CASPER

CS773: Computer Architecture for Performance and Security

Lecture 7: 10K Feet View on the Domain Specific Architectures(DSAs)

https://www.cse.iitb.ac.in/~biswa/

Thanks Moore (Moore's Law)

Transistors doubling every 18 months ☺ 180 nm to 7nm ☺

Three levels of cache, prefetcher, TLBs,

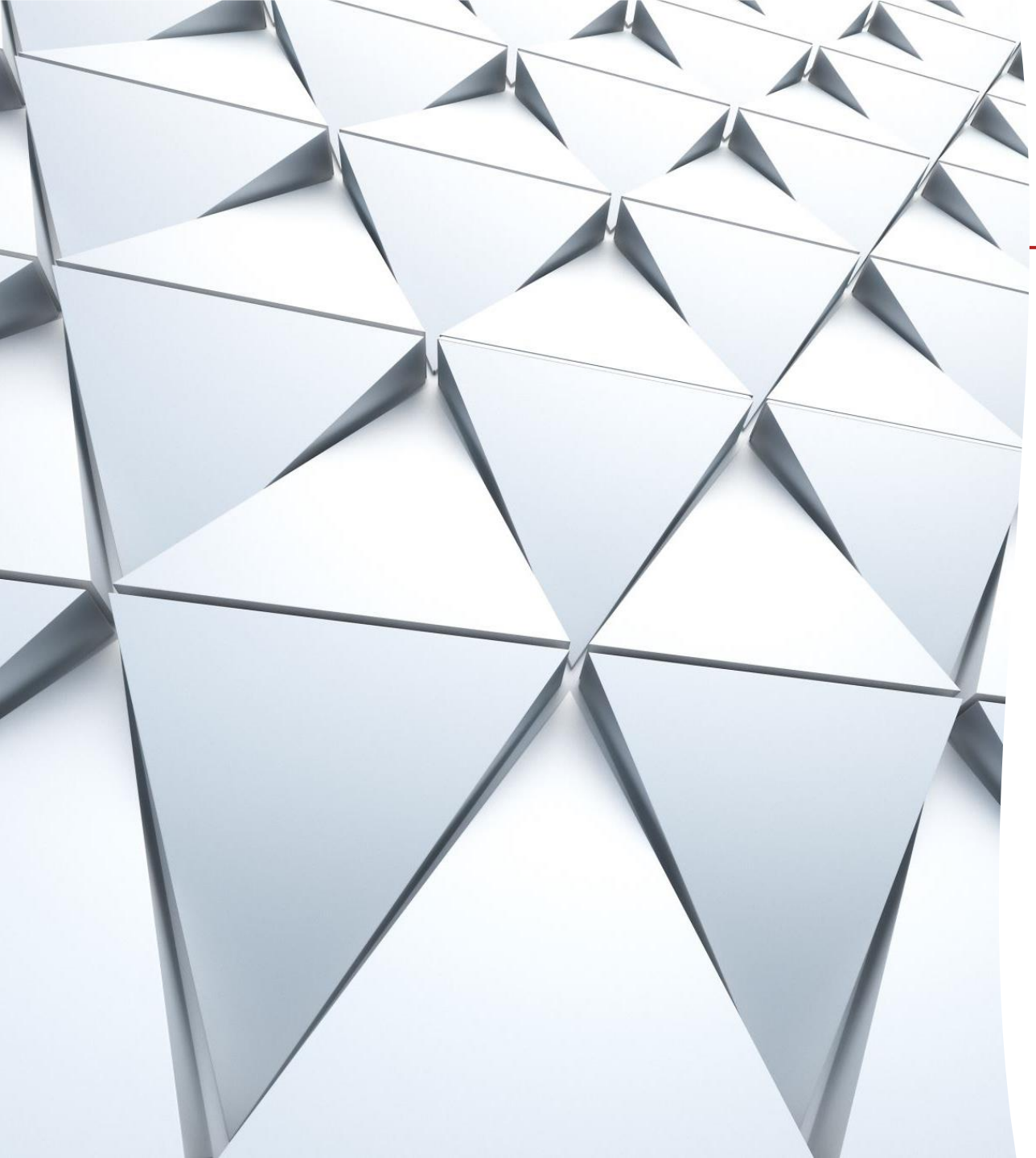
Multi-level BTBs, branch predictors

Speculative execution, O3 scheduling, 20-stage pipeline

Multi-threading

Multi-core

Performance, performance and performance !!



Dennard Scaling is dead

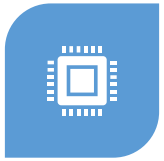
- Power density used to be constant ☹
- So more transistors meaning more power ☹
- Energy numbers:
- LOAD/STORE: 150 pJ
- 32-bit addition: 7 pJ
- 8-bit addition: 0.2 pJ

Do we need them all?

COMPUTER ARCHITECTURE RESEARCH: LARGE SOFTWARE WRITTEN IN C/C++/FORTRAN

GENERAL PURPOSE IN NATURE

WHAT ABOUT NEURAL NETS, VISION, AI, GRAPH ANALYTICS?



WE DO NOT NEED A POWERFUL GENERAL-PURPOSE PROCESSOR

INSTEAD, WE NEED A SUPER-EFFICIENT DOMAIN SPECIFIC COMPUTE ENGINE

SOFTWARE PORTABILITY ☹

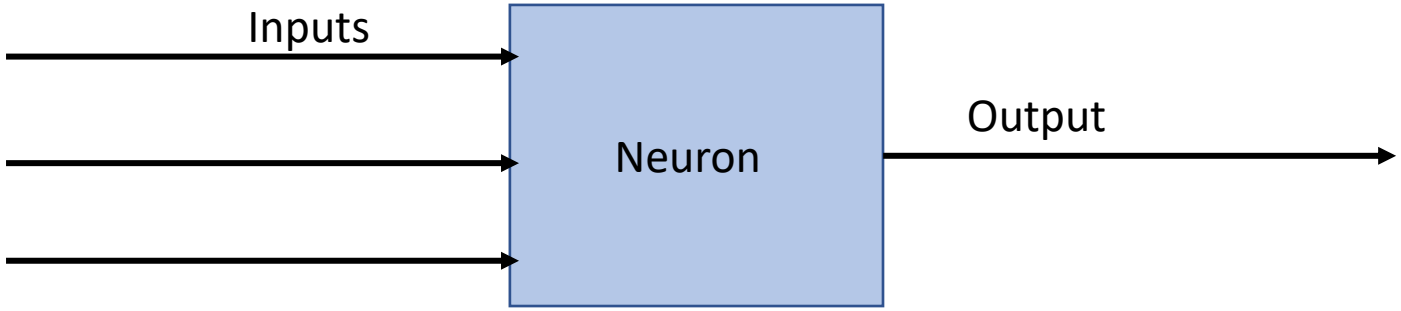
DSA Mantra in 2020

- Use software controlled dedicated memory with no hierarchy ??
- More ALUs and memory units instead of microarchitecture optimizations
- Forget ILP and think about domain specific parallelism.
- Reduce the data type and size for high memory bandwidth
- Domain specific languages for DSA: TensorFlow for example

DNNs (Deep Neural Networks)

Computes the non-linear activation function of weighted sum of inputs.

