Topic 8

Specialized Architecture

Xinfei Guo
xinfei.guo@sjtu.edu.cn

July 18th, 2022

Recall: specialized horsemen

- "We will use all of that dark silicon area to build **specialized cores**, each of them tuned for the task at hand (10-100x more energy efficient), and only turn on the ones we need…"
- Insights:
  - Power is now more expensive than area
  - Specialized logic can improve energy efficiency by 10-1000x
- This led to today's heterogeneous systems

90

T8 learning goals

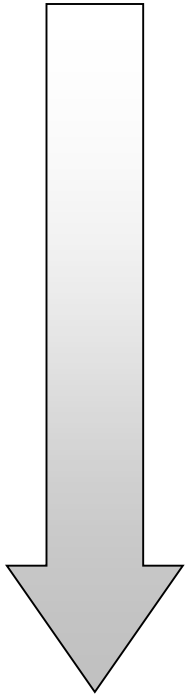
- Reconfigurable Architecture
- Domain Specific Architecture (DSA)
  - AI Accelerators

RECONFIGURABLE COMPUTING

Calculating Class Grades

```
grade = 0.1 × mt1 + 0.2 × mt2
      +0.3 × hw + 0.4 × proj;
```

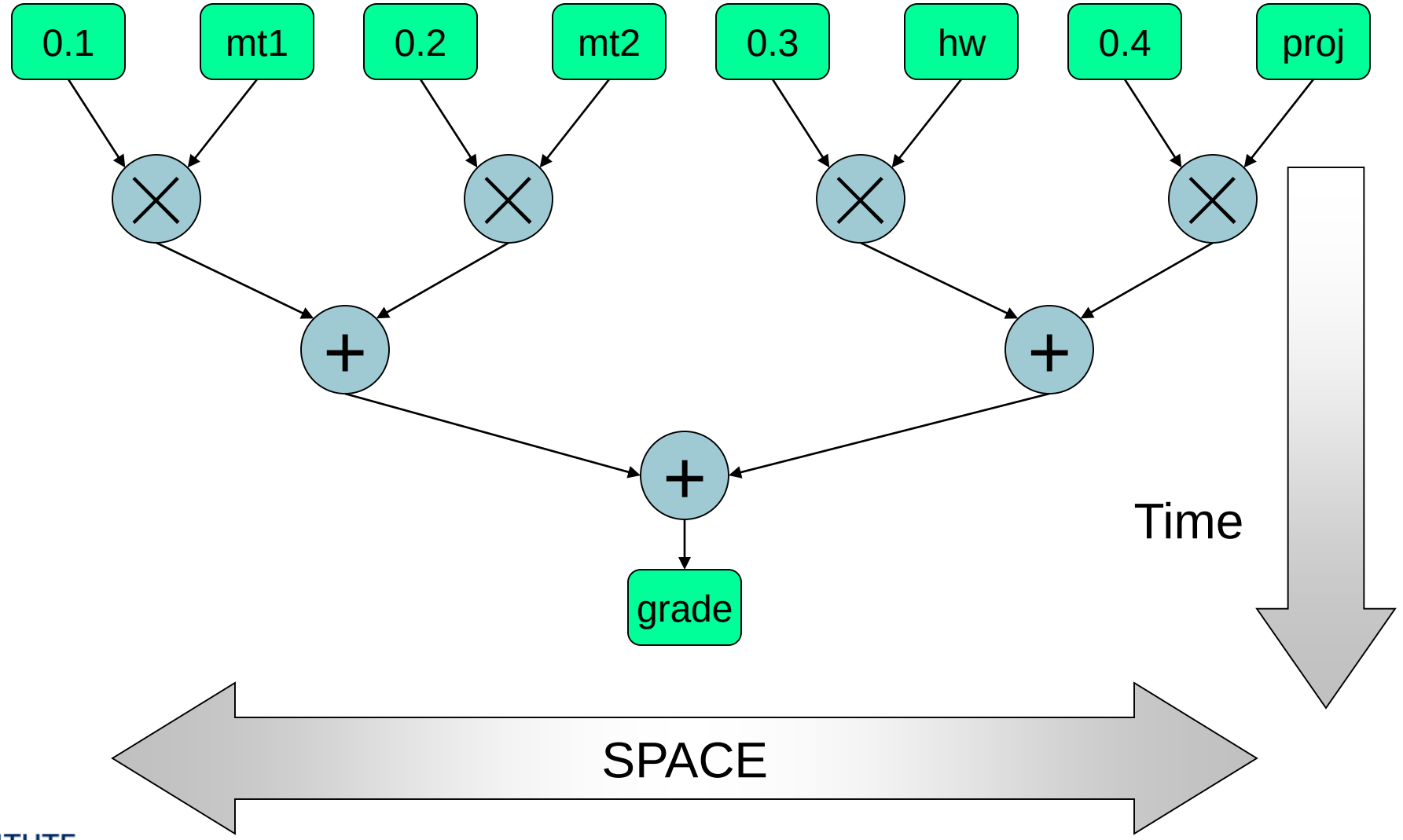
```
grade = 0;
tmp = 0.1 × mt1;
grade = grade + tmp;
tmp = 0.2 × mt2;
grade = grade + tmp;
tmp = 0.3 × hw;
grade = grade + tmp;
tmp = 0.4 × proj;
grade = grade + tmp;
```



