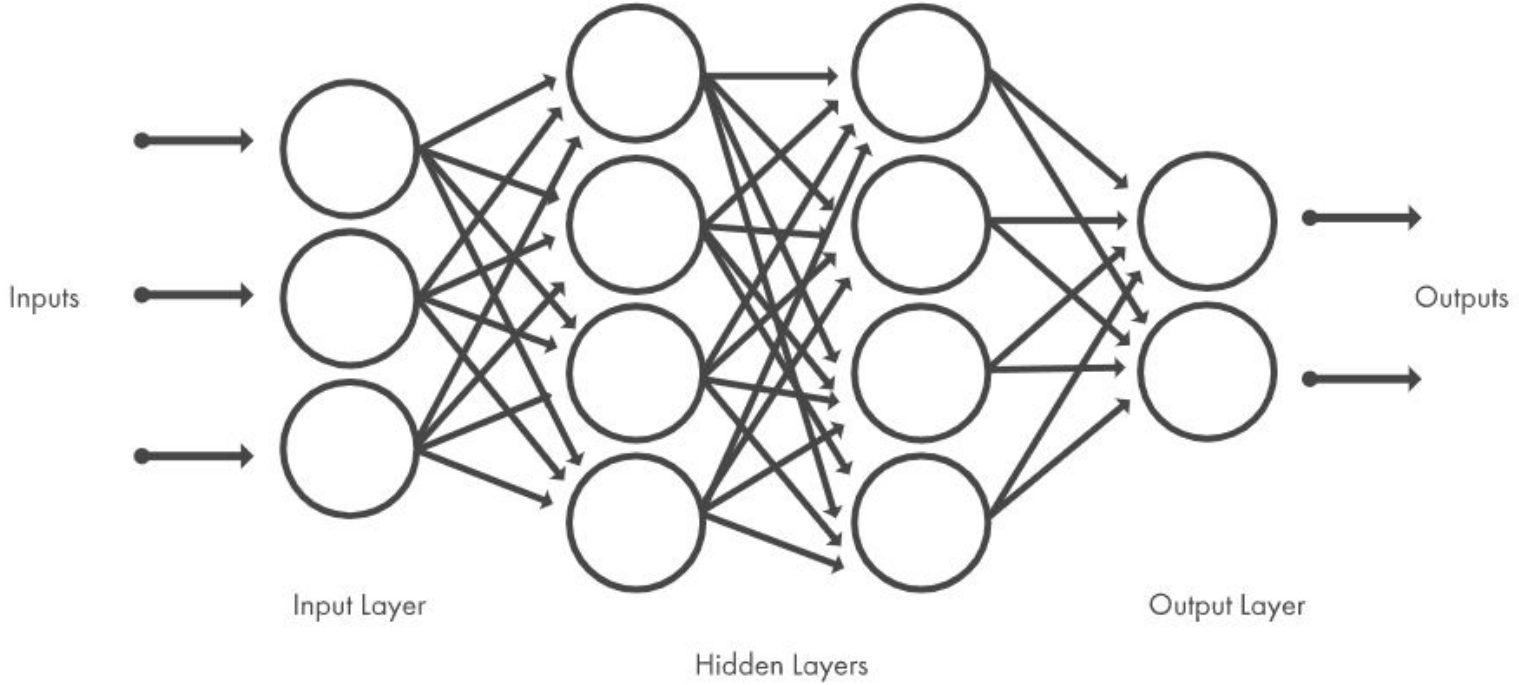
Nikoleta-Markela Iliakopoulou
Burak Ocalan
Pavlo Pastaryev
Selin Yildirim

Tensor Processor Unit (TPU)

Deep Neural Networks



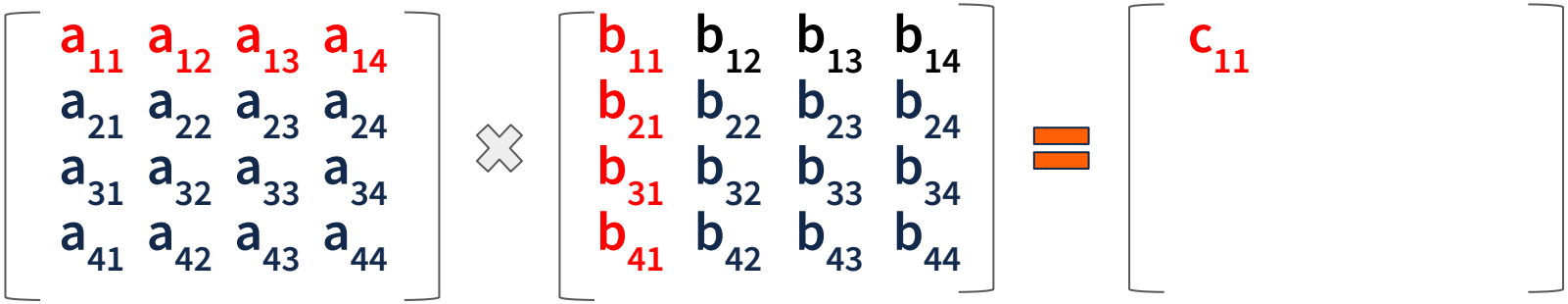
DNN Workloads

- Convolutions, matrix-matrix (M-M), matrix-vector (M-V), application of non-linearities, calculation of loss functions, weight updates …

- Convolution is the most expensive one & can be converted to matrix multiplication

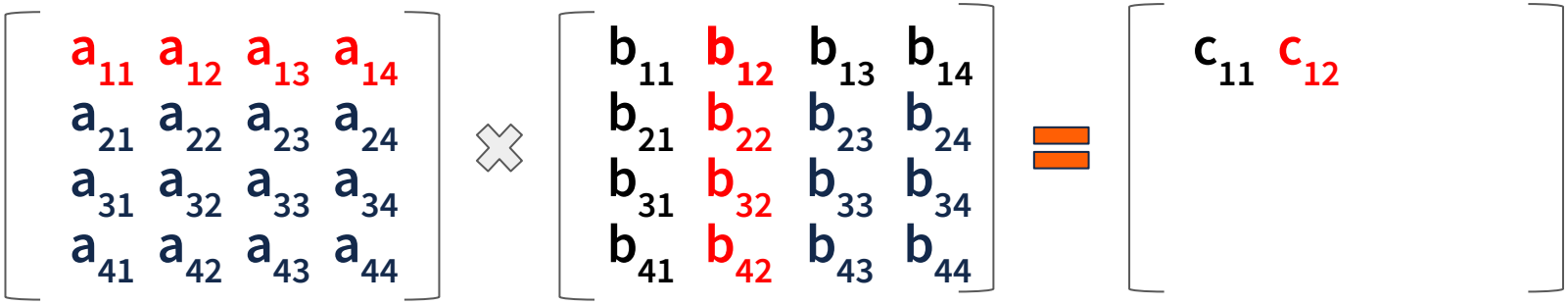
- **The idea:** Accelerate matrix multiplication

Matrix Multiplication



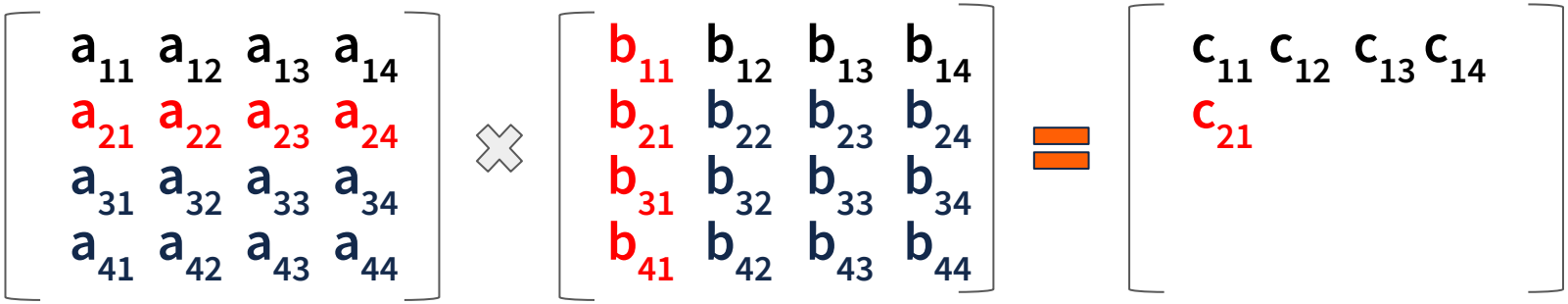

```
for (int i=1; i<=4; i++)
   for (int j=1; j<=4; j++) {
      sum = 0;
      for (int k=1; k<=4; k++)
         sum = sum + a[i][k] * b[k][j];
      c[i][j] = sum;
   }
```

Matrix Multiplication



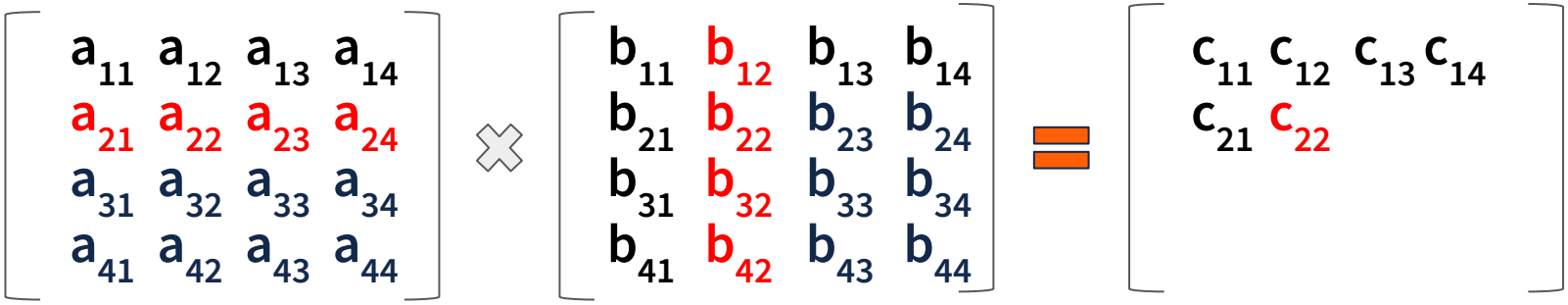

```
for (int i=1; i<=4; i++)
   for (int j=1; j<=4; j++) {
      sum = 0;
      for (int k=1; k<=4; k++)
         sum = sum + a[i][k] * b[k][j];
      c[i][j] = sum;
   }
```