E.g., image processing (does the image contain a dog)
input:pixels, activation function: f(x)=max(x,0), can have many layers

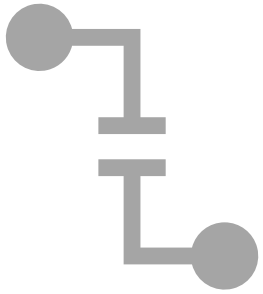


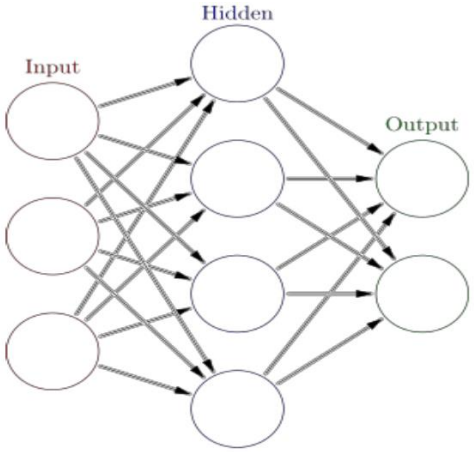
Examples

Name	DNN layers	Weights	Operations/Weight
MLP0	5	20M	200
MLP1	4	5M	168
LSTM0	58	52M	64
LSTM1	56	34M	96
CNN0	16	8M	2888
CNN1	89	100M	1750

Training (learning) vs Inference

Once the neural architecture has been selected (#layers, architecture etc.)





Next step: learn the weights associated with each edge of neural network graph.

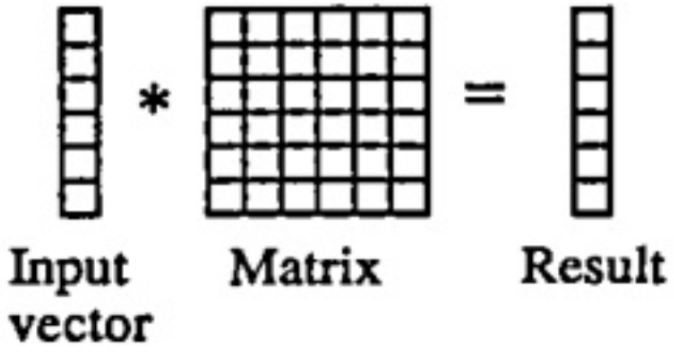
Most DNNs are supervised: training set to learn where data is preprocessed