Time

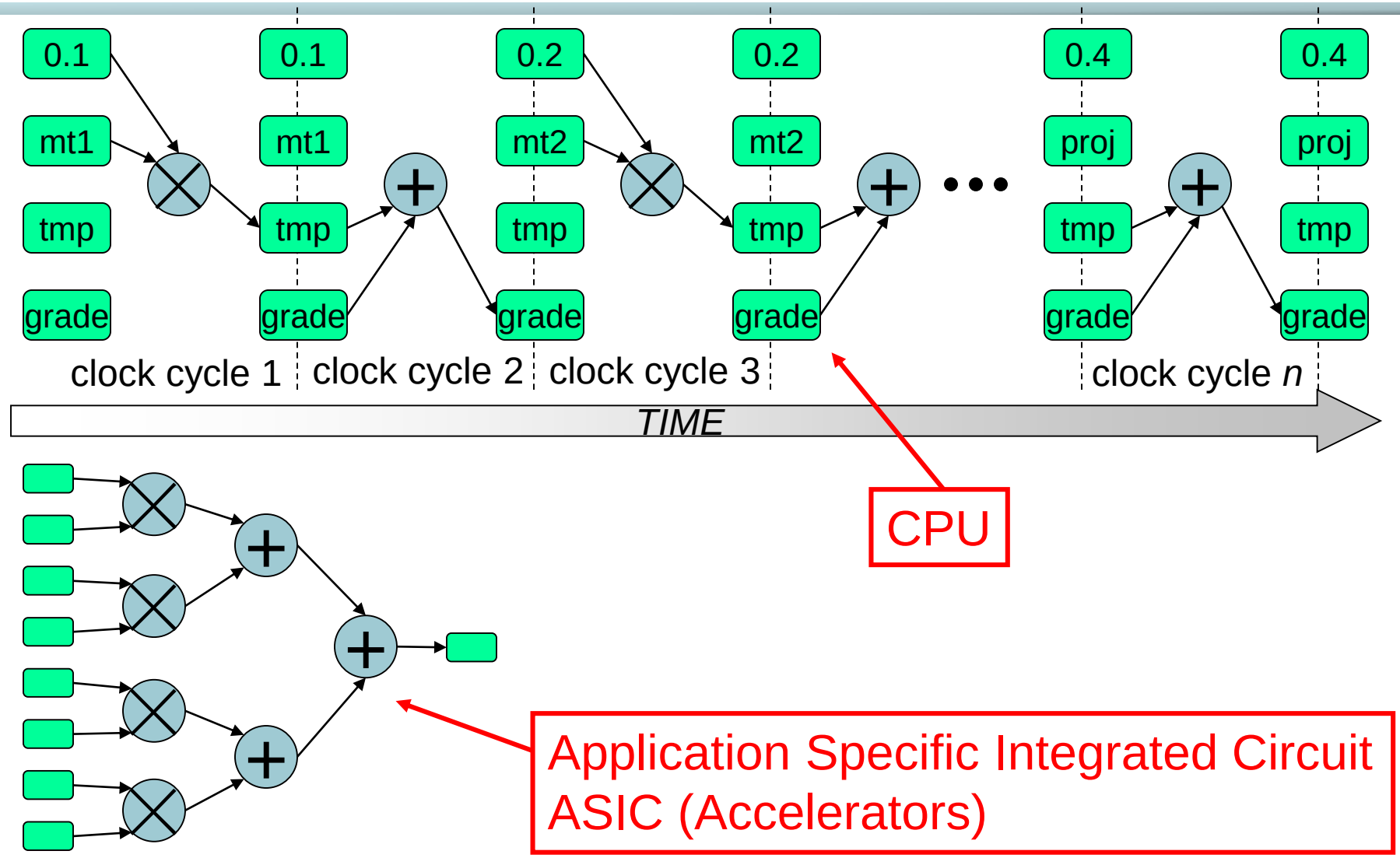
Slide Credit: Hayden So

Computing Final Grade (2)



Slide Credit: Hayden So

2 Ways to Compute



Slide Credit: Hayden So

Processor vs ASIC

- Take longer to compute
  - slow
- Flexible
- Need instructions to determine what to do on each cycle
- Space is bounded

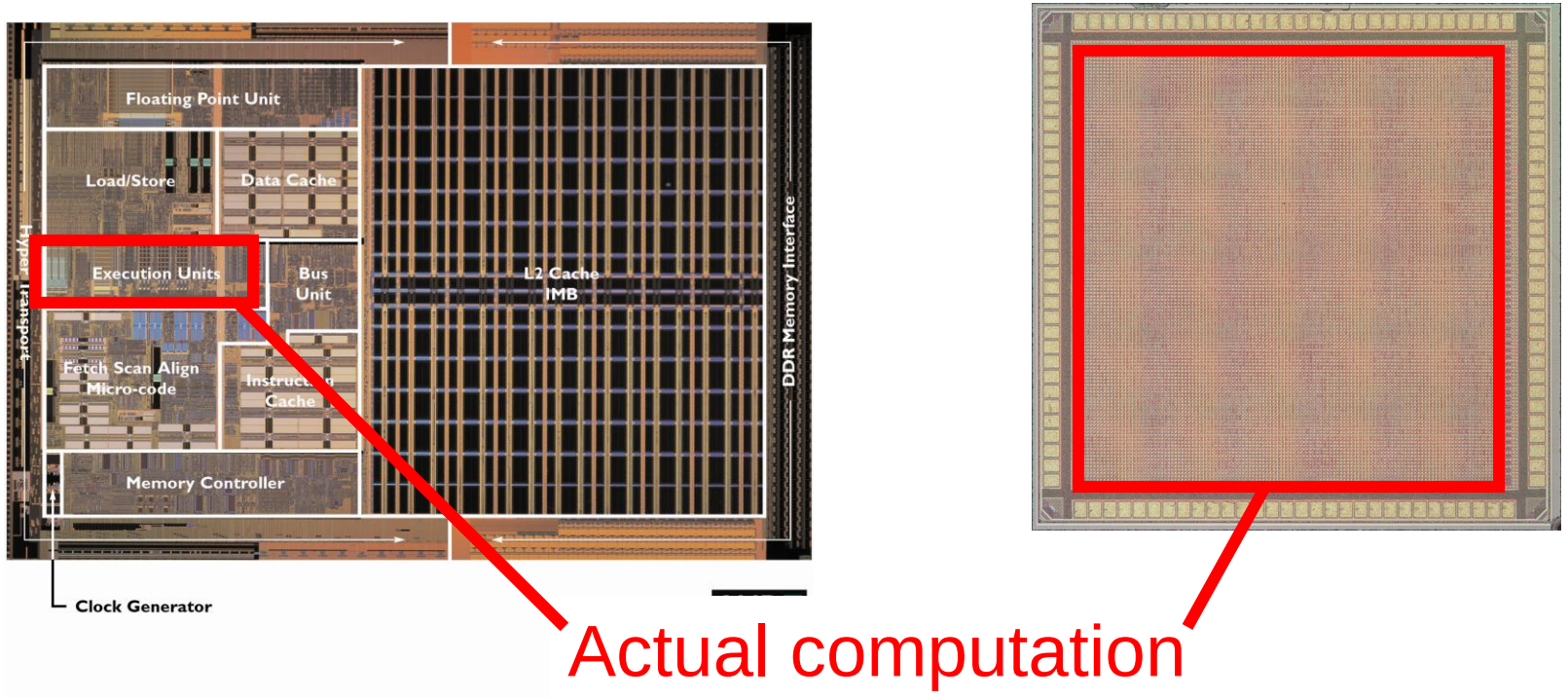
- Take shorter time to compute
  - fast
- Not Flexible
- No instruction
  - Same calculation every cycle
- Space unbounded