Matrix Multiplication



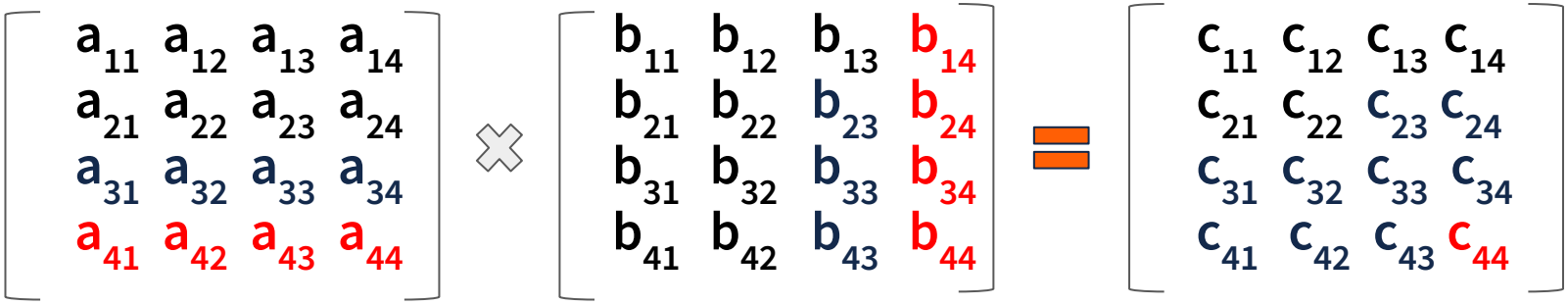

```
for (int i=1; i<=4; i++)
   for (int j=1; j<=4; j++) {
      sum = 0;
      for (int k=1; k<=4; k++)
         sum = sum + a[i][k] * b[k][j];
      c[i][j] = sum;
   }
```

Matrix Multiplication



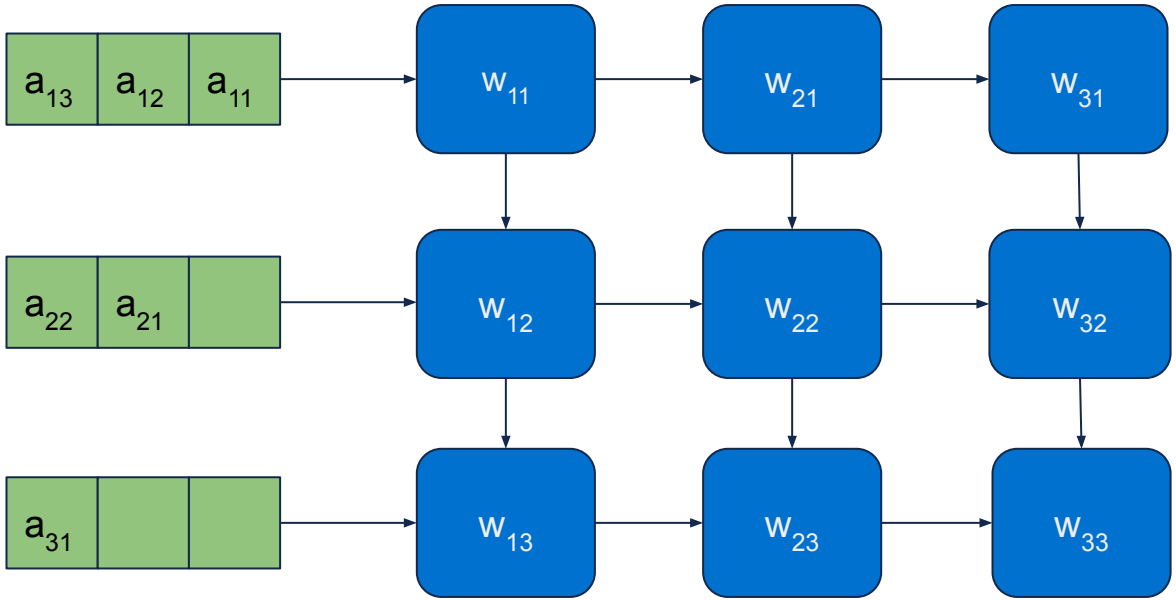

```
for (int i=1; i<=4; i++)
   for (int j=1; j<=4; j++) {
      sum = 0;
      for (int k=1; k<=4; k++)
         sum = sum + a[i][k] * b[k][j];
      c[i][j] = sum;
   }
```

Matrix Multiplication



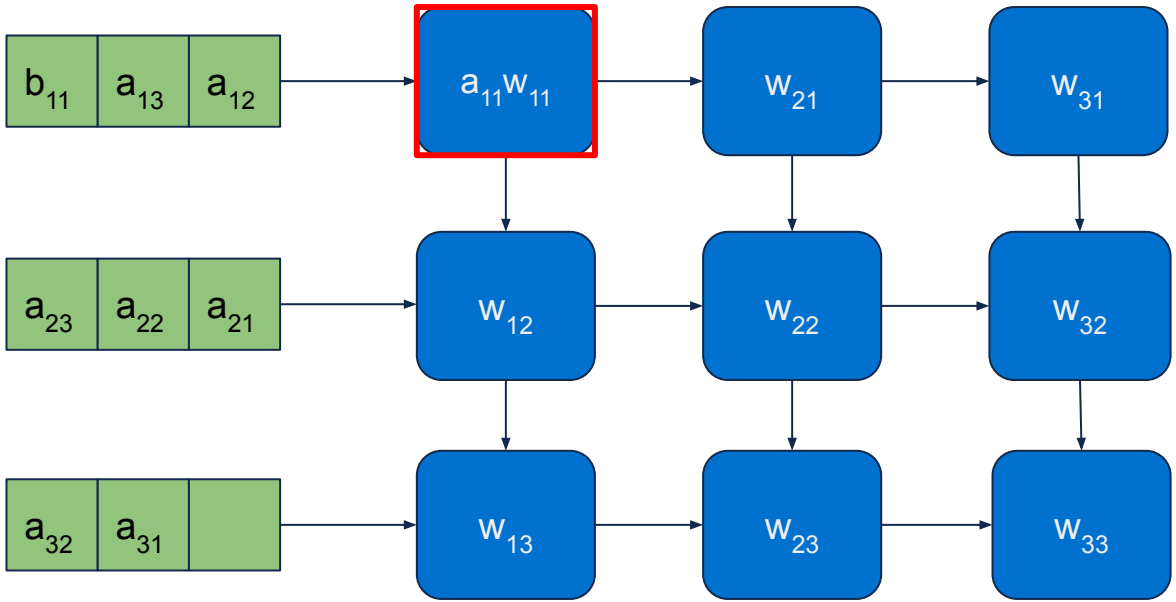

```
for (int i=1; i<=4; i++)
   for (int j=1; j<=4; j++) {
      sum = 0;
      for (int k=1; k<=4; k++)
         sum = sum + a[i][k] * b[k][j];
      c[i][j] = sum;
   }
```

Systolic Arrays



t = 0

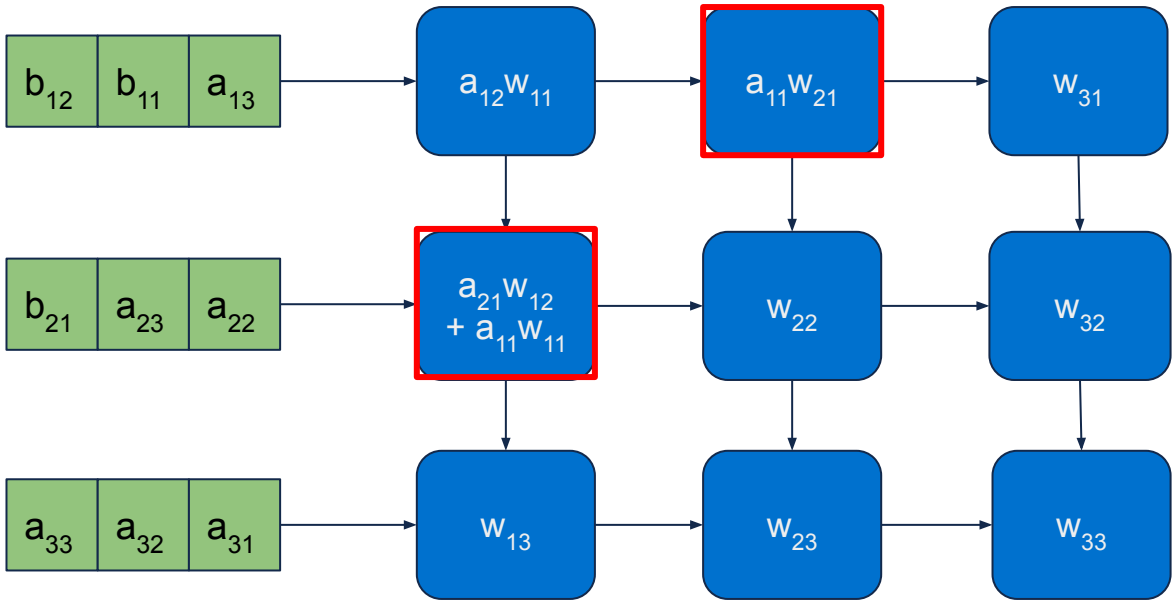
Systolic Arrays



t = 1

[W] * [A] = [X]

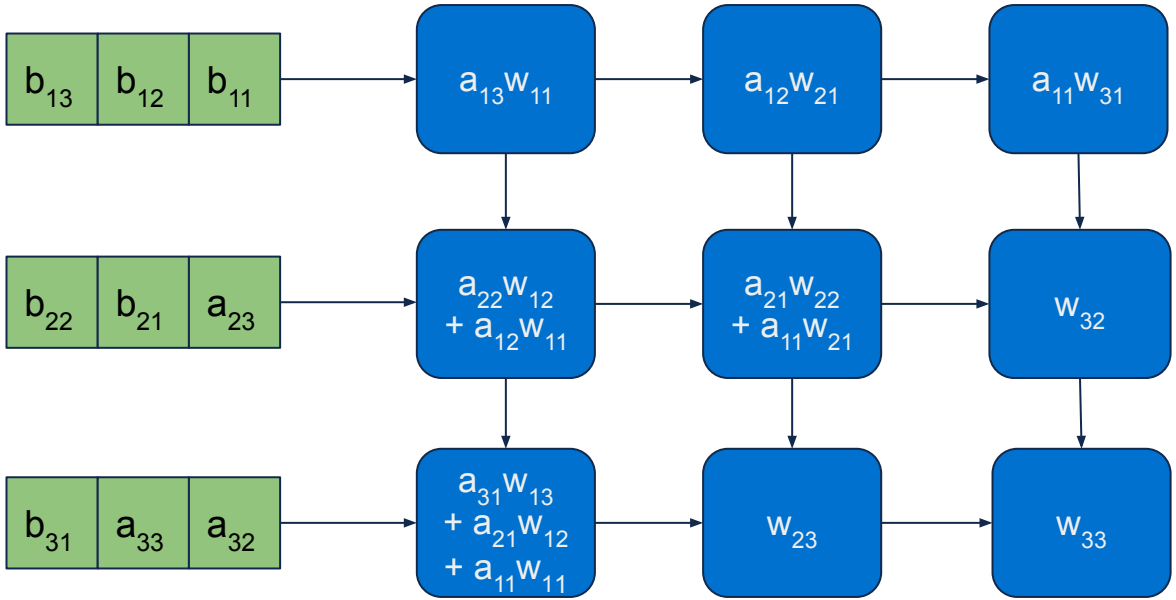
Systolic Arrays



t = 2

[W] * [A] = [X]