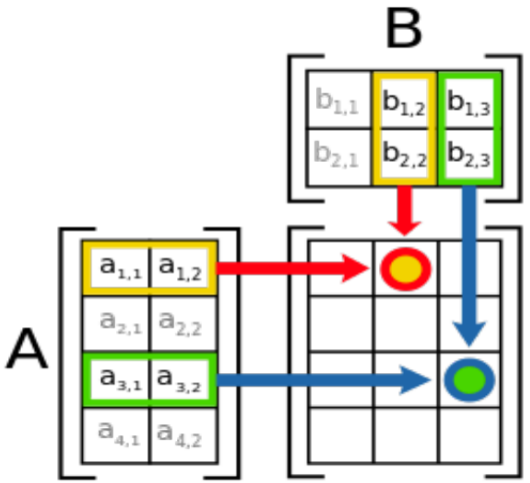
Training vs Inference

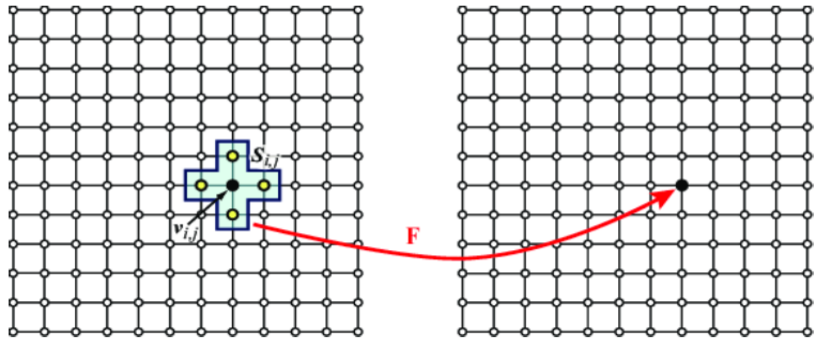
TRAINING CAN TAKE WEEKS TO MONTHS ☺

INFERENCE: MILLION TIMES LESS TIME ☺ ☺

Takeaway

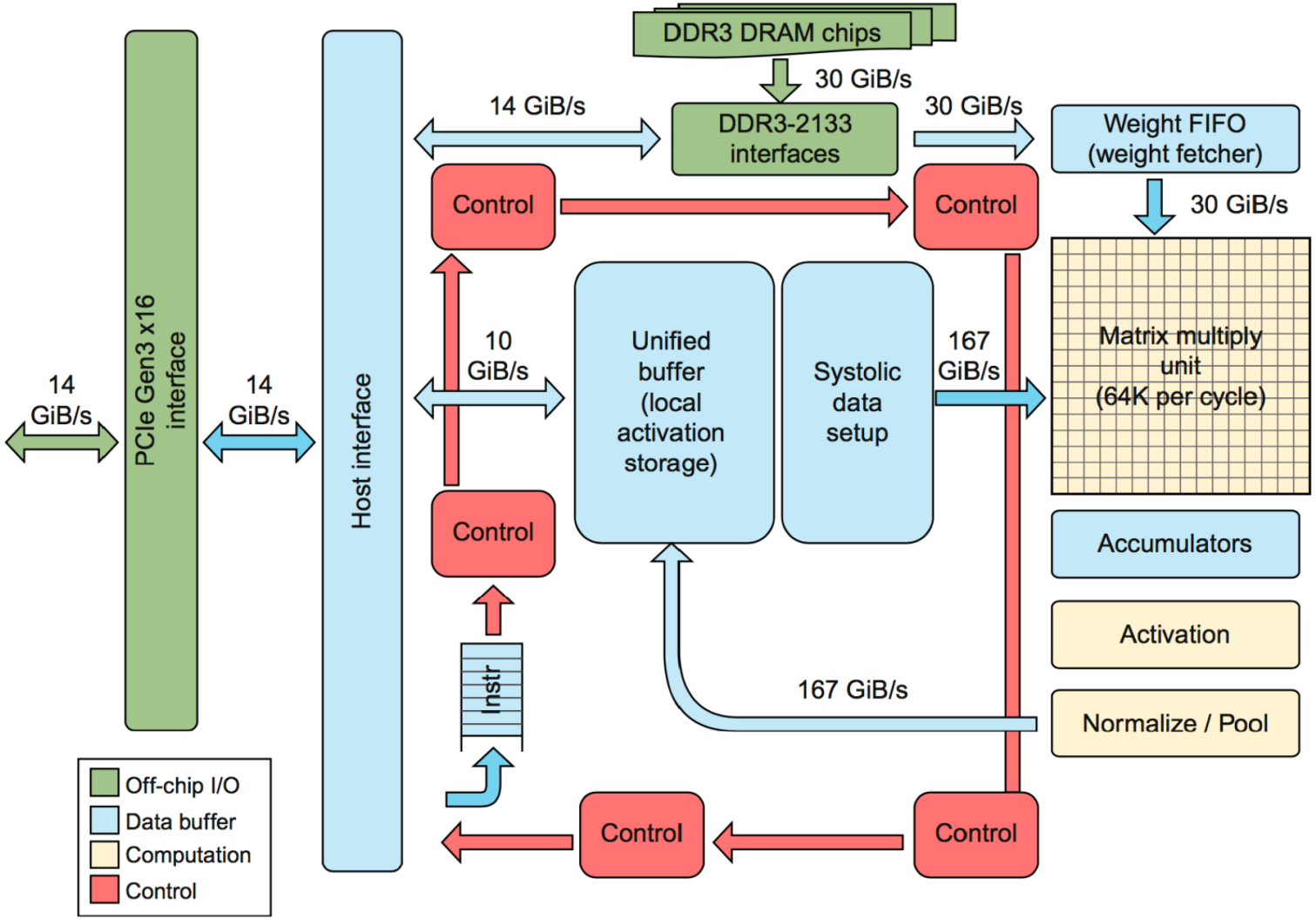






Need: highly efficient compute engines that can
perform matrix-multiply, vector-matrix-multiply, stencil computations

Google's TPU (Tensor Processing Unit)



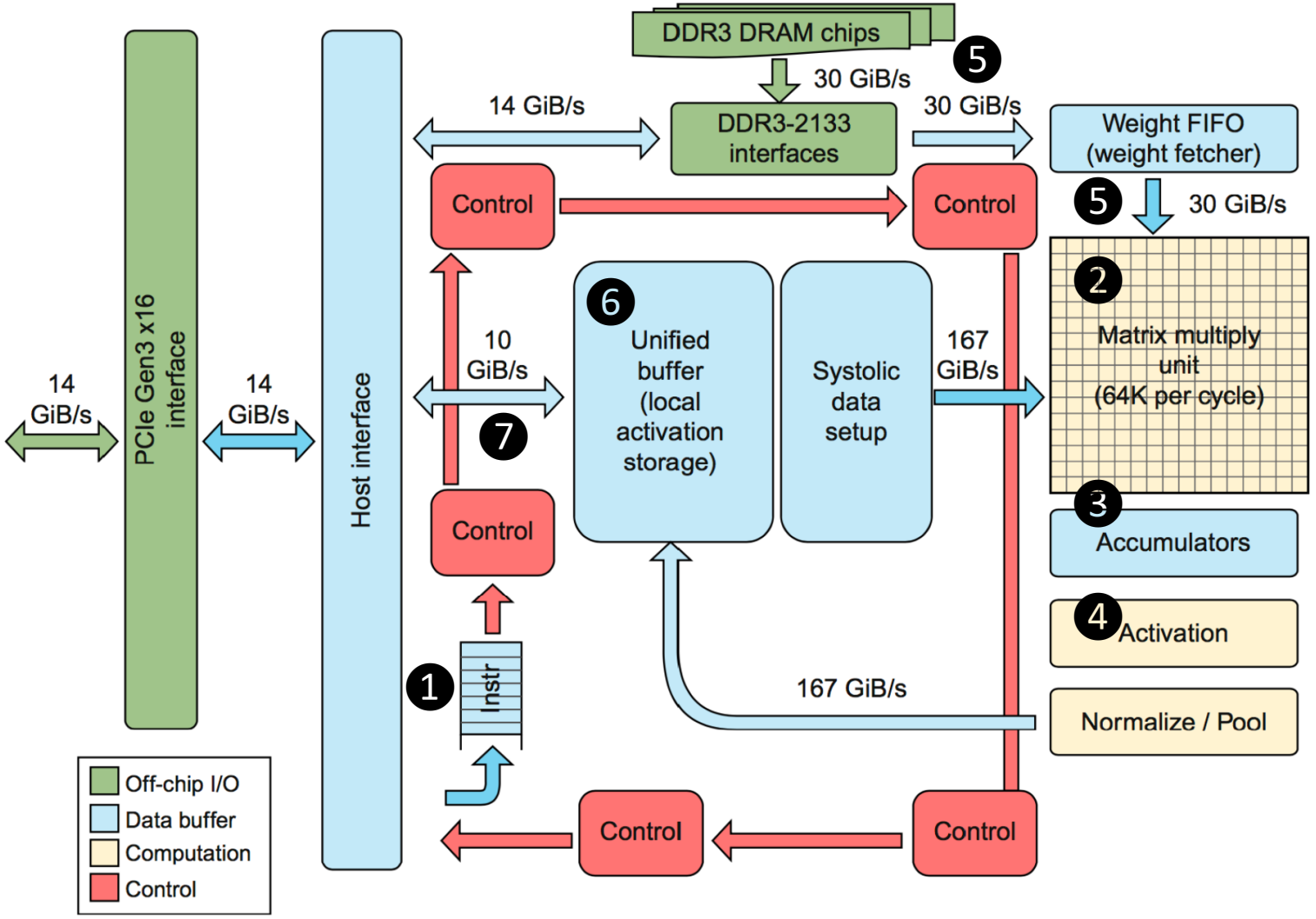
Heart of TPU (co-processor):