Temporal Computing

Spatial Computing

Slide Credit: Hayden So

Visualizing Spatial Computing

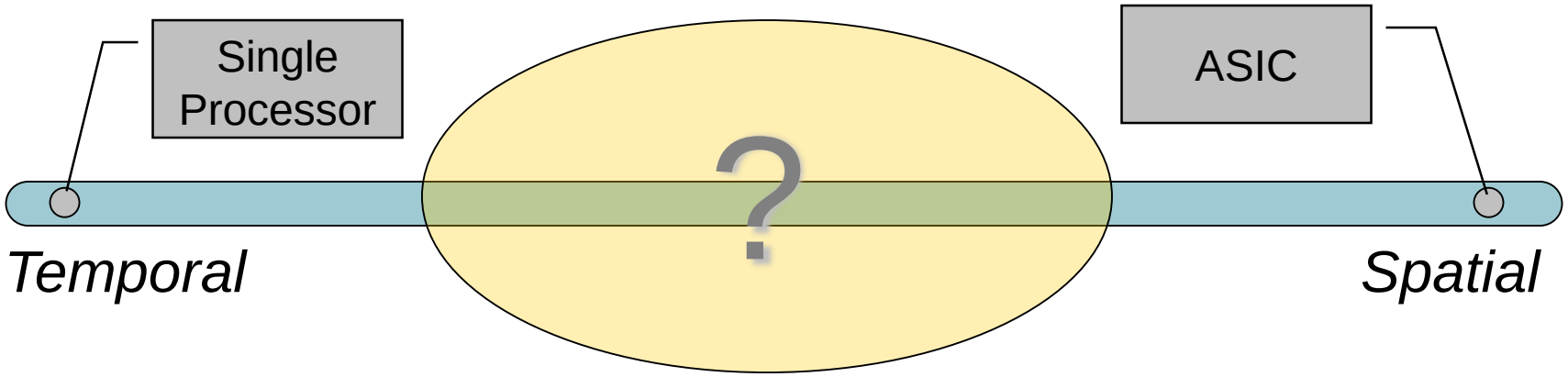


- AMD Opteron 64-bit processor
  - 1MB L2 Cache
- 193 mm sq
  - 0.18 micron CMOS
- 89W @ 1.8GHz
- ~3 Op / cycle (int op)

- Full Custom ASIC
  - 4x4 SVD Decomposition
- 3.5 mm sq
  - 90nm CMOS
- 34mW @ 100 MHz clock
- 70 GOPS = 700 Op / cycle

Slide Credit: Hayden So

Between Temporal & Spatial Computing



- Slow
- Flexible

Reconfigurable Computing

- Fast
- Inflexible

Spatial mapping of computation versus multiplexing of function units (as in processors) **relieves pressure for memory** capacity, BW, and promotes local communication patterns

Reconfigurable Computing

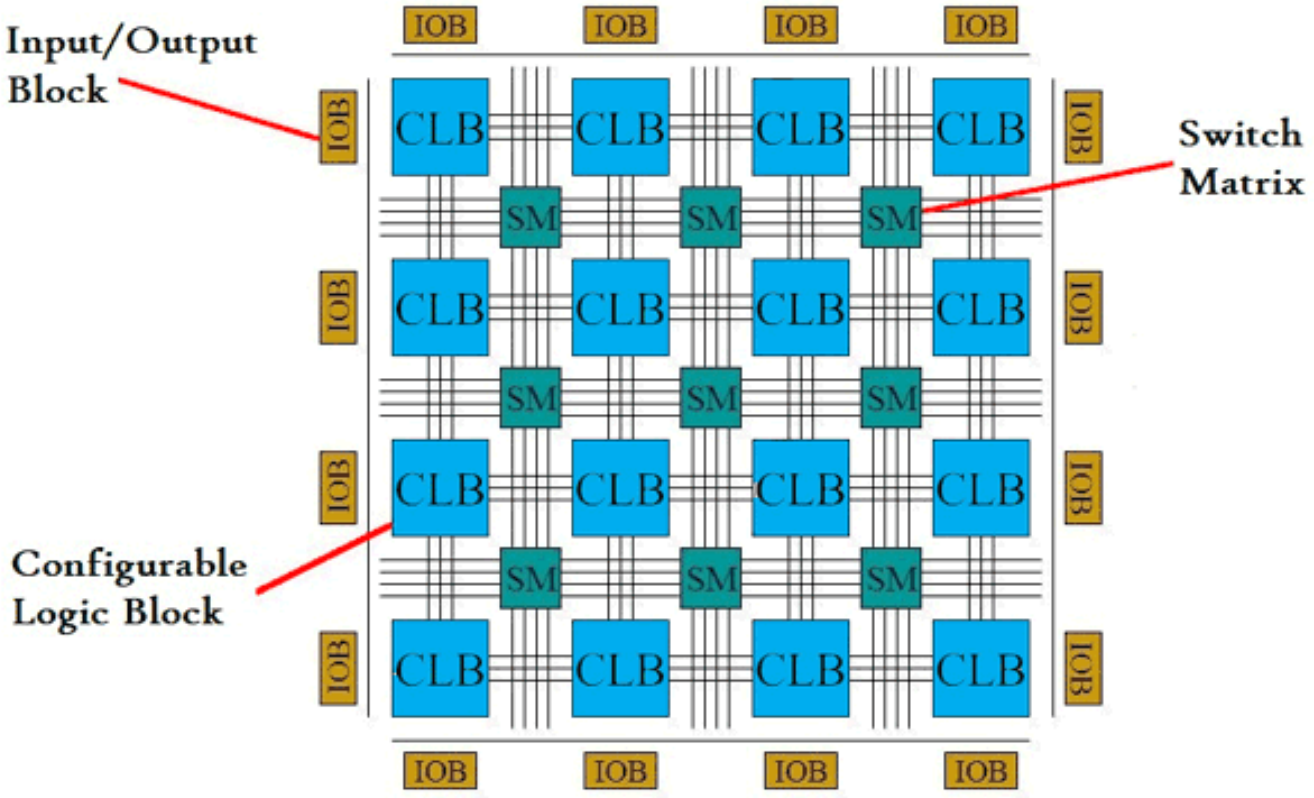
- No standard definition
- ***"Computing via a post-fabrication and spatially programmed connection of processing elements."***