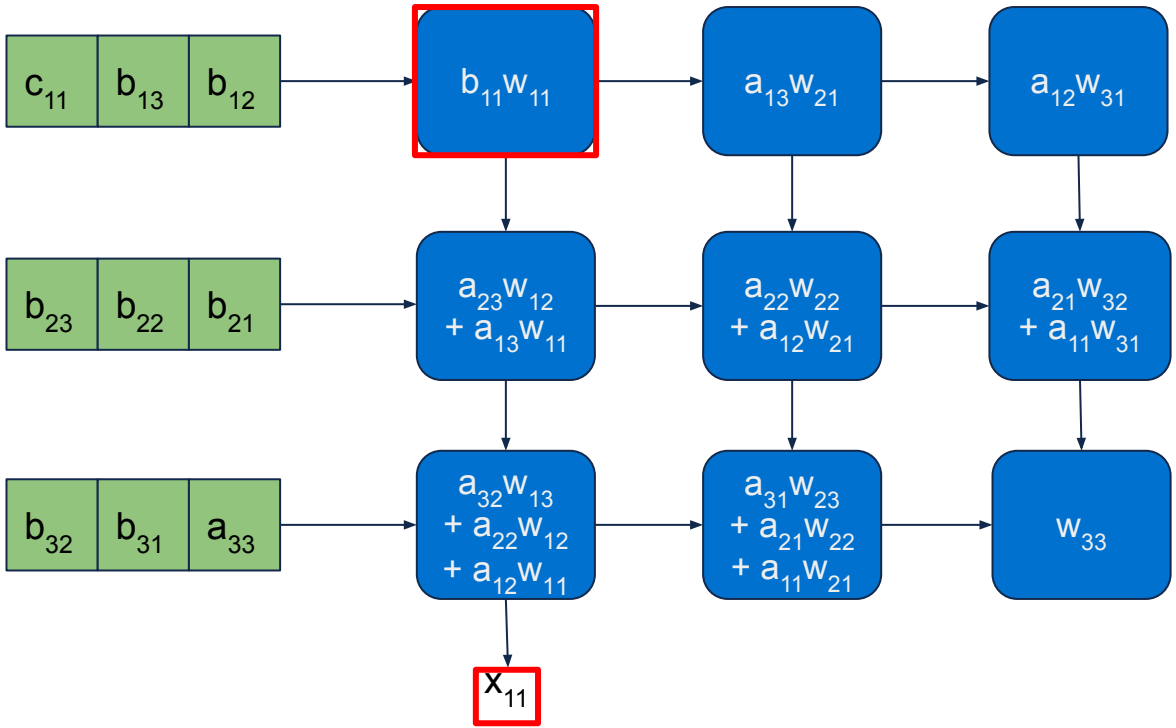
Systolic Arrays



t = 3

[W] * [A] = [X]

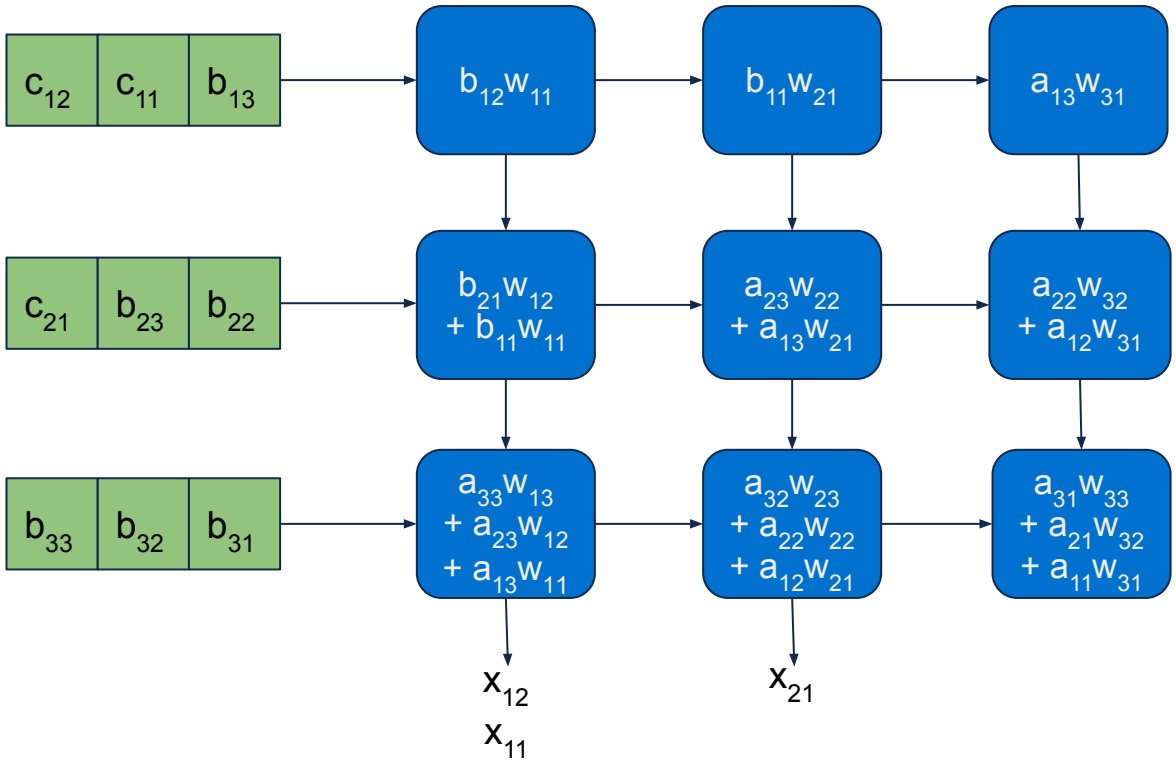
Systolic Arrays



t = 4

[W] * [A] = [X]
[W] * [B] = [Y]

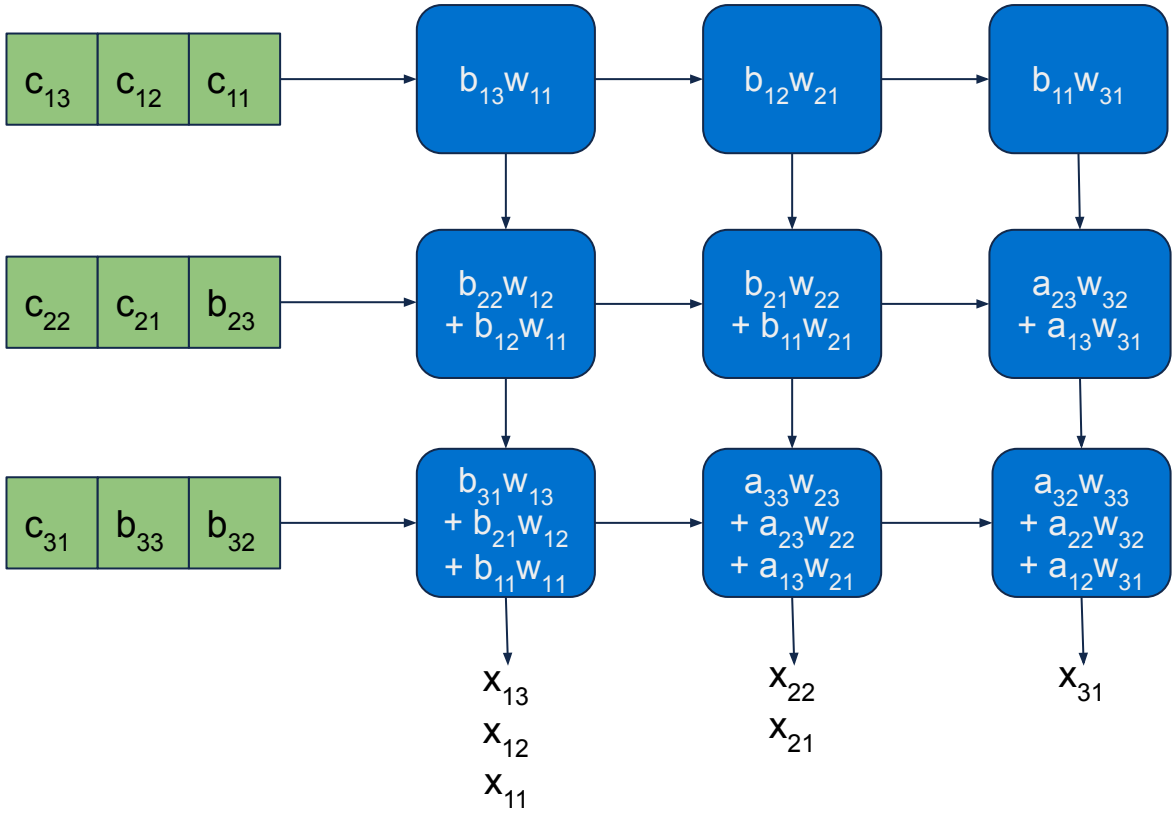
Systolic Arrays



t = 5

[W] * [A] = [X]
[W] * [B] = [Y]

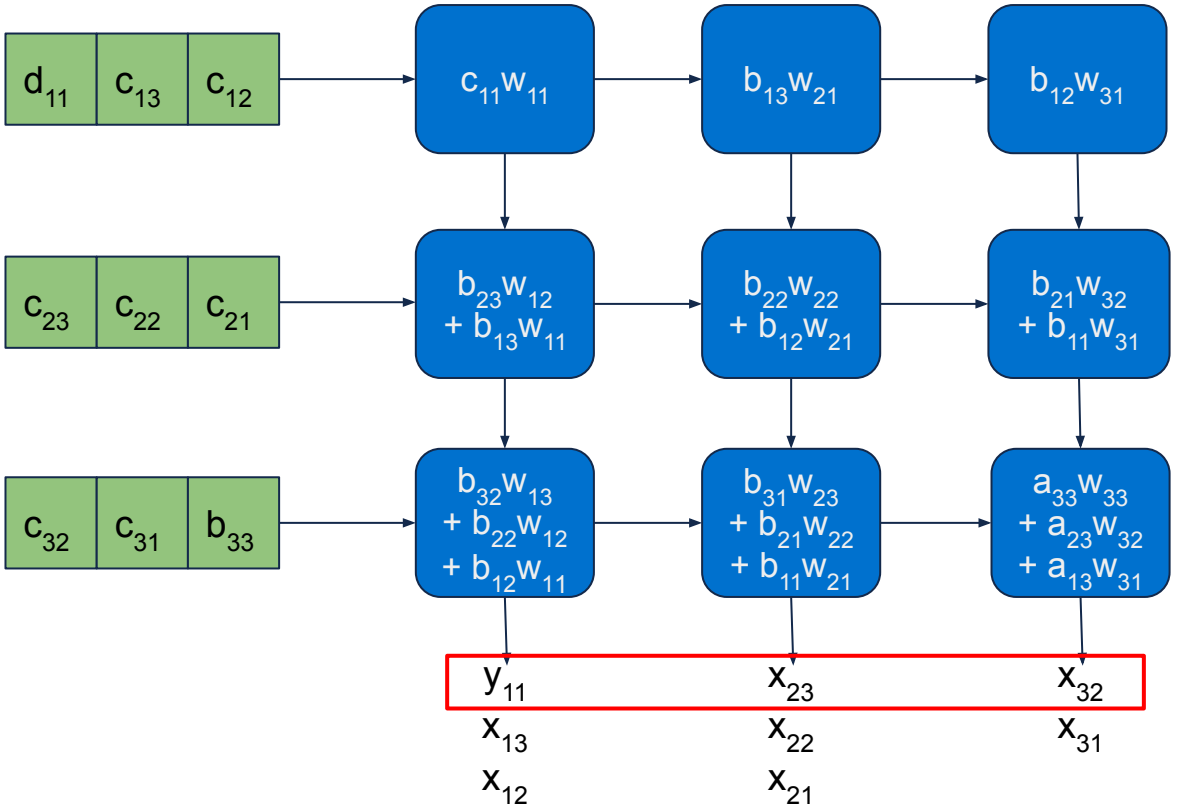
Systolic Arrays



t = 6

[W] * [A] = [X]
[W] * [B] = [Y]