(i) 65,563 (256X256) 8-bit ALU matrix multiply unit

(ii) Large software-managed on-chip memory

Where are the L1/L2/L3s ☹

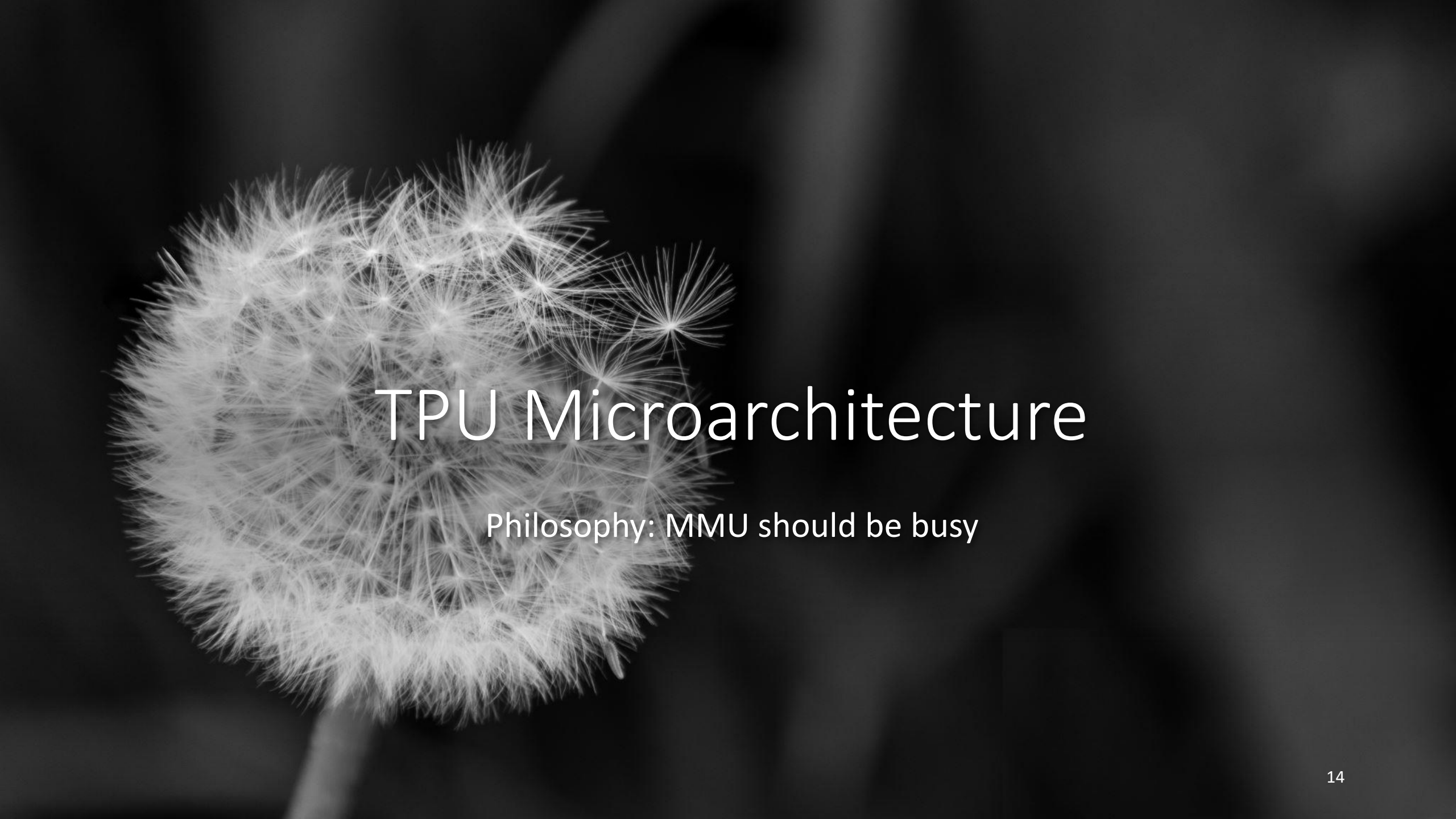
TPU Architecture



1. Host CPU sends TPU instructions over the PCIe bus into the instruction buffer. Internal blocks are connected through 2048-bit paths.
2. Heart of the TPU: MMU (Matrix-multiply-unit)
3. Products are stored in 4MB of 32-bit accumulators.
4. Non-linear functions are calculated.
5. Weights are fetched through weight fetcher.
6. Intermediate results are stored in a 24MB buffer, which serves as input to MMU again.
7. Programmable DMA controller transfers data to/from DRAM-unified buffer

TPU instructions (CISC ☺, no branch instructions ☺)

- Read_Host_Memory
- Read_Weights
- MatrixMultiply
- Activate
- Writes_Host_Memory