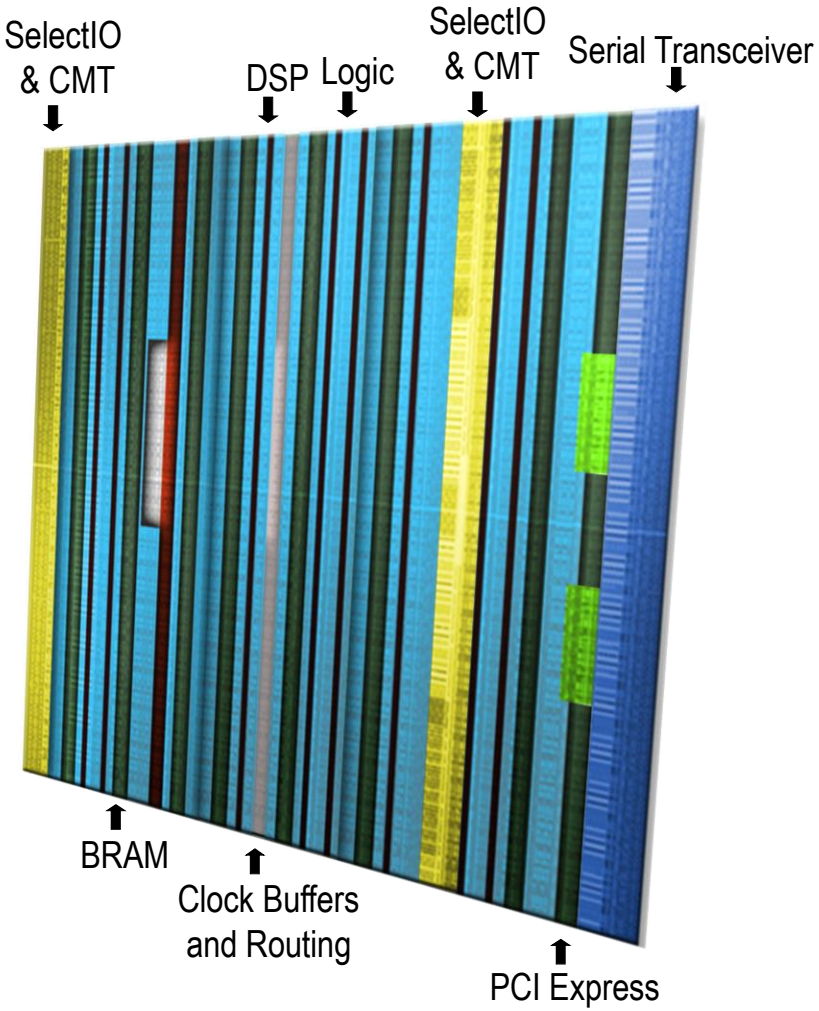
  -John Wawrzynek Sp04
- A computer that can *RE*-configure itself to perform computation spatially as needed
  - How often do we *RE*-configure?
  - Coarse-grain? Fine-grain?
- Example: FPGA

Introduction to FPGA

- Field Programmable Gate Array
- Began as ASIC replacements
  - ASIC that can be configured “in the field”
  - At power up, configuration is load to the chip
  - Chip acts as an ASIC until power down
- Modern FPGA more like computers
  - Exploit *dynamic*, *partial* ***re***configuration
  - Embedded processors
- Xilinx (now part of AMD), Altera (now part of Intel) were 2 major market leaders

FPGA Architecture

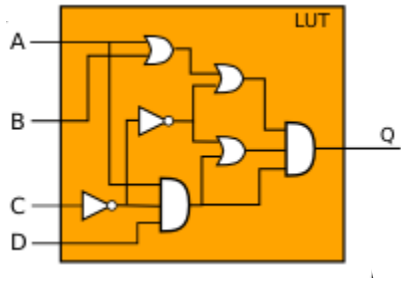




Xilinx Virtex 7

The LUT

- LUT: ***L***ook***u***p ***T***able
- A direct implementation of a truth table
  - Recall a truth table uniquely defines a circuit



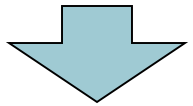
A	B	C	D	Q
0	0	0	0	0
0	0	0	1	0
		...		
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

- An n-LUT implements any n-input combinational logic
  - Depends on LUT configuration