Systolic Arrays



t = 7

[W] * [A] = [X]
[W] * [B] = [Y]
[W] * [C] = [Z]
…

Tensor Processing Unit (TPU) First Generation

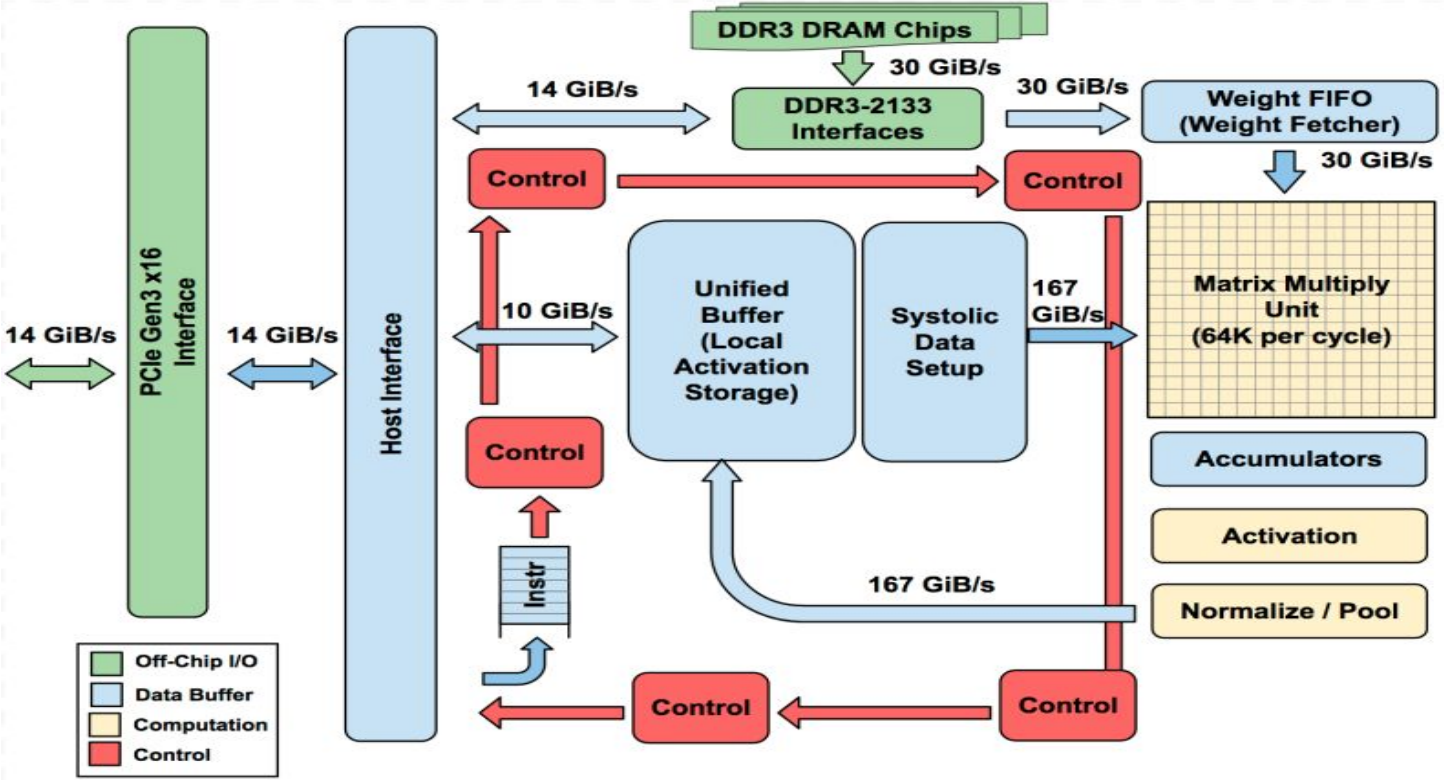
Tensor Processing Unit (TPU) - Goals

- Custom ASIC developed by Google for Neural Networks Acceleration
- Improve cost-performance over GPUs
- Run whole inference models in the TPU
- Flexible to meet NNs needs of 2015 and beyond

Tensor Processing Unit (TPU) - Design Choices

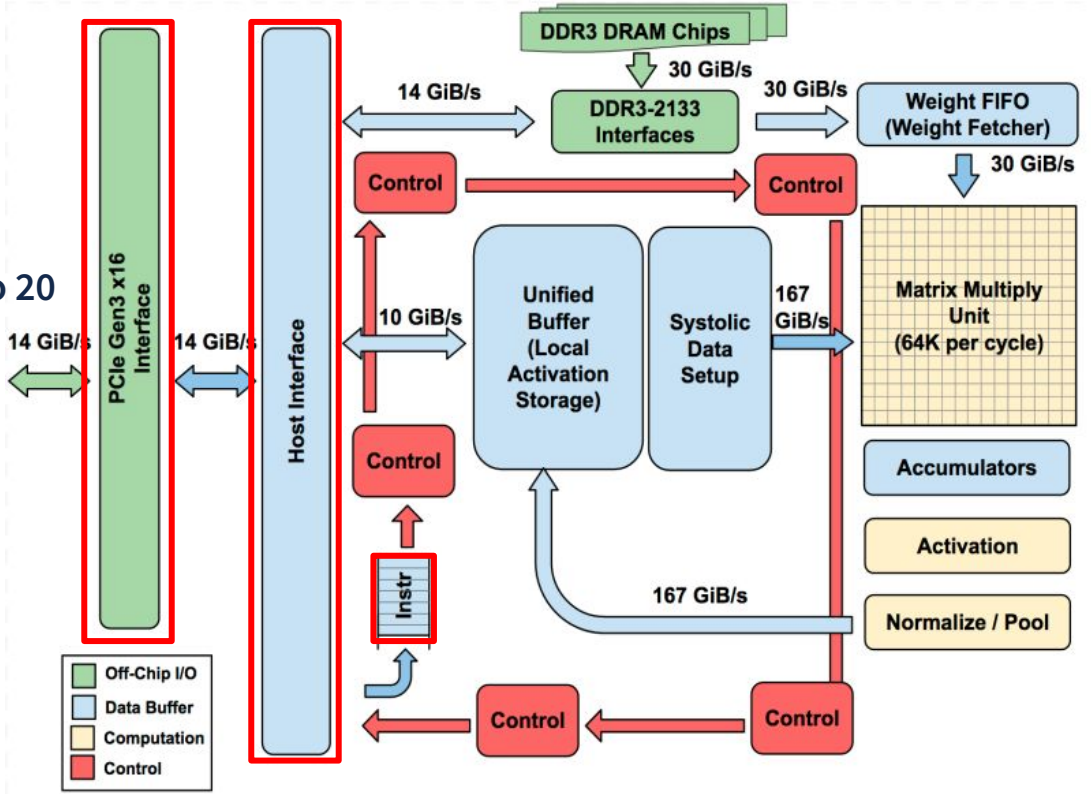
- Coprocessor on the PCIe I/O bus
- CPU sends TPU instructions for it to execute
- Closer in spirit to an FPU coprocessor than to a GPU
- Quantization - use simple 8-bit integer instead of floats reducing power consumption

TPU v1 Architecture



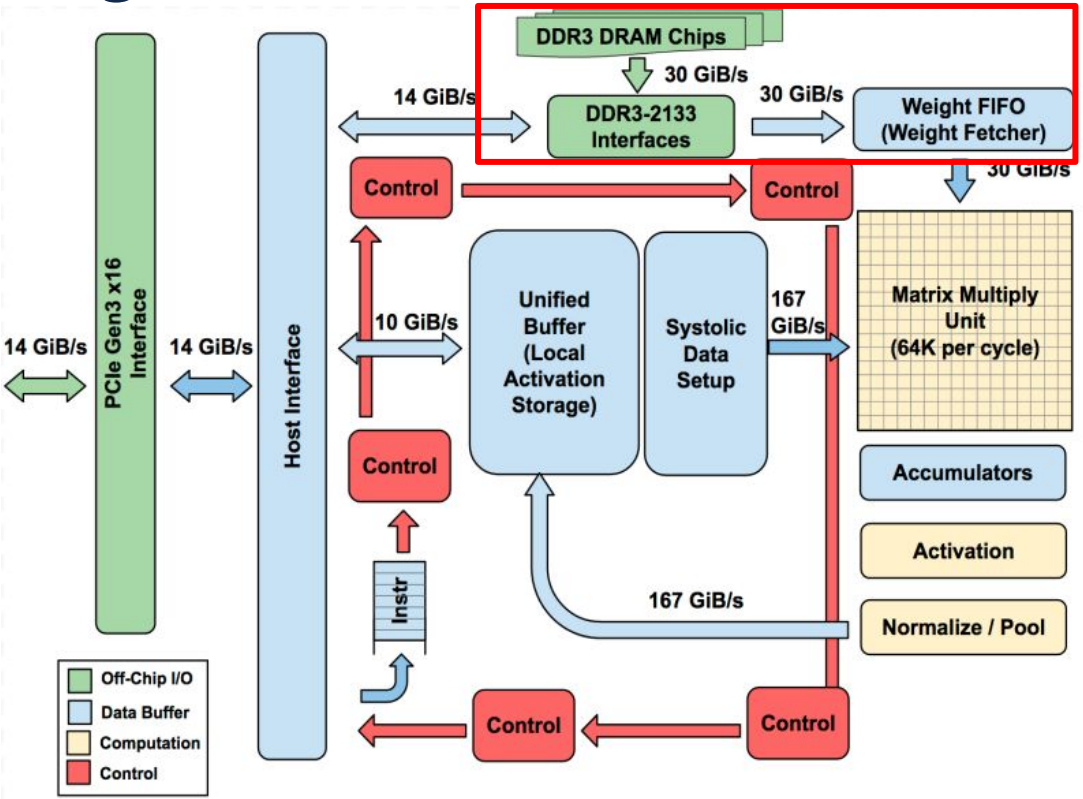
PCIe interface for TPU instructions

- TPU instructions are sent from the host over the PCIe into an instruction buffer
- CISC Instructions with average CPI 10 to 20



