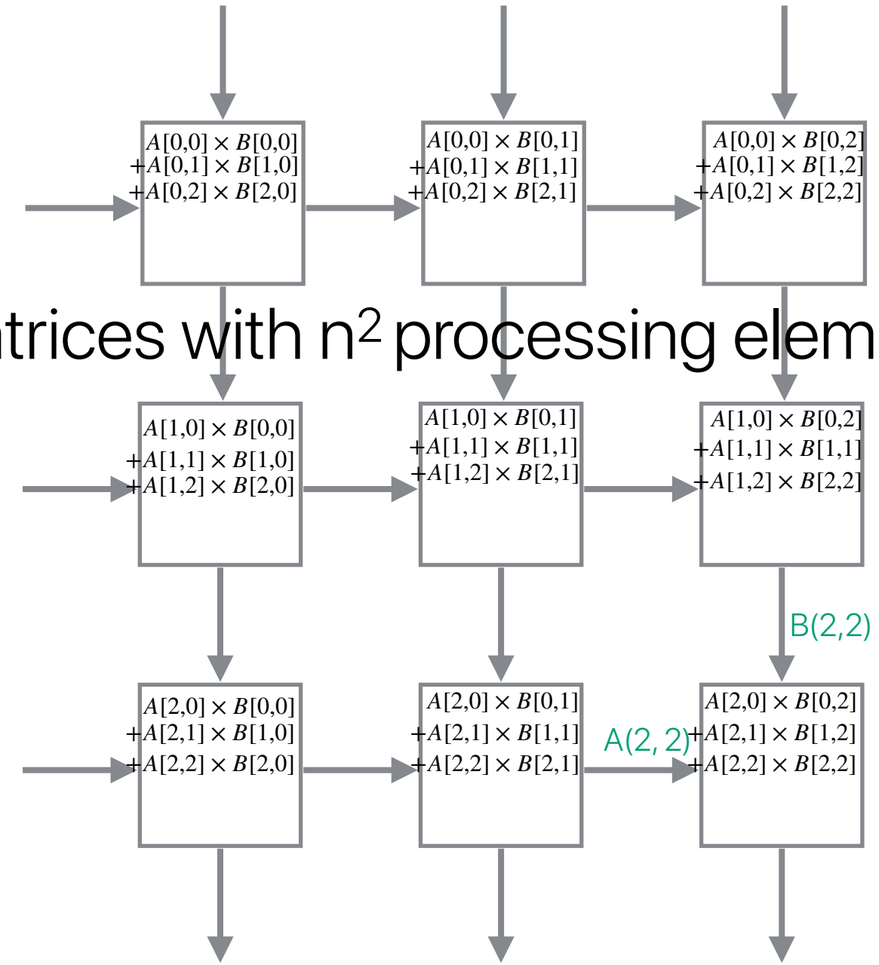
H.T. Kung, "Why systolic architectures?," in IEEE Computer, vol. 15, no. 1, pp. 37-46, Jan. 1982, doi: 10.1109/MC.1982.1653825.

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

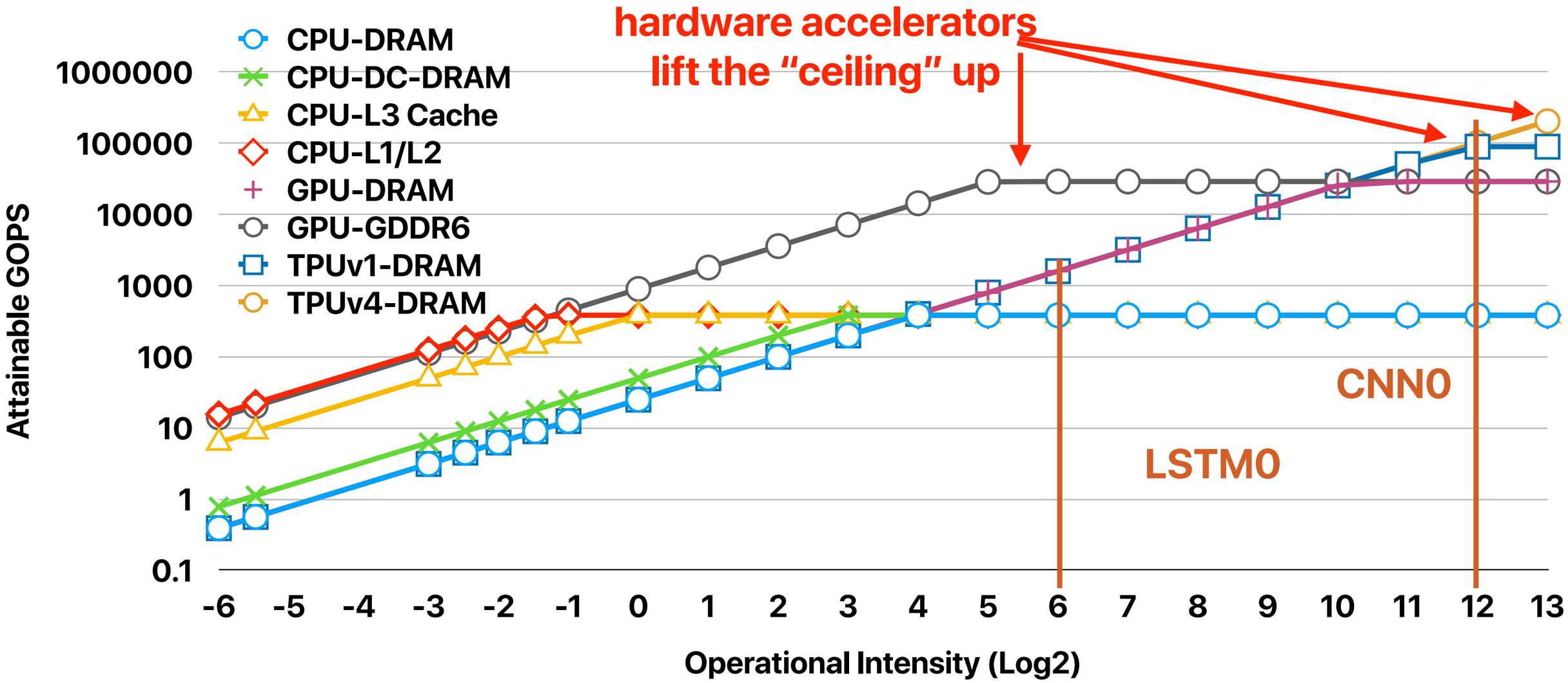
(0,0)	(0,1)	(0,2)
(1, 0)	(1, 1)	(1, 2)
(2, 0)	(2, 1)	(2, 2)