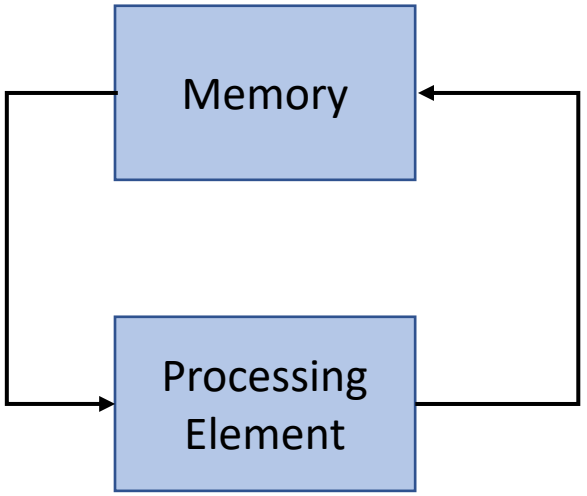
TPU Microarchitecture

Philosophy: MMU should be busy

Reading from 24MB Unified Buffer ☹

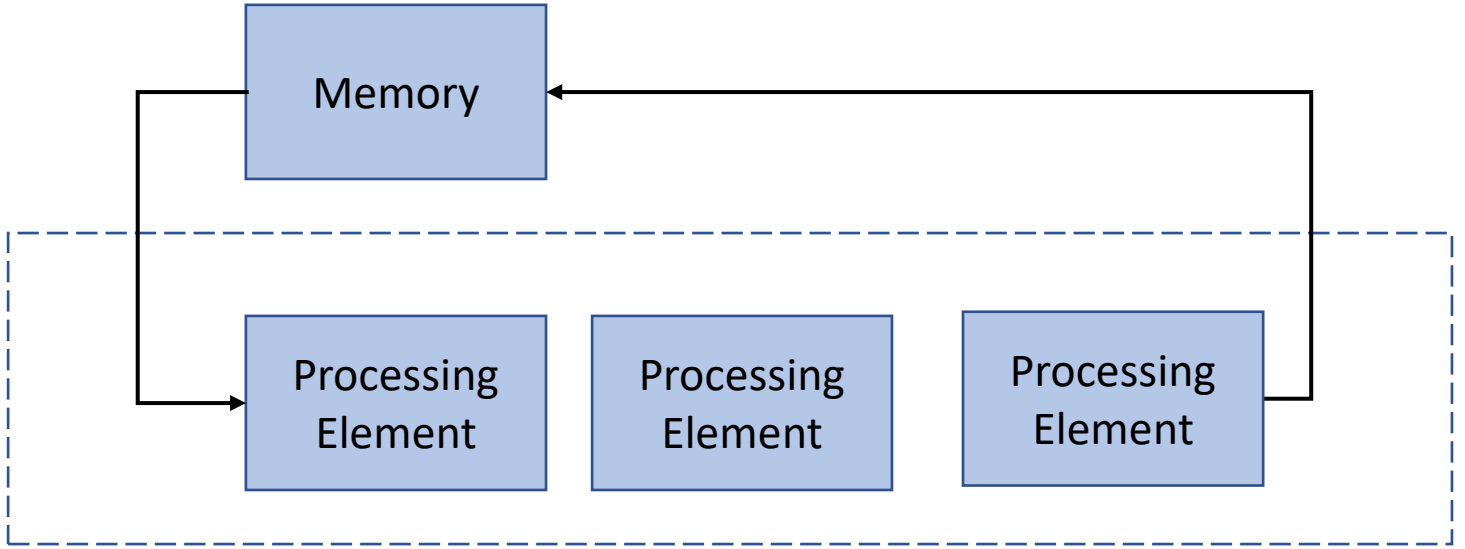
Expensive in terms of energy ☹

Systolic Arrays



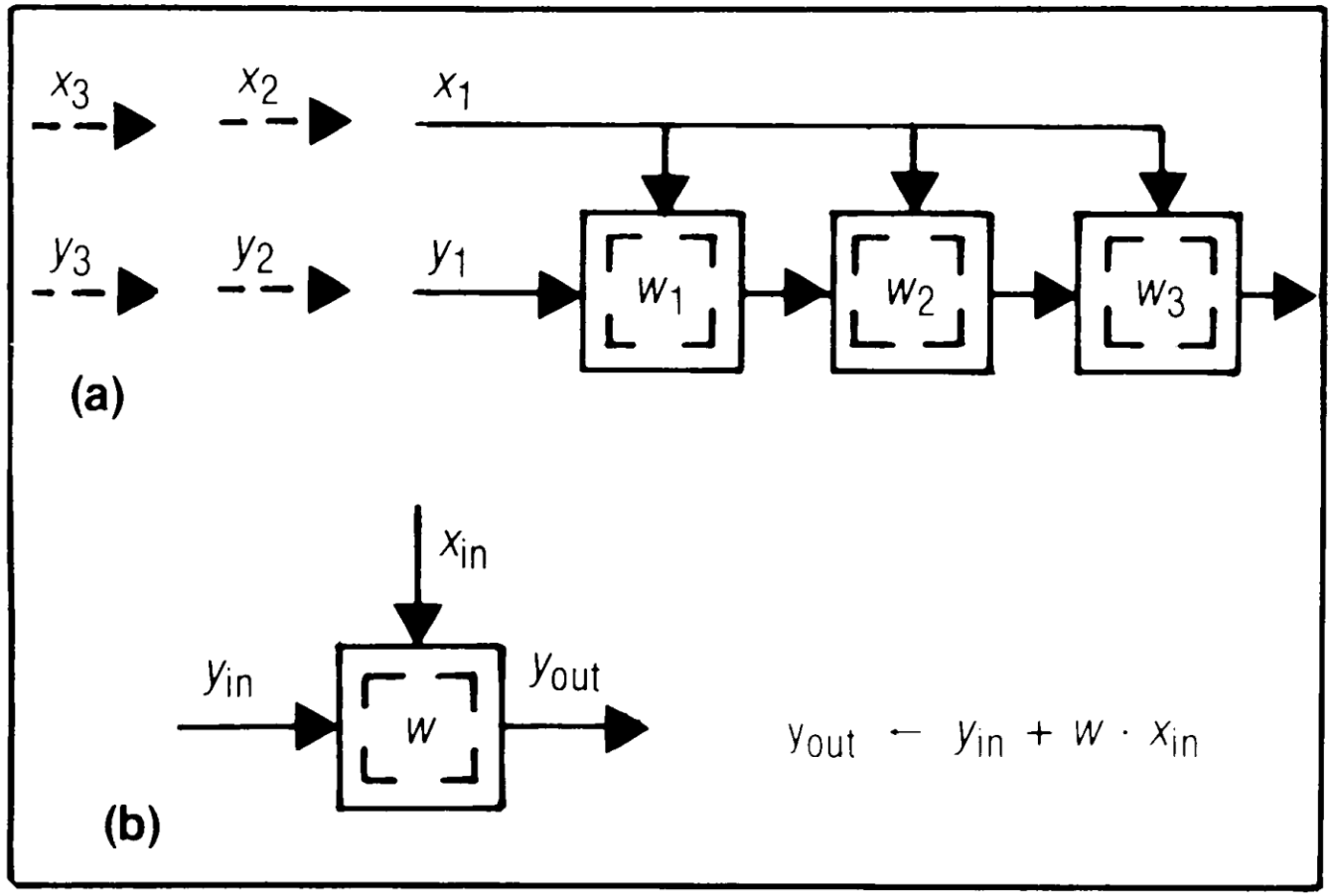
Conventional

http://www.eecs.harvard.edu/~htk/publication/1982-kung-why-systolic-architecture.pdf



Memory: Heart
PE: cells

A possible design



Usage

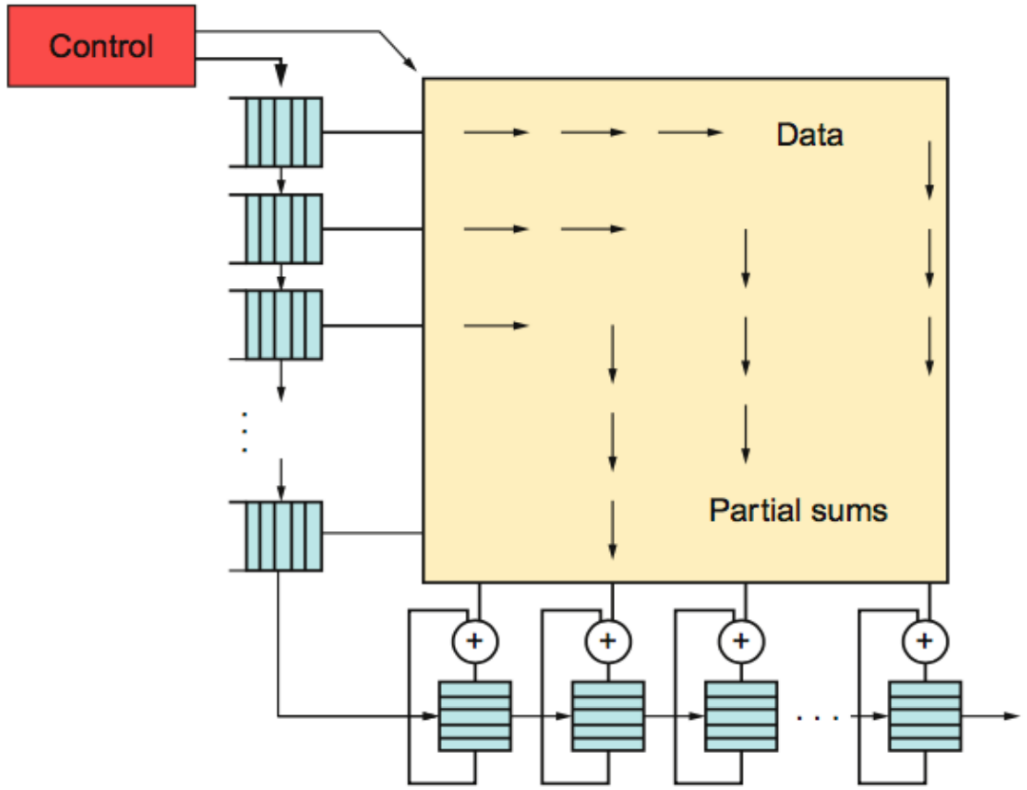
Pattern matching, filtering, ….

Given sequence of weights w1, w1, w3, …. and input sequence of x1, x2, …..

Result sequence: y1, y2, …..

y(i) = w1x1 + w2x2 + …..