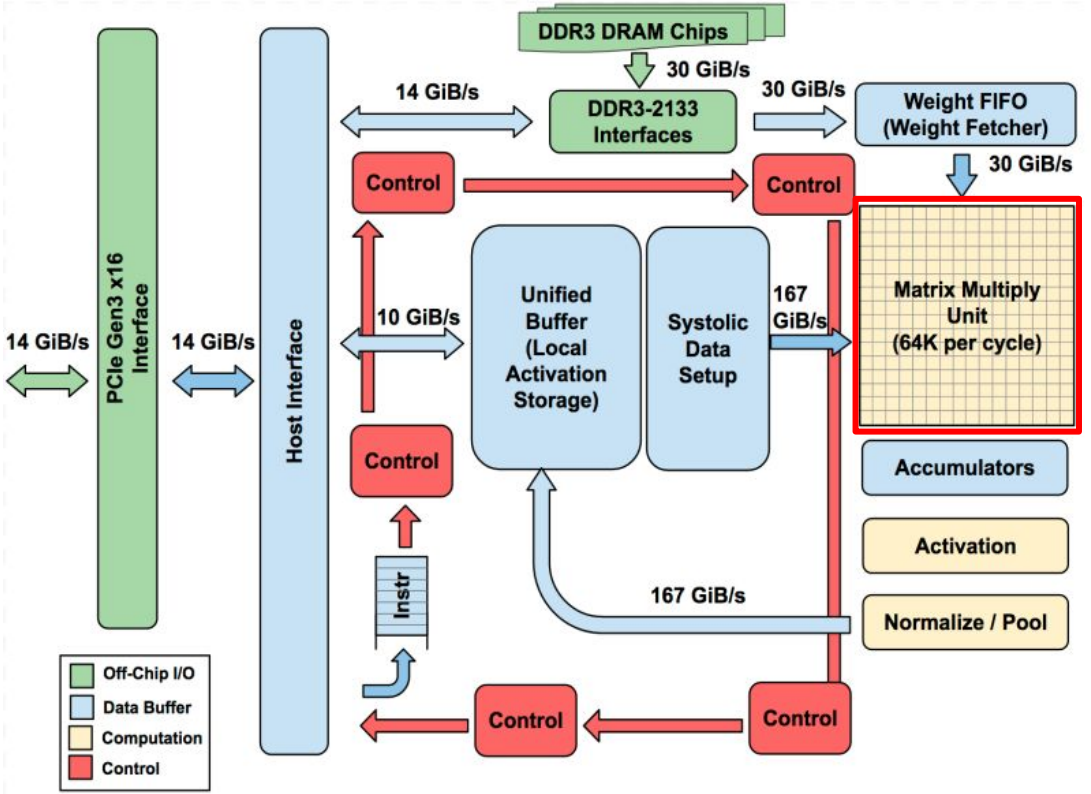
Weight Memory & Weight FIFO

- Off-chip 8 GiB DRAM
- Four tiles deep on chip weight FIFO
- Same weights reused across multiple inputs



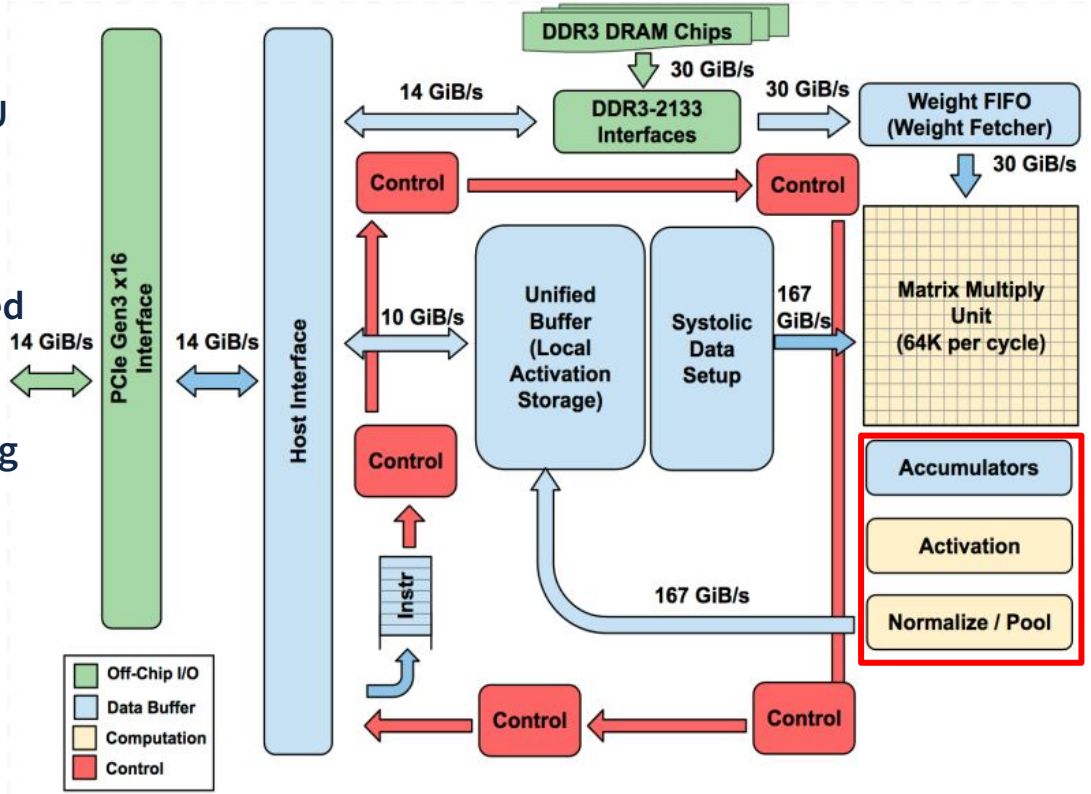
Matrix Multiply Unit

- 256x256 MACs
- 2 tiles of 64KiB weights
  - One active, the other used for double buffering
- Must keep the MMU busy
  - 4 stage pipeline for CISC instructions
- Implemented as a Systolic Array
- Designed for dense matrices



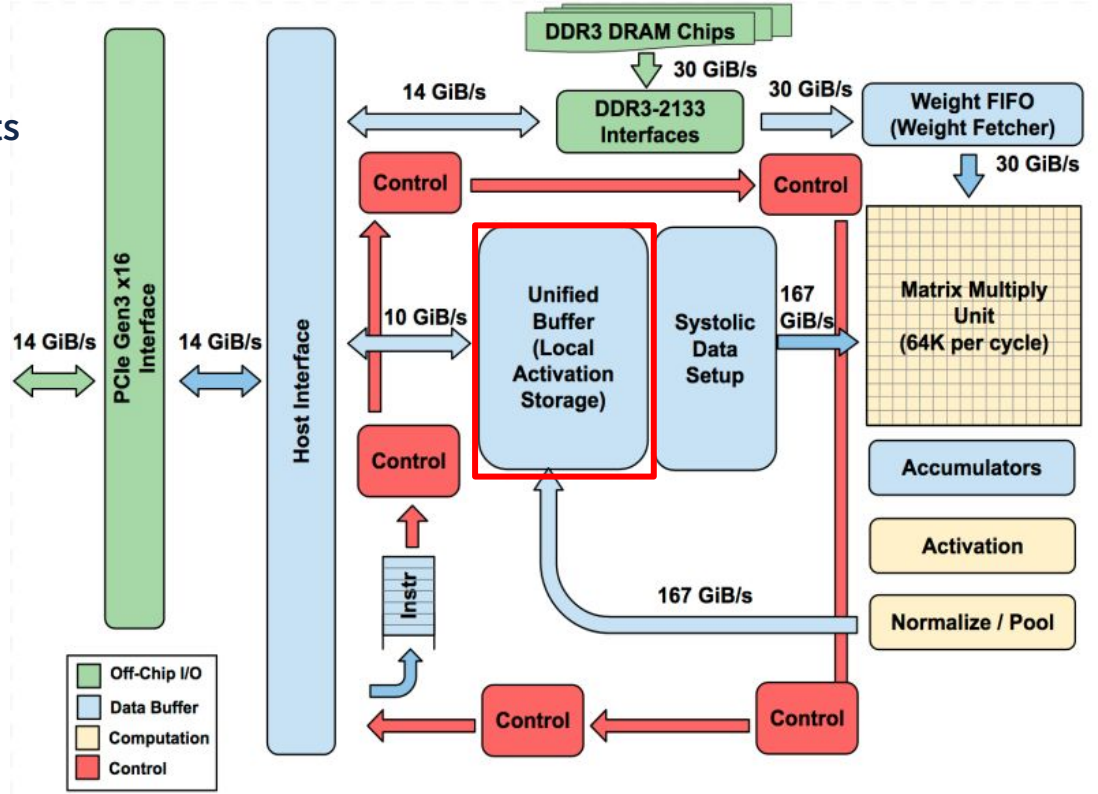
Accumulators

- Collect the 16-bit products of the MMU
- 4MiB: 4096, 256-element, 32-bit
  - Why 4K vectors?
    - Peak performance observed at 1350
    - Round up to 2048
    - Double for double buffering
- Calculate partial sums

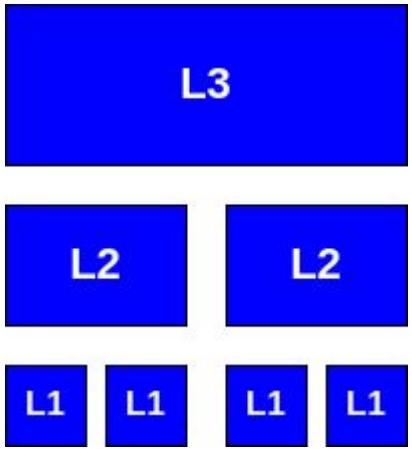


Unified Buffer

- 24 MiB on-chip for intermediate results
- Inputs to the MMU



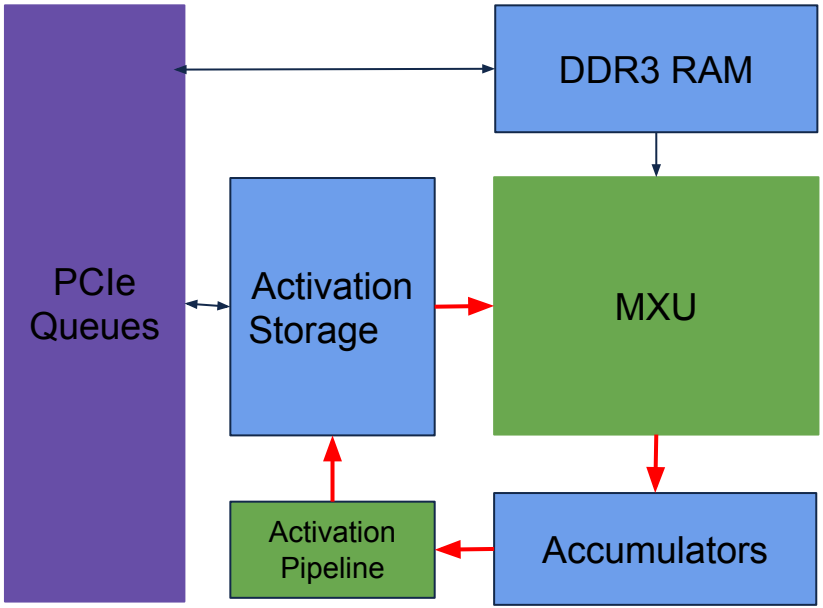
TPU Memory Management

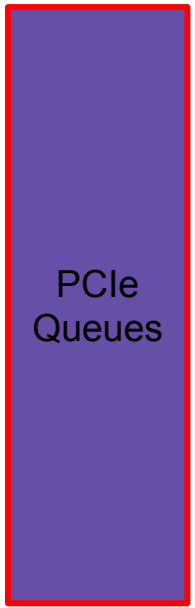


CPU implicit memory management

TPU explicit memory management

Designing Next Generations of TPU





?