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

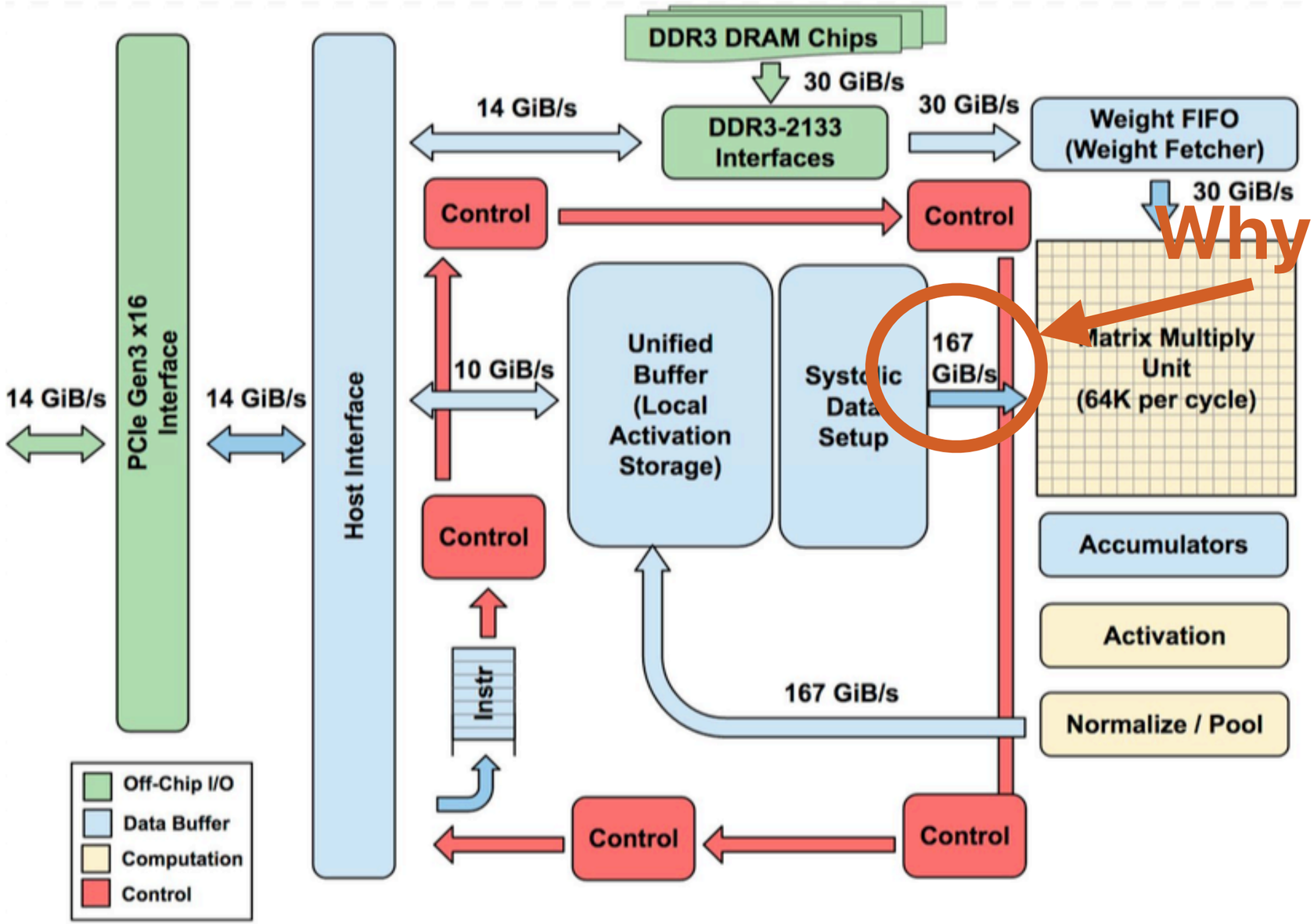
Why can accelerator help?

- 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

Placing TPUs in the roofline



TPU Block diagram



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