Making a 2-LUT from Truth Table

A	B	FALSE	AND	…	OR	…	NAND	TRUE
0	0	0	0		0		1	1
0	1	0	0		1		1	1
1	0	0	0	…	1	…	1	1
1	1	0	1		1		0	1

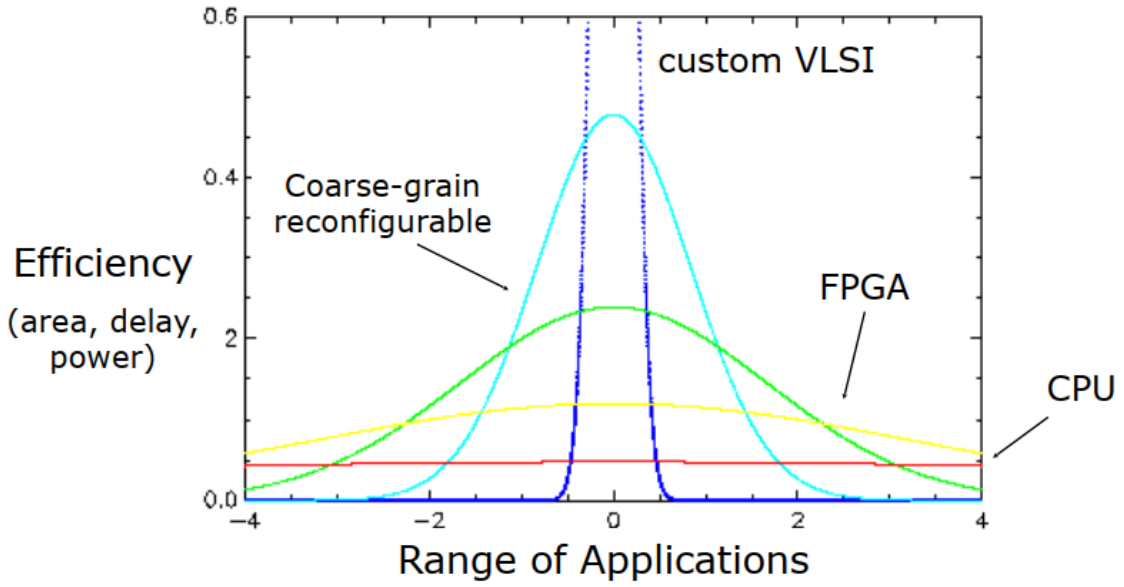
16 possible ways



4 configuration bits

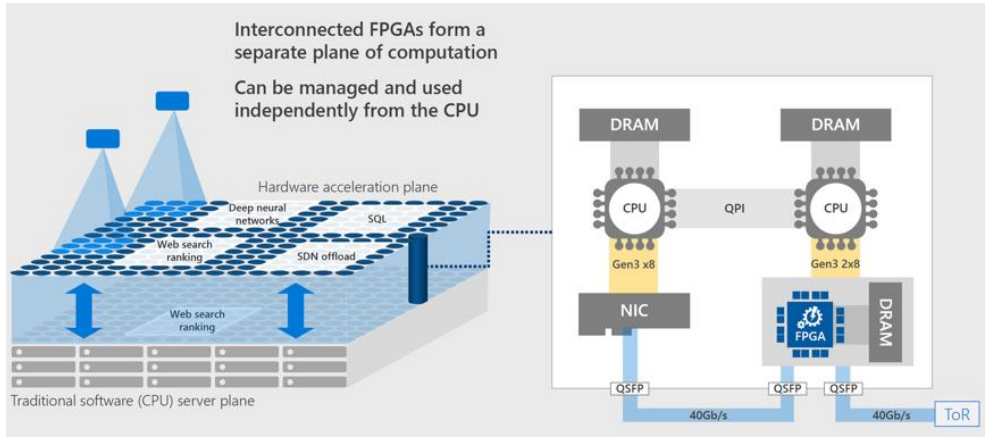
Why are FPGAs Interesting?

- "reconfigurability" or "reprogrammability" may offer other advantages over fixed logic?
  - In-field reprogramming
  - Application specific acceleration
  - Because of the regularity of their design, FPGAs have tracked Moore's Law better
  - FPGAs get higher performance "yield"
  - Customize datapaths/control to match application dataflow - eliminate ISA overhead

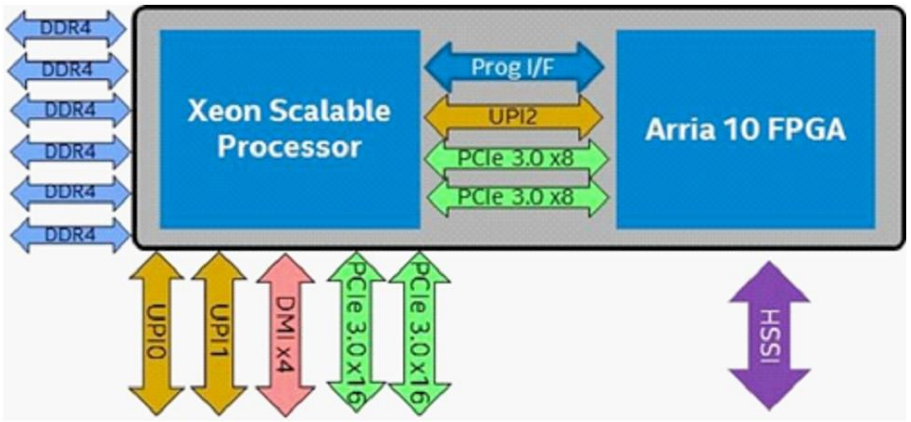


Is Reconfigurable Computing Becoming Mainstream?

AMD+Xilinx



Microsoft Azure



Intel+Altera

Microsoft Goes All in for FPGAs to Build Out AI Cloud

Michael Feldman | September 27, 2016 08:42 CEST

E-mail Tweet Like +1 Share 163

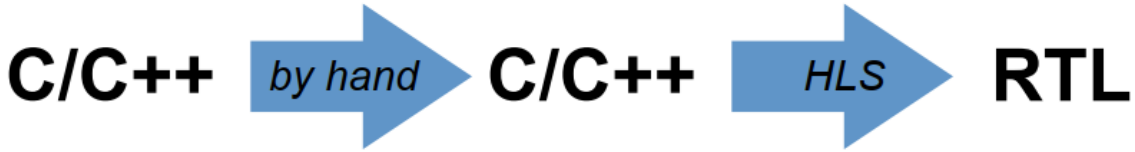
Software giant bets the (server) farm on reconfigurable computing

Microsoft has revealed that Altera FPGAs have been installed across every Azure cloud server, creating what the company is calling "the world's first AI supercomputer." The deployment spans 15 countries and represents an aggregate performance of more than one exa-op. The announcement was made by Microsoft CEO Satya Nadella and engineer Doug Burger during the opening keynote at the Ignite Conference in Atlanta.

The FPGA build-out was the culmination of more than five years of work at Microsoft to find a way to accelerate machine learning and other throughput-demanding applications and services in its Azure cloud. The effort began in earnest in 2011, when the company launched Project Catapult, the R&D initiative to design an acceleration fabric for AI services and applications. The rationale was that CPU evolution, a la Moore's Law, was woefully inadequate in keeping up with the demands of these new hyperscale applications. Just as in traditional high performance computing, multicore CPUs weren't keeping up with demand.

BUT

- Lack of programming model with convenient and effective tools.
- Most successful computing applications using reconfigurable devices involve substantial “hand mapping”. Essentially circuit design.
- Lack of fast compile/debug loop
- What is next then?
  - High level synthesis (HLS)