TPU v1

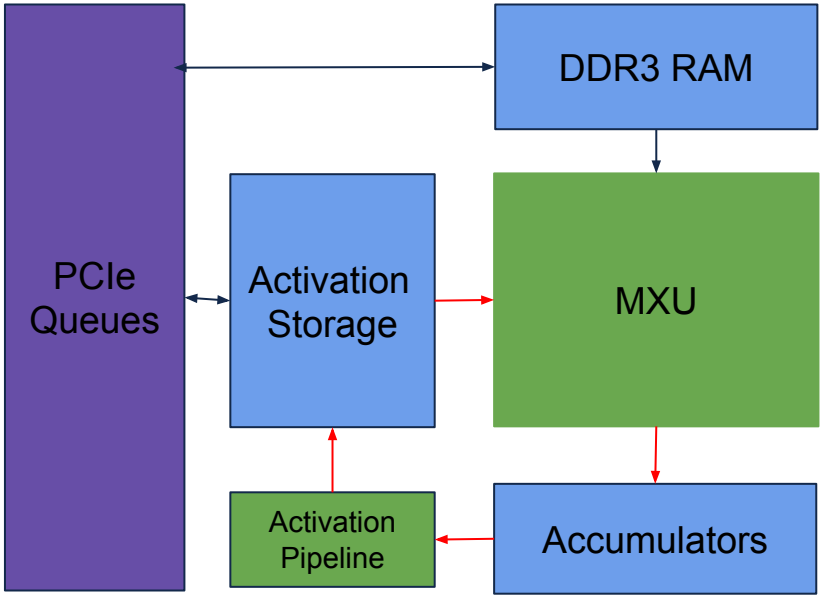
TPU v2/3

Problem : Training

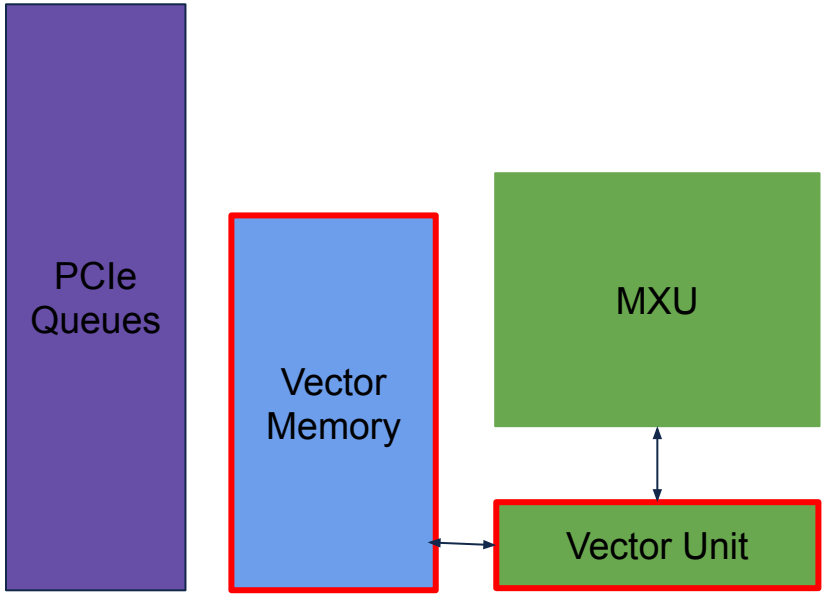
- Computation
- Memory
- Programmability
- Quantization

Solution: Vector Units & Vector Memory

- Read-only weights → Write (*Computation*)
- 2D 128x8 vector registers (*Storage*)
- Replaces the fixed-function datapath (*Programmability*)
- Bfloat16 instead of int8 (*No quantization*)



TPU v1



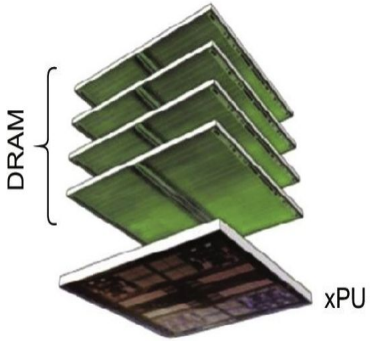
TPU v2/3

Problem : Memory Bandwidth

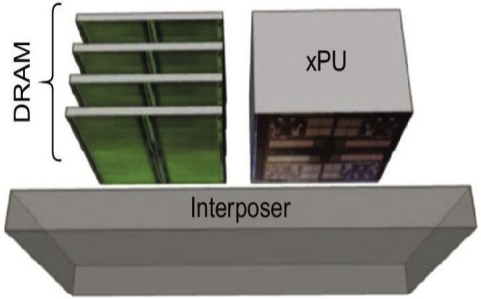
- TPUv1 was memory bound
- TPUv2 required even more memory for training

Solution: High Bandwidth Memory (HBM)

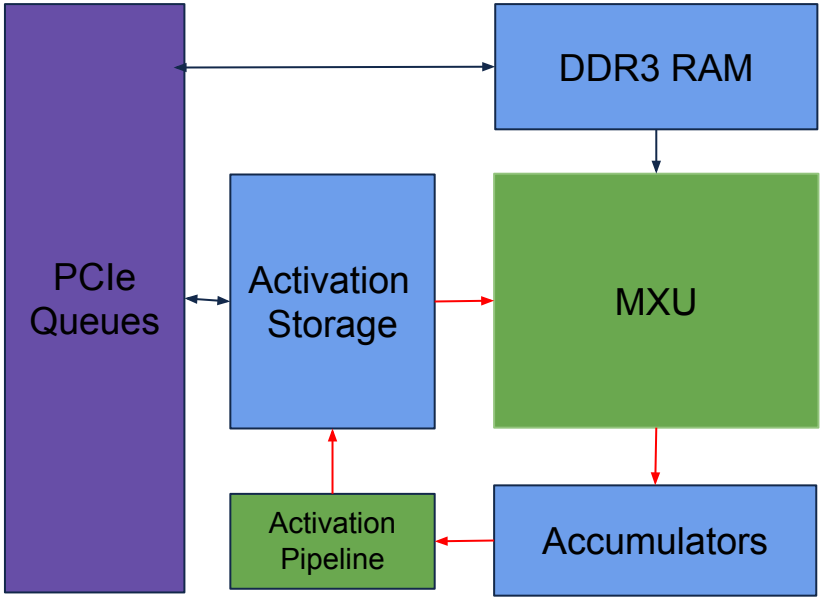
- HMB DRAM (20x bw than v1)
- 4 short stacks of DRAM chips



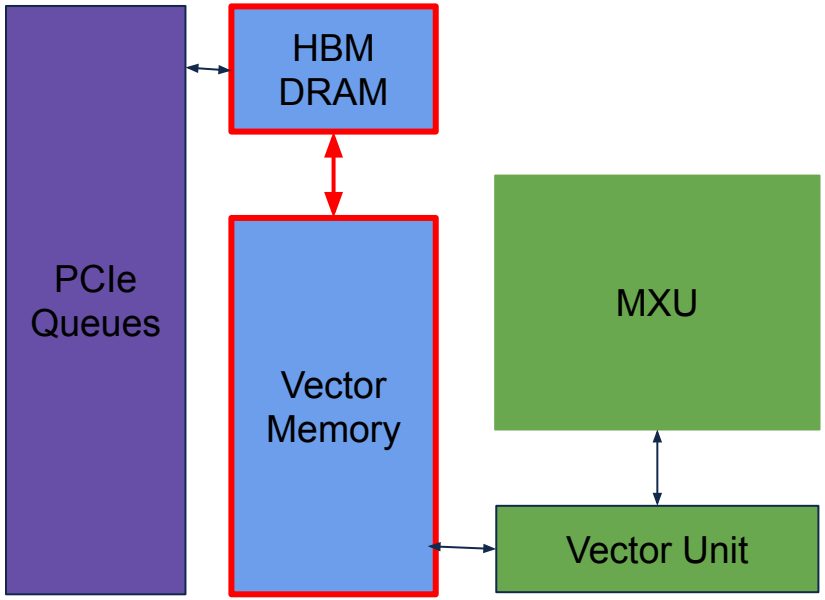
Vertical stacking (3D)



Interposer stacking (2.5D)