Systolic Execution of MMU



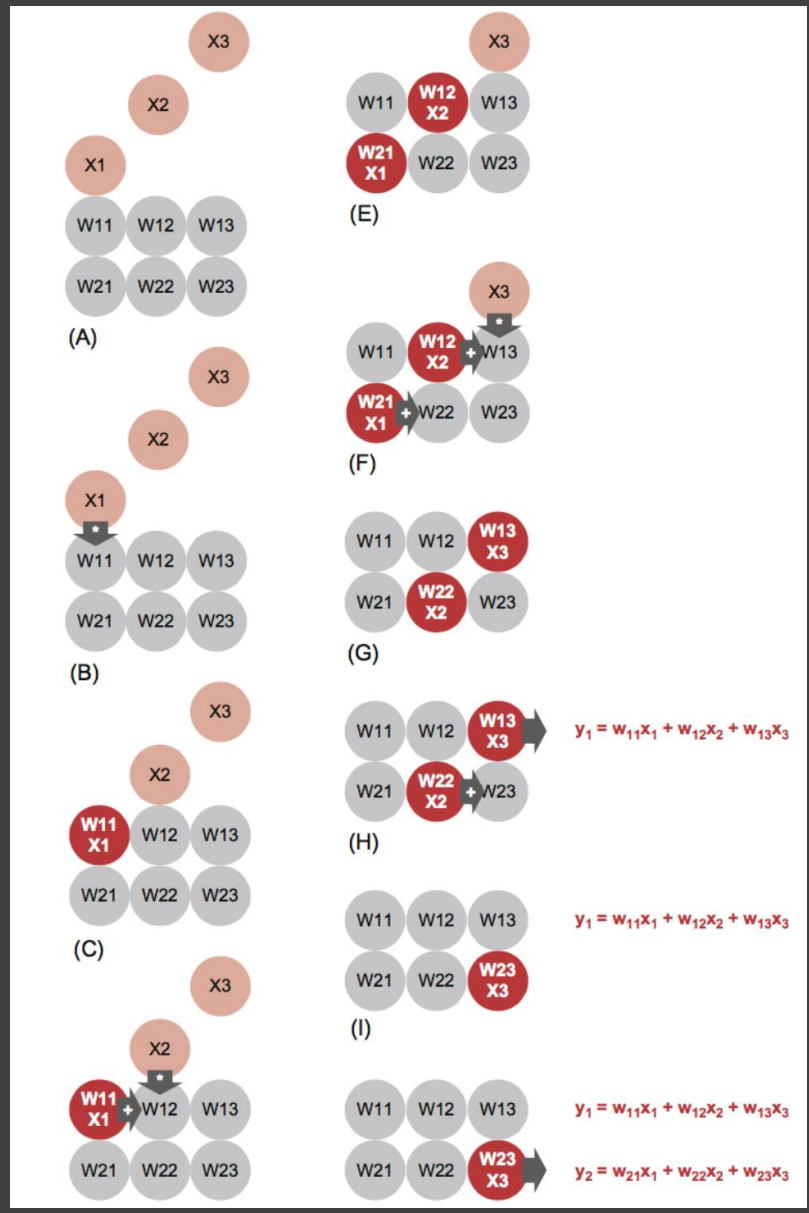
2-D collection of arithmetic units, operate independently

Independent computation of partial sum

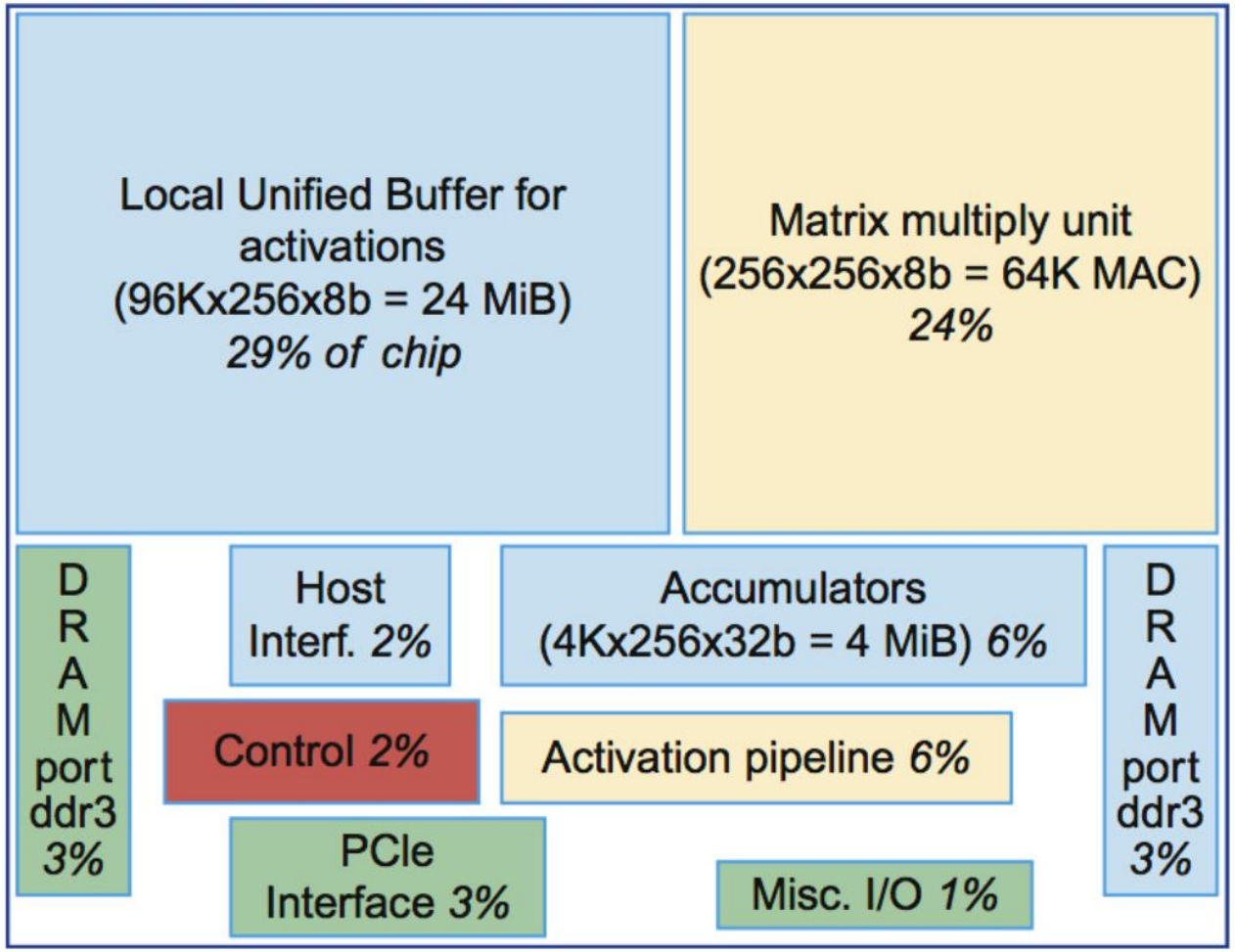
Relies on data coming from multiple directions, arriving at cells at a regular intervals where it is combined

Data flows through an array as an advancing wave front, similar to blood being pumped in circulatory system

An Example


X3
X2
X1
W11 W12 W13
W21 W22 W23
(A)
X3
X2
X1
W11 W12 W13
W21 W22 W23
(B)
X3
X2
W11 X1
W12 W13
W21 W22 W23
(C)
X3
X2
W11 X1
W12 W13
W21 W22 W23
X3
W11 W12 X2 W13
W21 X1 W22 W23
(E)
X3
W11 W12 X2 W13
W21 X1 W22 W23
(F)
W11 W12 W13 X3
W21 W22 X2 W23
(G)
W11 W12 W13 X3
W21 W22 X2 W23
$y_1 = w_{11}x_1 + w_{12}x_2 + w_{13}x_3$
(H)
W11 W12 W13
W21 W22 W23 X3
$y_1 = w_{11}x_1 + w_{12}x_2 + w_{13}x_3$
(I)
W11 W12 W13
W21 W22 W23 X3
$y_1 = w_{11}x_1 + w_{12}x_2 + w_{13}x_3$
$y_2 = w_{21}x_1 + w_{22}x_2 + w_{23}x_3$