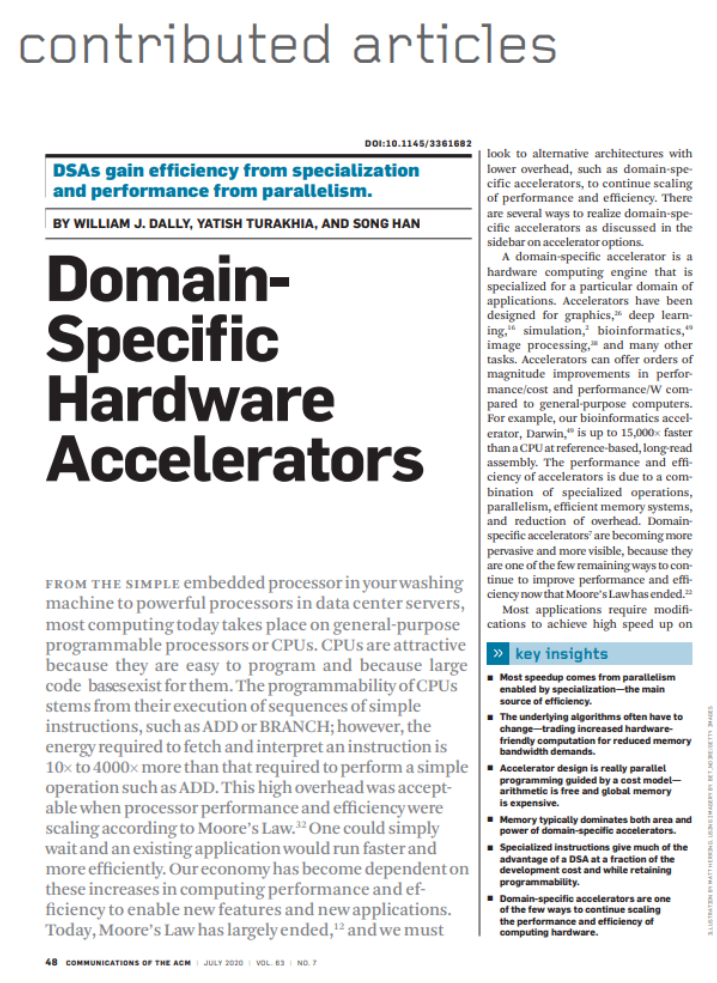


	Micro-processor	GPGPU	FPGA
Programming Models	**A**	**B**	**C**
Fast Compile/Debug	**A**	**B**	**F**
Operating Environment	**A**	**A**	**F**

DOMAIN SPECIFIC ARCHITECTURE (DSA)

Some background

- "…the energy required to fetch and interpret an instruction is 10× to 4000× more than that required to perform a simple operation such as ADD…"
- "A domain-specific accelerator is a hardware computing engine that is specialized for a particular domain of applications."
- "Most applications require modifications to achieve high speed up on domain-specific accelerators."
- "A well-designed accelerator covers the broadest possible space of applications—accelerating a domain rather than a single application."
- "Adding domain-specific instructions to a programmable processor provides the efficiency of the specialized instruction while retaining flexibility."



contributed articles

DOI:10.1145/3361682

DSAs gain efficiency from specialization and performance from parallelism.

BY WILLIAM J. DALLY, YATISH TURAKHIA, AND SONG HAN

Domain-Specific Hardware Accelerators

FROM THE SIMPLE embedded processor in your washing machine to powerful processors in data center servers, most computing today takes place on general-purpose programmable processors or CPUs. CPUs are attractive because they are easy to program and because large code bases exist for them. The programmability of CPUs stems from their execution of sequences of simple instructions, such as ADD or BRANCH; however, the energy required to fetch and interpret an instruction is 10× to 4000× more than that required to perform a simple operation such as ADD. This high overhead was acceptable when processor performance and efficiency were scaling according to Moore's Law.[32] One could simply wait and an existing application would run faster and more efficiently. Our economy has become dependent on these increases in computing performance and efficiency to enable new features and new applications. Today, Moore's Law has largely ended,[12] and we must look to alternative architectures with lower overhead, such as domain-specific accelerators, to continue scaling of performance and efficiency. There are several ways to realize domain-specific accelerators as discussed in the sidebar on accelerator options.

A domain-specific accelerator is a hardware computing engine that is specialized for a particular domain of applications. Accelerators have been designed for graphics,[26] deep learning,[16] simulation,[2] bioinformatics,[49] image processing,[38] and many other tasks. Accelerators can offer orders of magnitude improvements in performance/cost and performance/W compared to general-purpose computers. For example, our bioinformatics accelerator, Darwin,[49] is up to 15,000× faster than a CPU at reference-based, long-read assembly. The performance and efficiency of accelerators is due to a combination of specialized operations, parallelism, efficient memory systems, and reduction of overhead. Domain-specific accelerators' are becoming more pervasive and more visible, because they are one of the few remaining ways to continue to improve performance and efficiency now that Moore's Law has ended.[22]

Most applications require modifications to achieve high speed up on

» key insights

- Most speedup comes from parallelism enabled by specialization—the main source of efficiency.
- The underlying algorithms often have to change—trading increased hardware-friendly computation for reduced memory bandwidth demands.
- Accelerator design is really parallel programming guided by a cost model—arithmetic is free and global memory is expensive.
- Memory typically dominates both area and power of domain-specific accelerators.
- Specialized instructions give much of the advantage of a DSA at a fraction of the development cost and while retaining programmability.
- Domain-specific accelerators are one of the few ways to continue scaling the performance and efficiency of computing hardware.

ILLUSTRATION BY MATT HERRING, USING IMAGERY BY KEY_NOIR/GETTY IMAGES

48 COMMUNICATIONS OF THE ACM | JULY 2020 | VOL. 63 | NO. 7

https://dl.acm.org/doi/pdf/10.1145/3361682

The Age of "Domain - Specific Processors*"