TPU v1



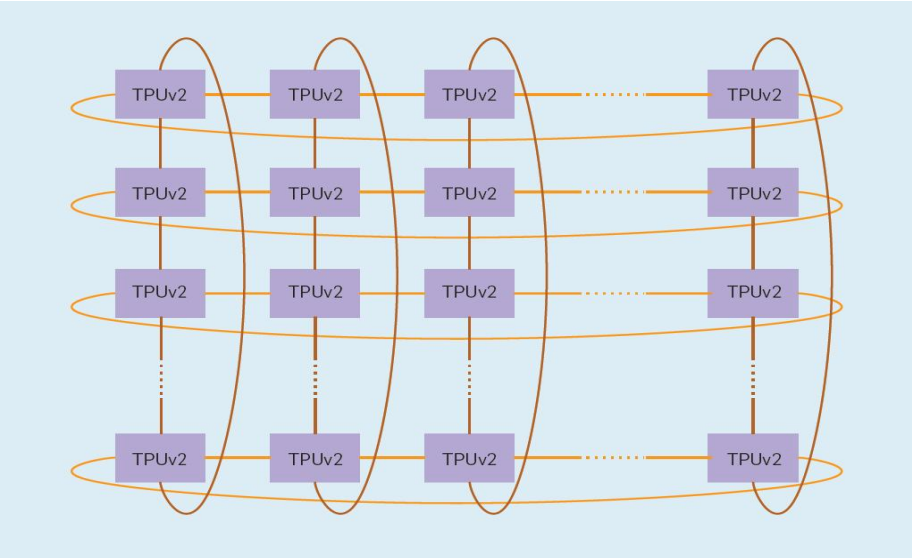
TPU v2/3

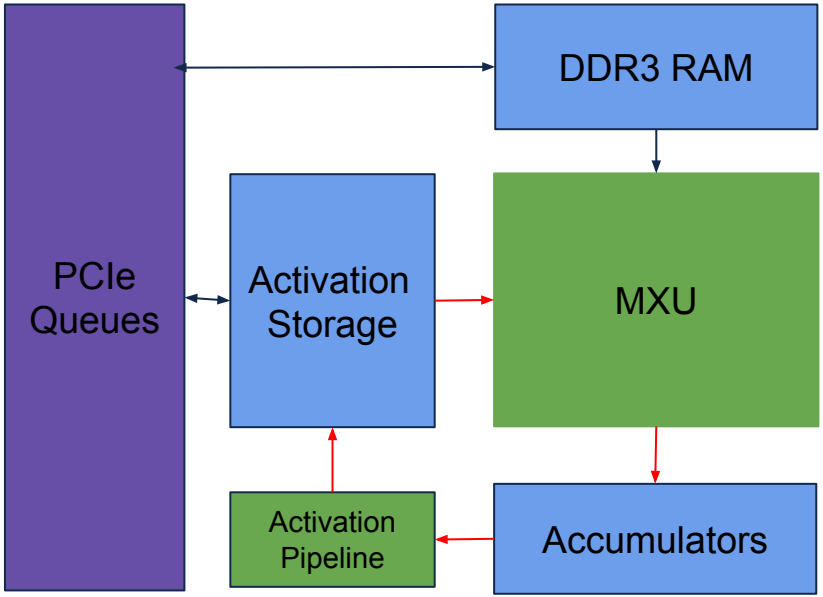
Problem : ML on a Supercomputer Scale

- Chip-to-Chip

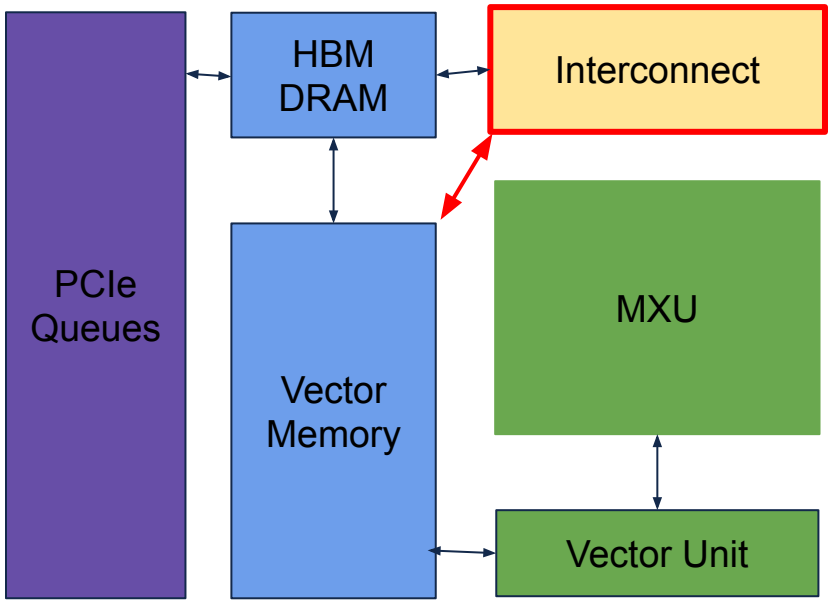
  Interconnect fabric

- 256 chips in 2D-Torus





TPU v1

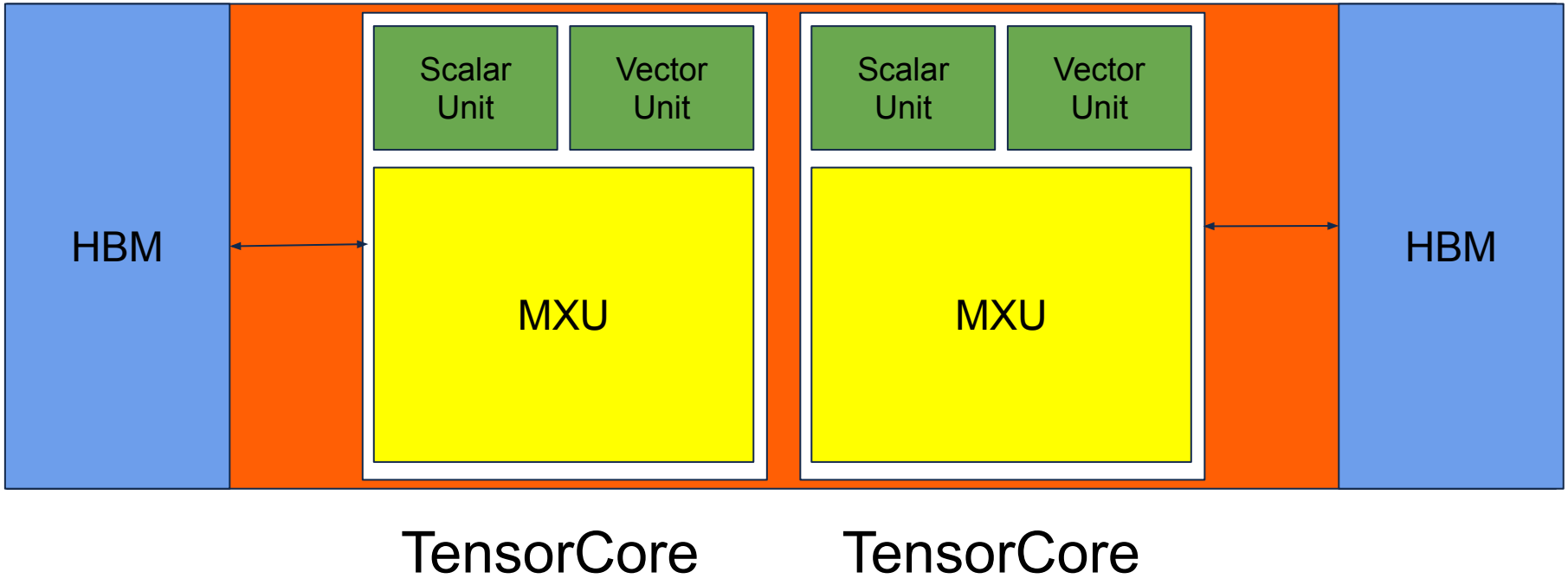


TPU v2/3

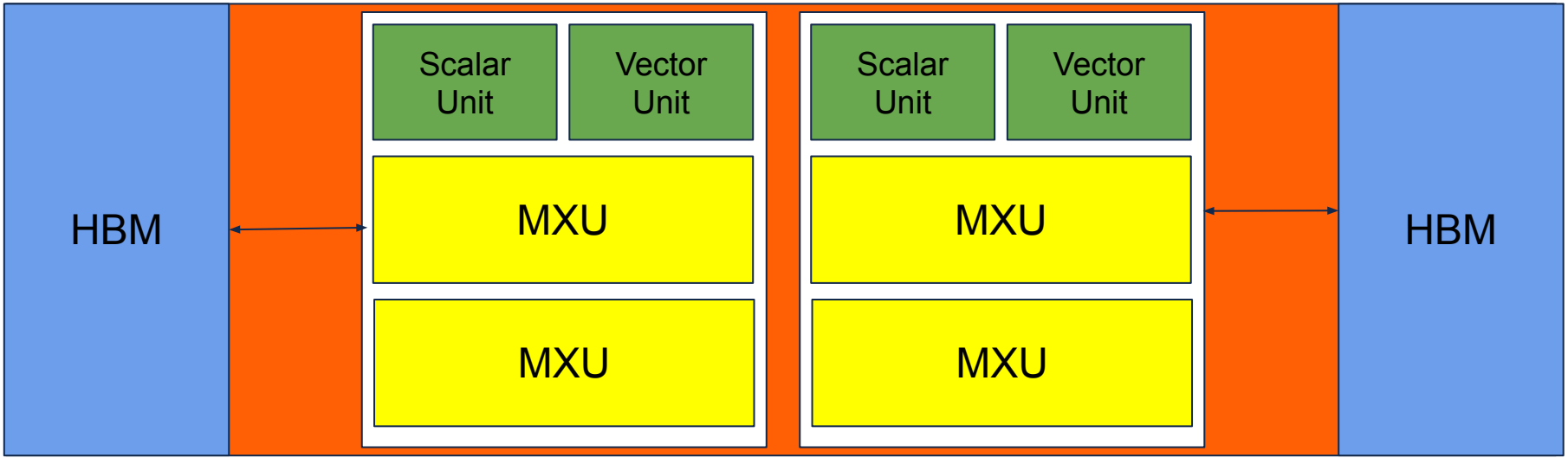
Other Features of TPUv2/3

- Instructions no longer delivered from the CPU
- 322-bit VLIW instructions
- Scalar memory

TPU v2 Board


HBM
Scalar Unit
Vector Unit
MXU
Scalar Unit
Vector Unit
MXU
HBM
TensorCore
TensorCore

TPU v3 Board



HBM capacity/bandwidth: 32GiB, 900 GB/s
Measured min/mean/max power: 123/220/262 W
Peak compute per chip: 123 teraflops
Peak compute per pod (1024 chips): 126 petaflops
Bisection bandwidth per pod (1024 chips): 6.4 TB/s

TPU v4

Added CMEM (Common Memory)

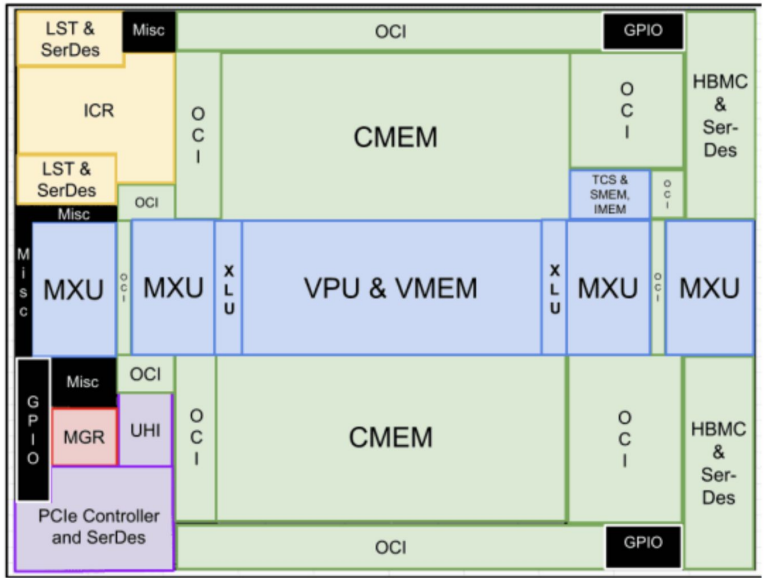
- Small (128 MB) scratchpad SRAM
- Reduces the number to the slowest and least energy efficient memory

Added OCI (On-Chip Interconnect)

- Connects all components of the chip together
- Allows to better control which memory is used, how data is transferred

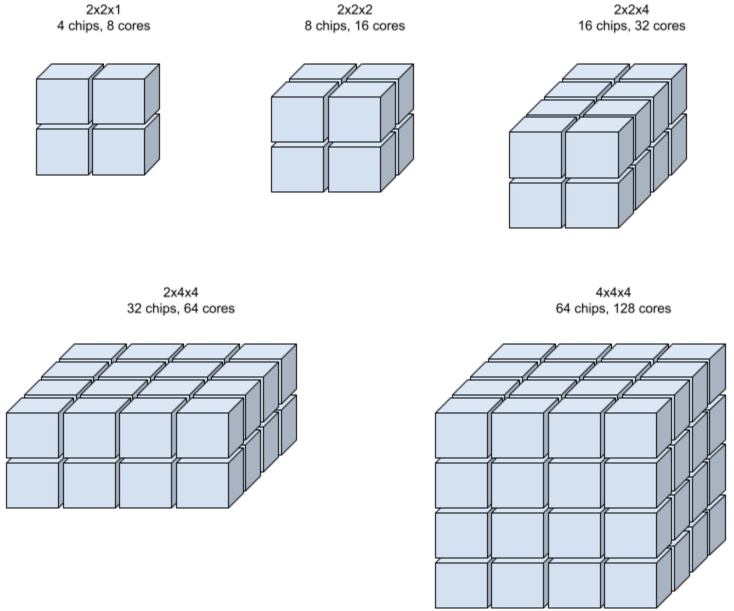
4D Tensor DMA (Dynamic Memory Access)

- The chip contains tensor DMA engines that decode and execute DMA instructions
- Offloads work from the main TensorCore