TPU Floorplan



Half of the size of Intel Haswell ☺

DSA Mantra and TPU

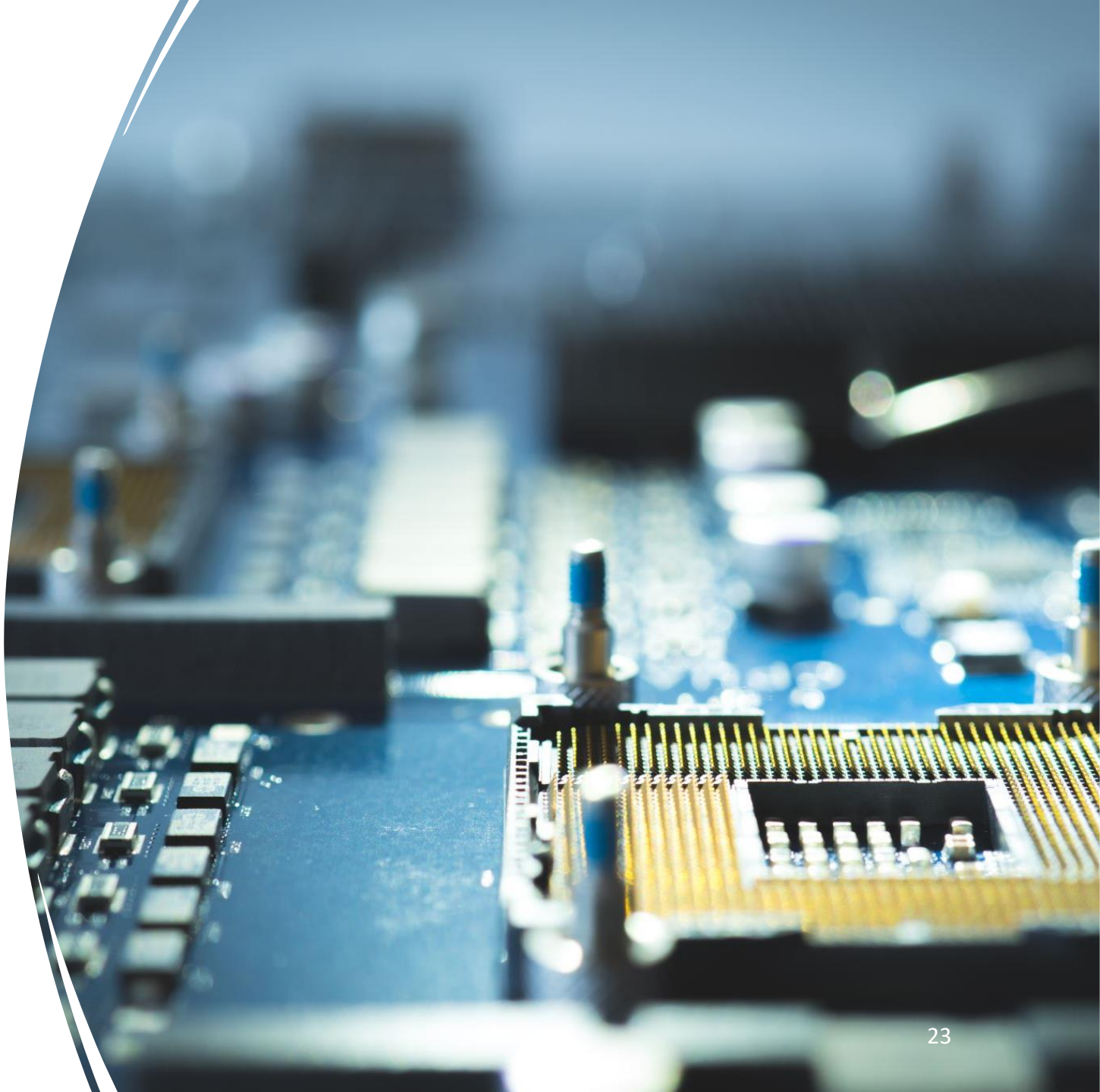
- Use dedicated memories: 24MB buffer and 4MB accumulators
- Invest resource in arithmetic units
- Easiest form of parallelism: 2D systolic array
- Reduce the data size: 8-bit integers
- Domain-specific programming language: TensorFlow

Bottlenecks

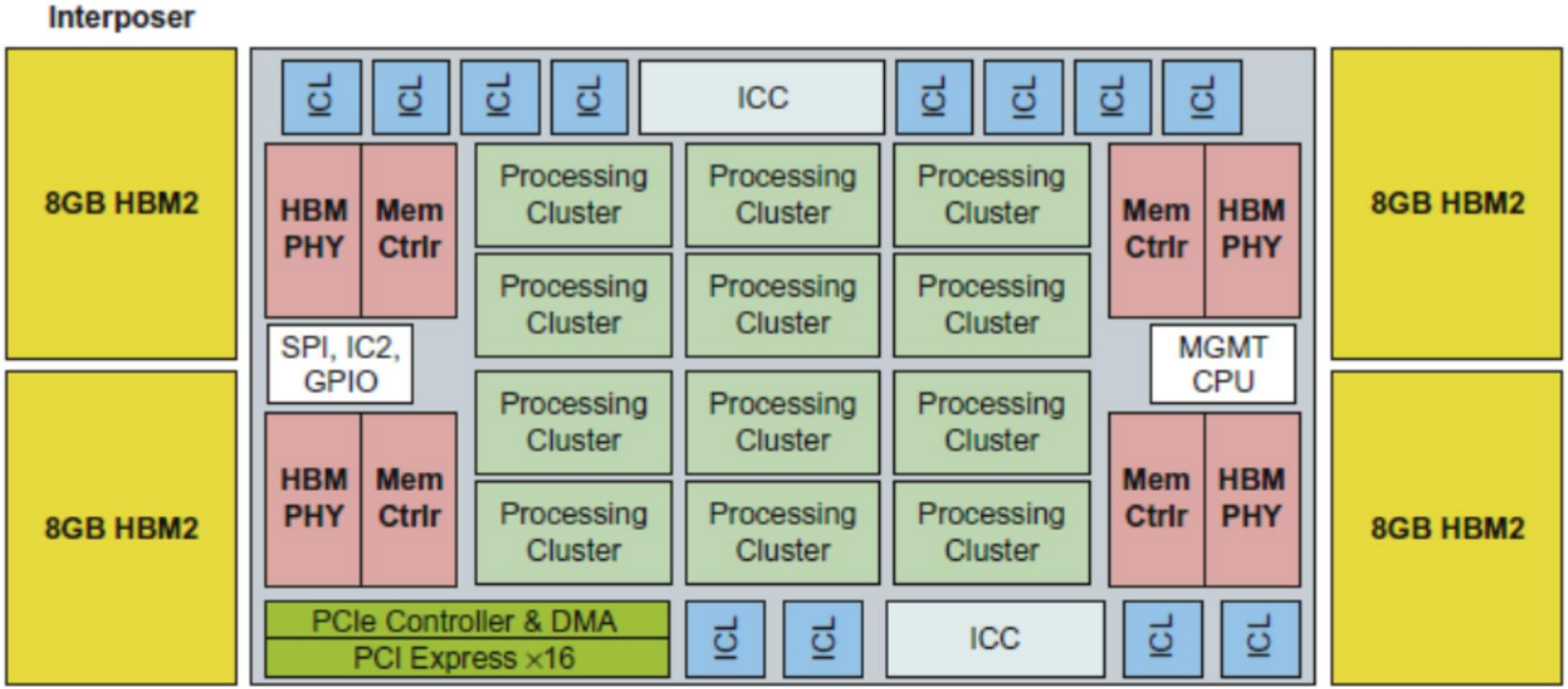
- DRAM bandwidth, again ☹
- Use high bandwidth memory
- We did not cover the system stack of TPU.