(*): term coined by David Patterson

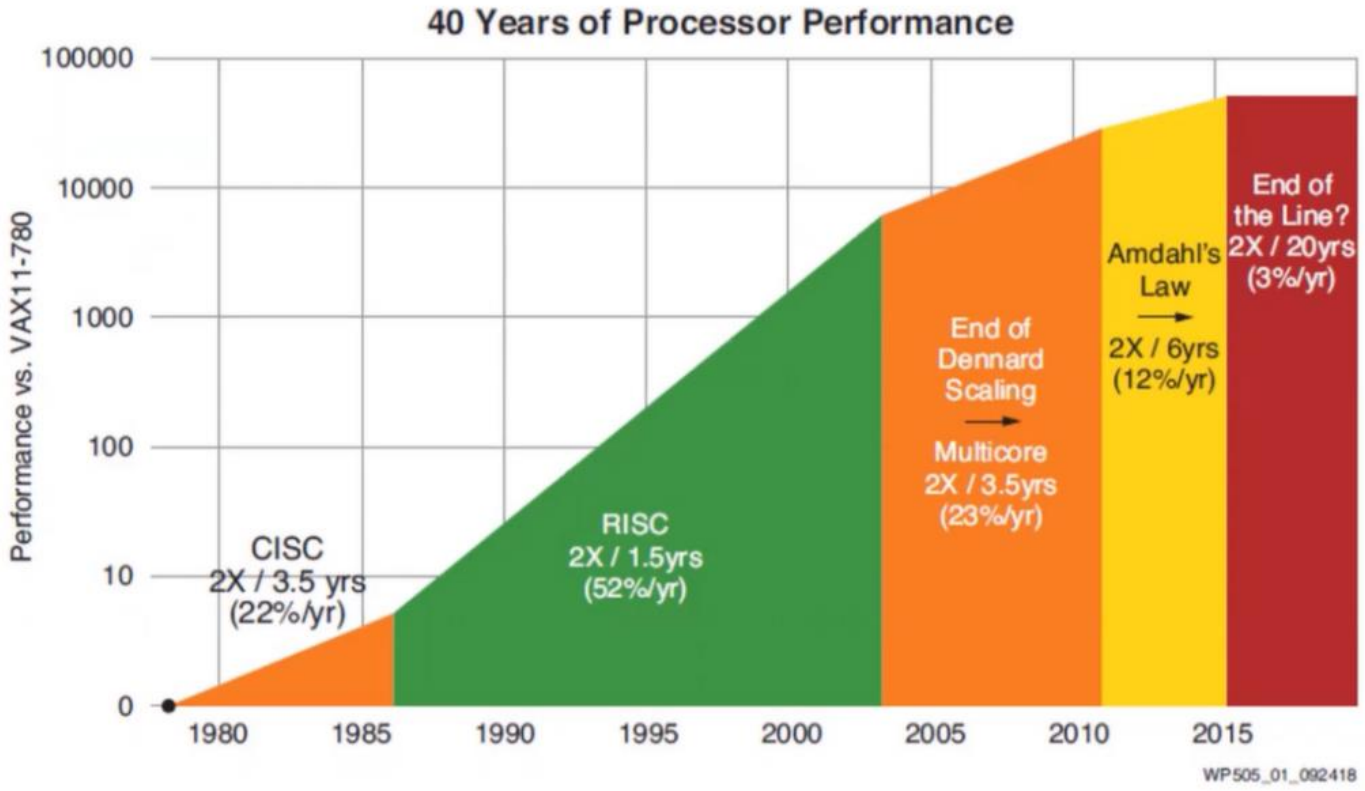


Figure 1: **Processor Performance vs. Time**

Source: Xilinx Whitepaper 505 – Versal ACAP
(reference to J. Hennessy, D. Patterson, *Computer Architecture: A Quantitative Approach* (6th Edition, 2019))

- 1995 – 2003:

  Performance due to process technology (higher frequency, but same power consumption) and memory access optimization (Dennard Scaling)

- 2003: Multicore

  Coming to its limits for tasks that cannot be parallelized (Amdahl's Law)

- 2015: Domain-Specific Processors

  Moving to heterogenous multicore designs

DSA everywhere!

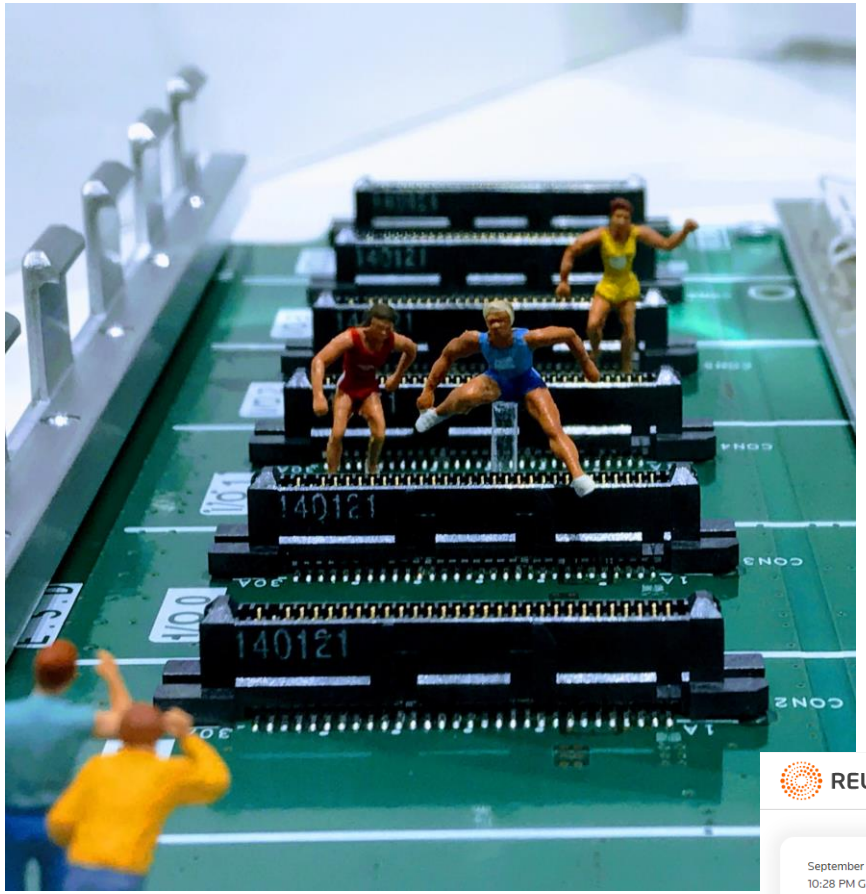


Figure: unsplash

网易首页 > 网易号 > 正文

抖音：造芯！

2022-07-15 00:57:02 来源: 半导体圈 广东

重磅！百度宣布自研的第二代百度昆仑AI芯片实现量产

2021-08-19 09:39 OFweek人工智能网

腾讯“走芯”！首次披露三款自研芯片研发进展...国内互联网巨头皆要造“芯”，进展如何?

2021-11-04 08:36 来源：证券时报 作者：吴永芳

据阿里、百度之后，腾讯也开始涉足芯片领域。

Google claims its new TPUs are 2.7 times faster than the previous generation

REUTERS® World Business Legal Markets Breakingviews Technology Investiga

September 9, 2021
10:28 PM GMT+8
Last Updated 10 months ago

Disrupted

Facebook developing machine learning chip - The Information

Reuters

Domain-specific Architectures

- CPU is good, but not the best, for
  - …many many applications
- Need factor of 100 improvements in number of operations per instruction
  - Requires domain specific architectures
  - For ASICs, costs cannot be amortized over large volumes
  - FPGAs are less efficient than ASICs