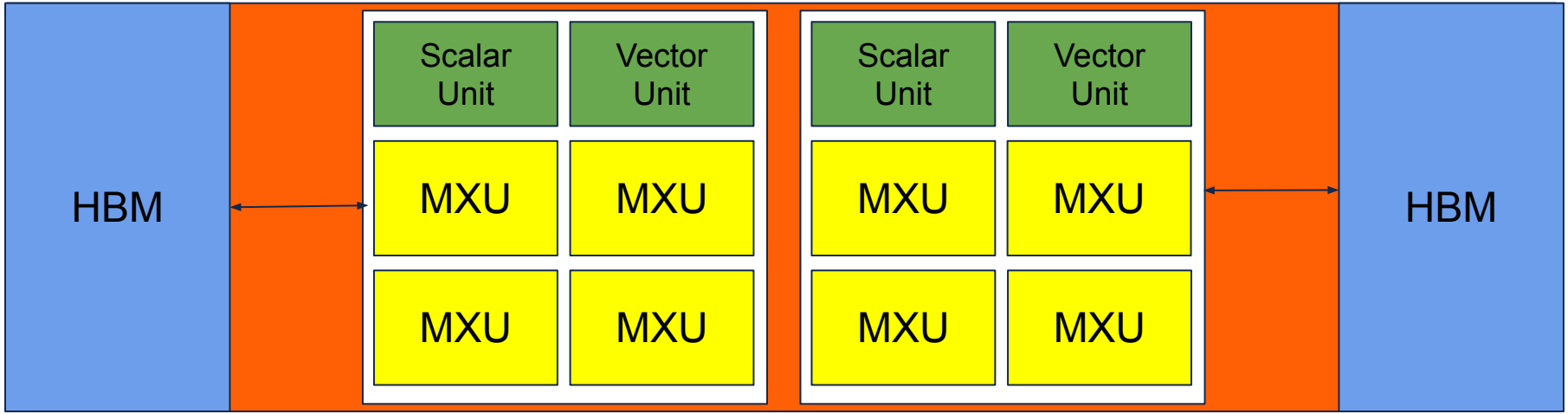


TPU v4

- Improved Interconnect Topology
- Now able to connect up to 4096 chips in a 3D Torus



TPU v4 Board



HBM capacity/bandwidth: 32GiB, 1200 GB/s (33% ↑)
Measured min/mean/max power: 90/170/192 W (~25% ↓)
Peak compute per chip: 275 teraflops (~2 times ↑)
Peak compute per pod (4096 chips): 1.1 exaflops (~10 times ↑)
Bisection bandwidth per pod (4096 chips): 24 TB/s (~4 times ↑)

Eyeriss

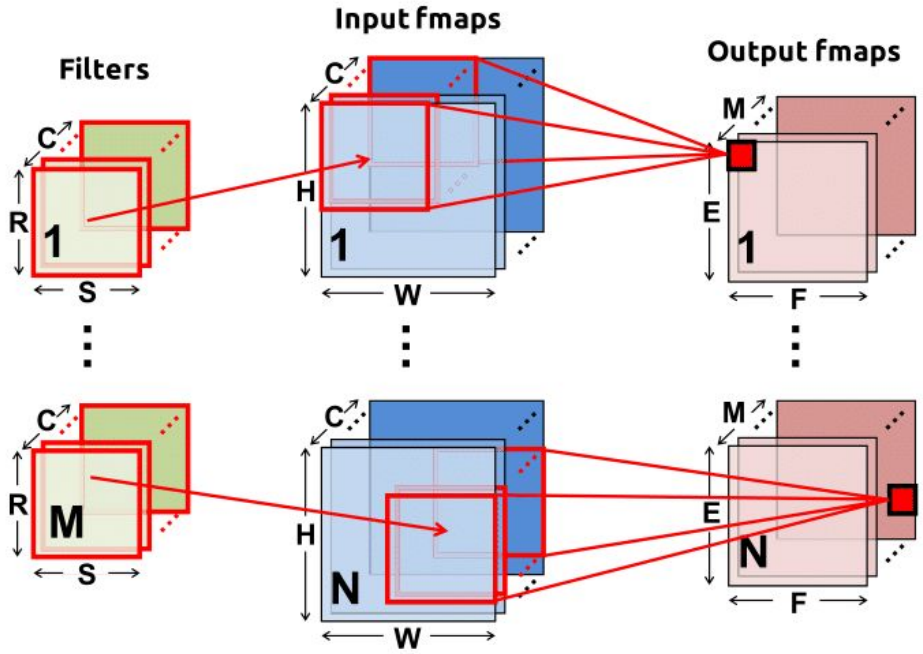


Fig. 1. Computation of a CNN layer.

Eyeriss

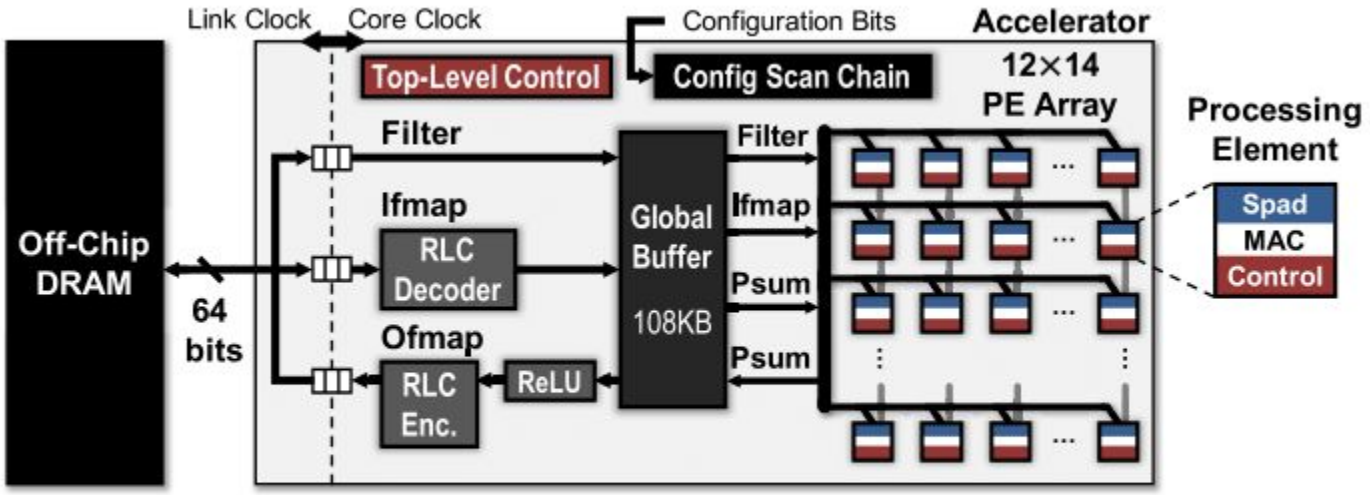


Fig. 2. Eyeriss system architecture.

Eyeriss

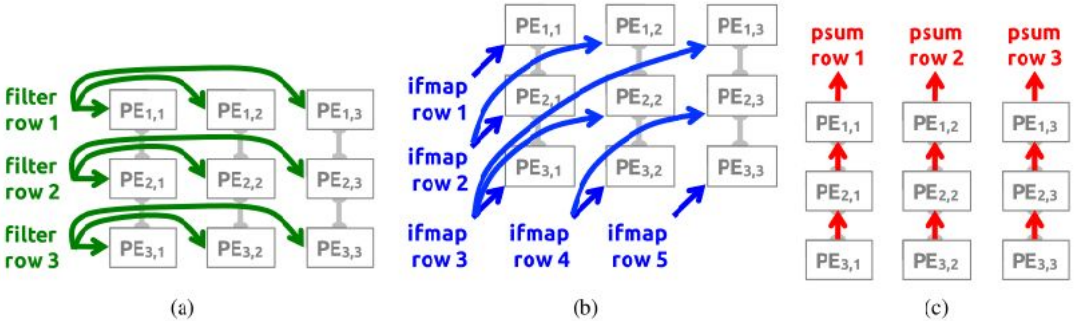


Fig. 4. Dataflow in a PE set for processing a 2-D convolution. (a) Rows of filter weights are reused across PEs horizontally. (b) Rows of ifmap values are reused across PEs diagonally. (c) Rows of psums are accumulated across PEs vertically. Reuse and accumulation of data within a PE set reduce accesses to the GLB and DRAM, saving data movement energy cost. In this example, the number of filter rows (R), ifmap rows (H), and ofmap rows (E) are 3, 5, and 3, respectively. Therefore, the PE set size is 3×3. Filter and ifmap values from different rows are sent to the PE set in a time-interleaved fashion; all the PEs that reuse the same value receive it at the same cycle. The psums generated from one PE are sent to its neighbor PE immediately.

Eyeriss

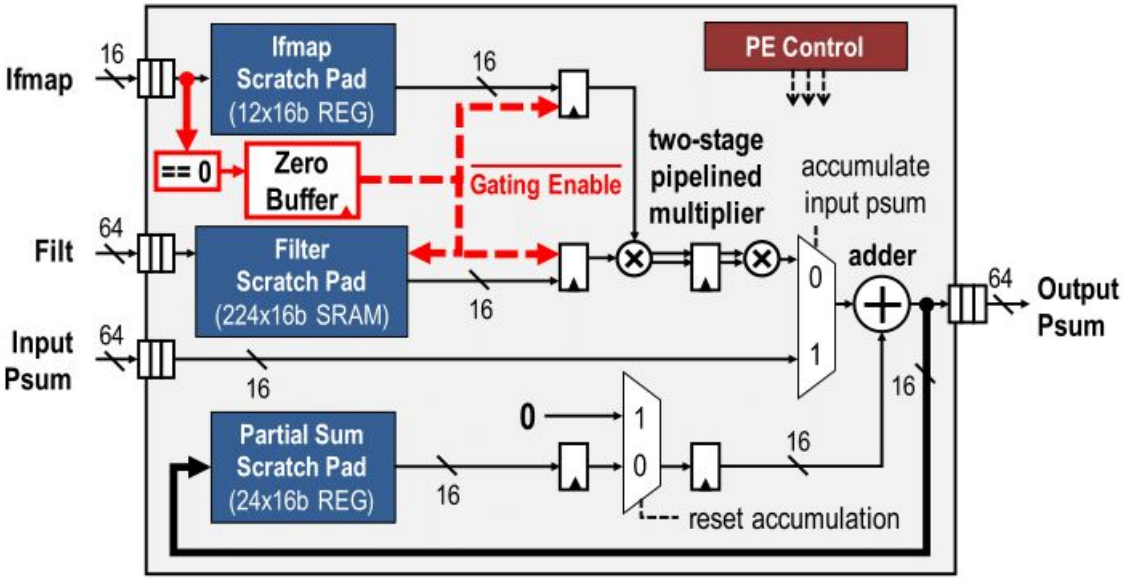


Fig. 12. PE architecture. The datapaths in red show the data gating logic to skip the processing of zero ifmap data.

References

Jouppi, Norman P., et al. "In-datacenter performance analysis of a tensor processing unit." Proceedings of the 44th annual international symposium on computer architecture. 2017.

Jouppi, Norman P., et al. "A domain-specific supercomputer for training deep neural networks." Communications of the ACM 63.7 (2020): 67-78. Retrieved from: https://cacm.acm.org/magazines/2020/7/245702-a-domain-specific-supercomputer-for-training-deep-neural-networks/fulltext

Jouppi, Norman P., et al. "xTen lessons from three generations shaped Google's TPUv4i: Industrial product." 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA). IEEE, 2021.

Hennessy, John L., and David A. Patterson. Computer architecture: a quantitative approach. Elsevier, 2011.

Google LLC., https://cloud.google.com/tpu/docs/system-architecture-tpu-vm

Google LLC., https://cloud.google.com/blog/products/ai-machine-learning/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu

Y. -H. Chen, T. Krishna, J. S. Emer and V. Sze, "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks," in *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127-138, Jan. 2017, doi: 10.1109/JSSC.2016.2616357.

Deep Neural Network Diagram, Retrieved from: https://www.researchgate.net/figure/Concept-of-Deep-Learning_fig19_319999320