Try it out:

https://cloud.google.com/tpu/docs/tpus



Intel Crest (16-bit fixed point, 32X32 matrices)

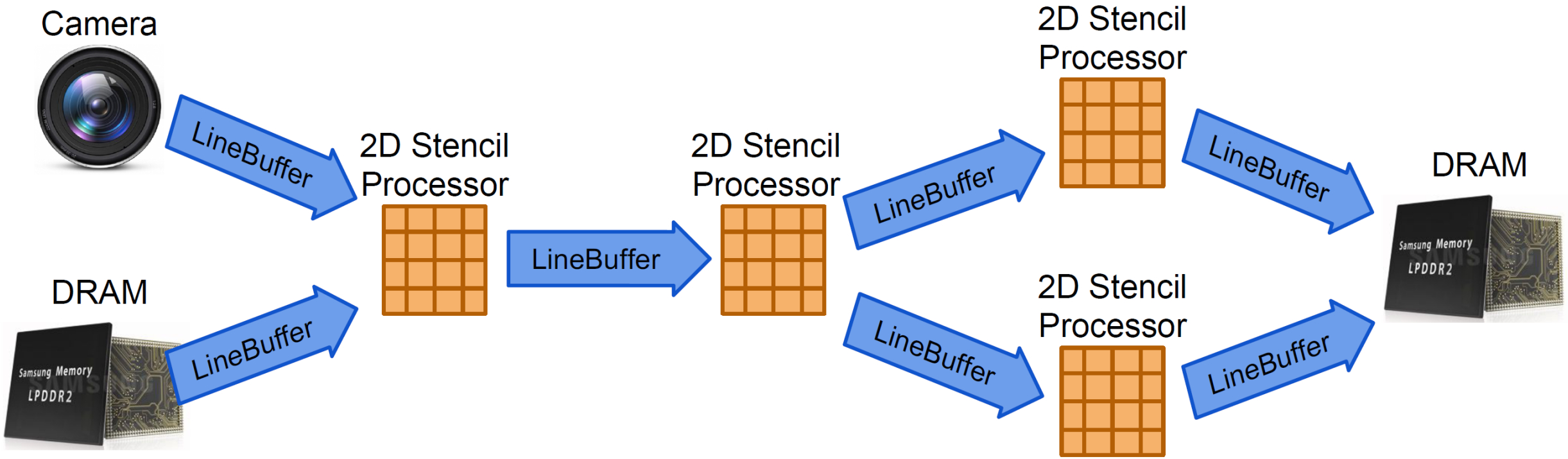



DNN to Vision

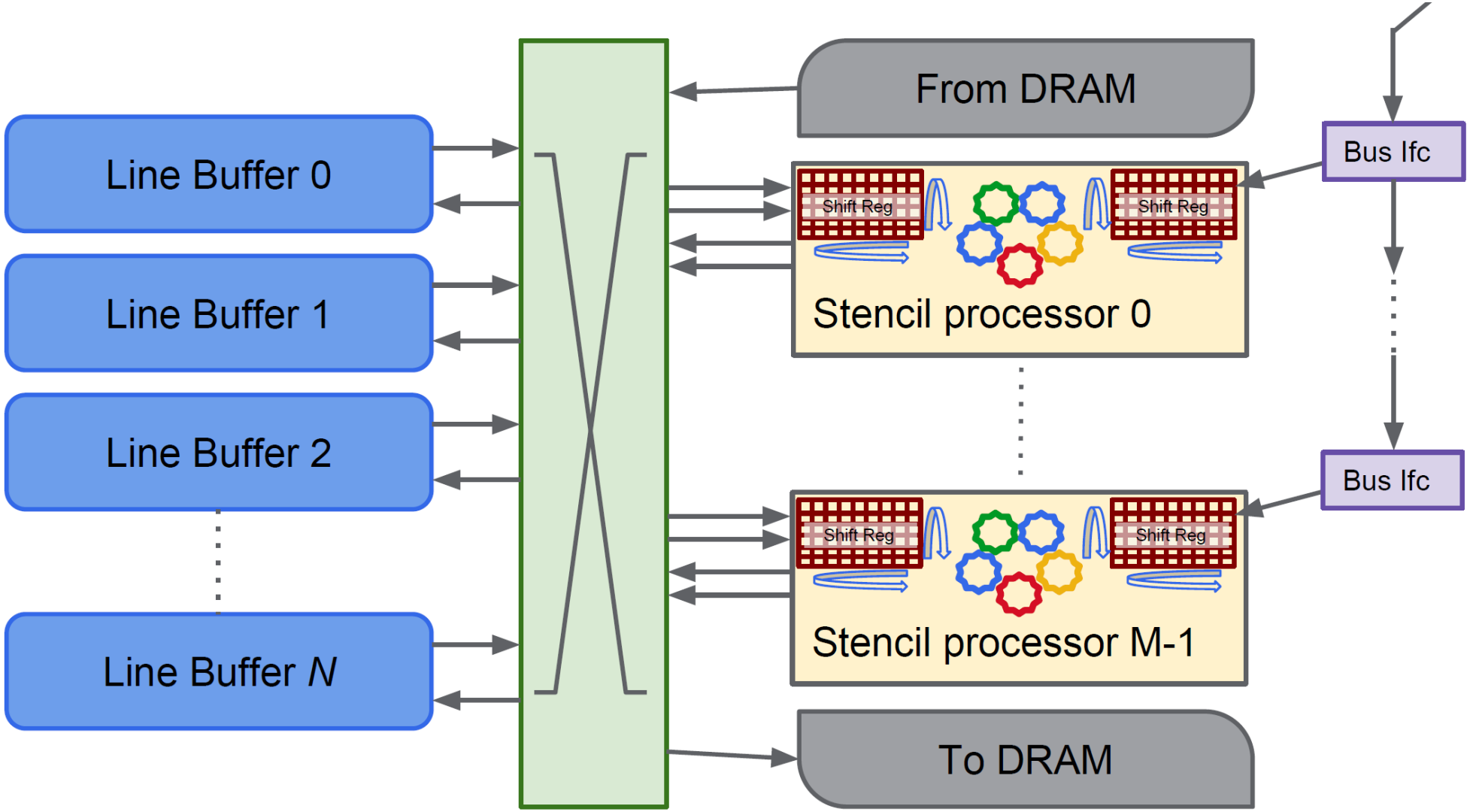
Google’s Pixel Visual Core

- https://blog.google/products/pixel/pixel-visual-core-image-processing-and-machine-learning-pixel-2/

The flow


Camera
LineBuffer
2D Stencil Processor
LineBuffer
2D Stencil Processor
LineBuffer
2D Stencil Processor
LineBuffer
DRAM
DRAM
LineBuffer
LineBuffer
2D Stencil Processor
LineBuffer
Samsung Memory
LPDDR2
Samsung Memory
LPDDR2

Hardware view (Image/stencil processor: Systolic)



Thanks