Guidelines for DSAs

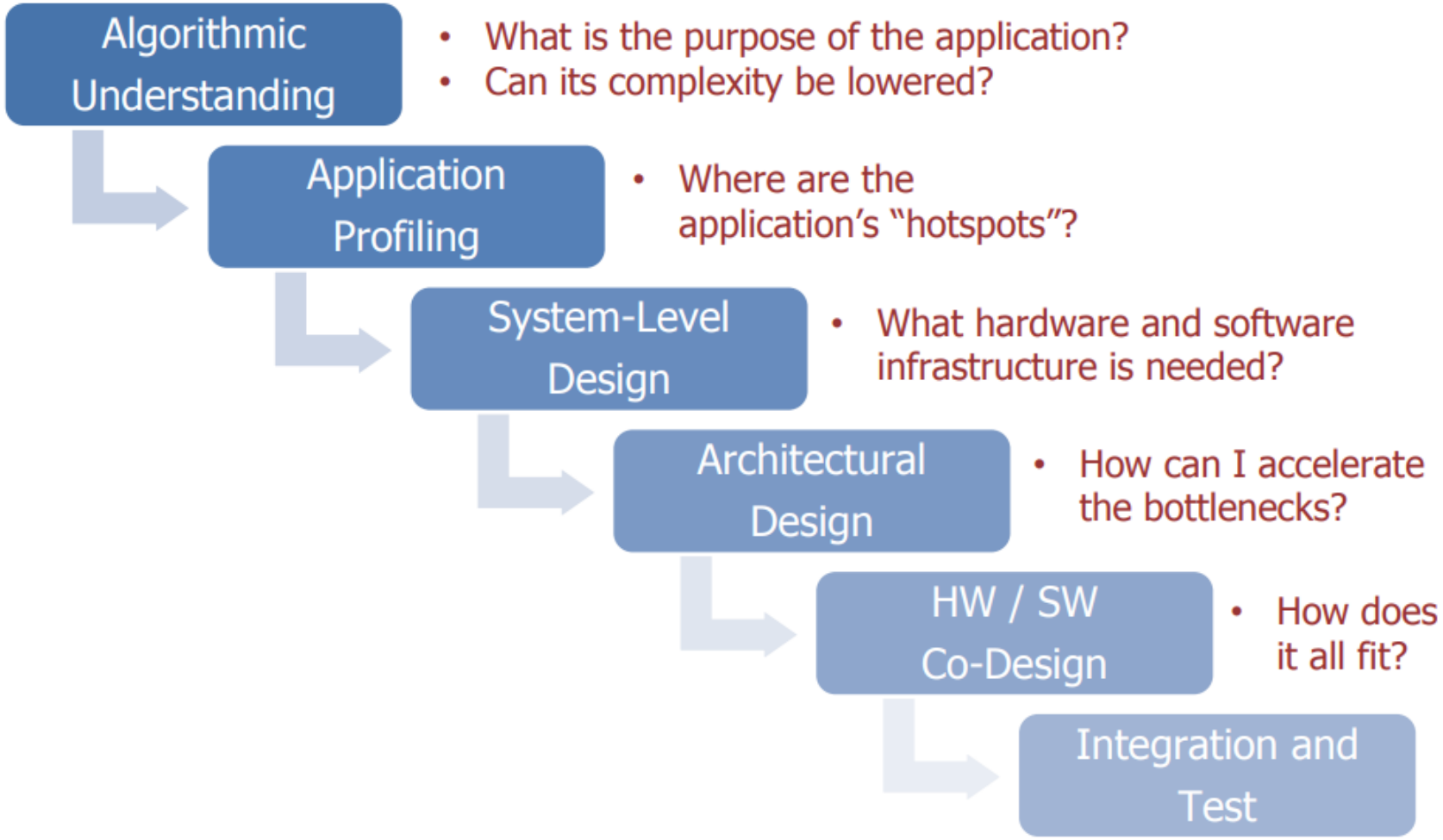
- Use dedicated memories to minimize data movement
  - **Hardware-controlled multi-level cache -> domain-specific software controlled scratch-pad**
- Invest resources into more arithmetic units or bigger memories
  - **Core optimization (OoO, speculation, threading, etc) -> more domain-specific FU/memory**
- Use the easiest form of parallelism that matches the domain
  - **MIMD -> SIMD or VLIW that matches domain**
- Reduce data size and type to the simplest needed for the domain
  - **General-purpose 32/64 integer/float -> domain-specific 8/16 int/float**
- Use a domain-specific programming language
  - **General-purpose C/C++/Fortran -> Domain-specific language (TensorFlow for DNN)**

Guidelines for DSAs

Guideline	TPU	Catapult	Crest	Pixel Visual Core
Design target	Data center ASIC	Data center FPGA	Data center ASIC	PMD ASIC/SOC IP
1. Dedicated memories	24 MiB Unified Buffer, 4 MiB Accumulators	Varies	N.A.	Per core: 128 KiB line buffer, 64 KiB P.E. memory
2. Larger arithmetic unit	65,536 Multiply-accumulators	Varies	N.A.	Per core: 256 Multiply-accumulators (512 ALUs)
3. Easy parallelism	Single-threaded, SIMD, in-order	SIMD, MISD	N.A.	MPMD, SIMD, VLIW
4. Smaller data size	8-Bit, 16-bit integer	8-Bit, 16-bit integer 32-bit Fl. Pt.	21-bit Fl. Pt.	8-bit, 16-bit, 32-bit integer
5. Domain-specific lang.	TensorFlow	Verilog	TensorFlow	Halide/TensorFlow

Figure 7.3 The four DSAs in this chapter and how closely they followed the five guidelines. Pixel Visual Core typically has 2–16 cores. The first implementation of Pixel Visual Core does not support 8-bit arithmetic.

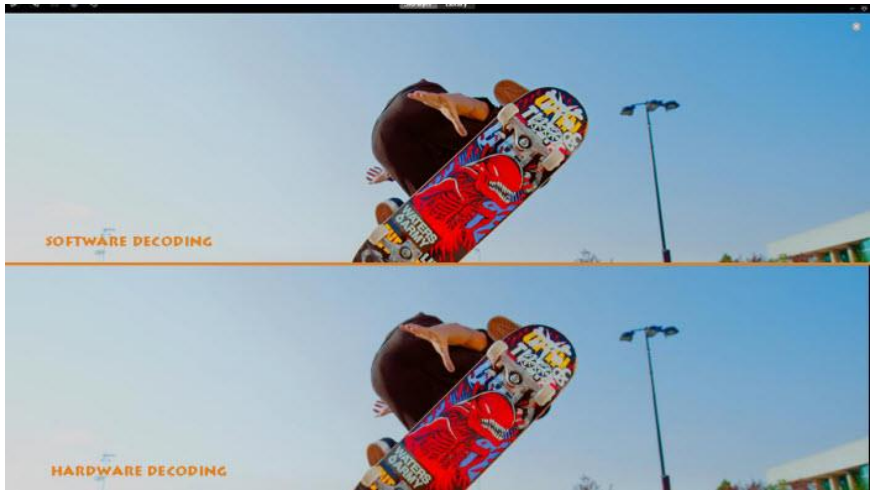
Typical Application Acceleration Methodology



source: http://class.ece.iastate.edu/cpre488/lectures/Lect-08.pdf

Hardware vs Software Acceleration

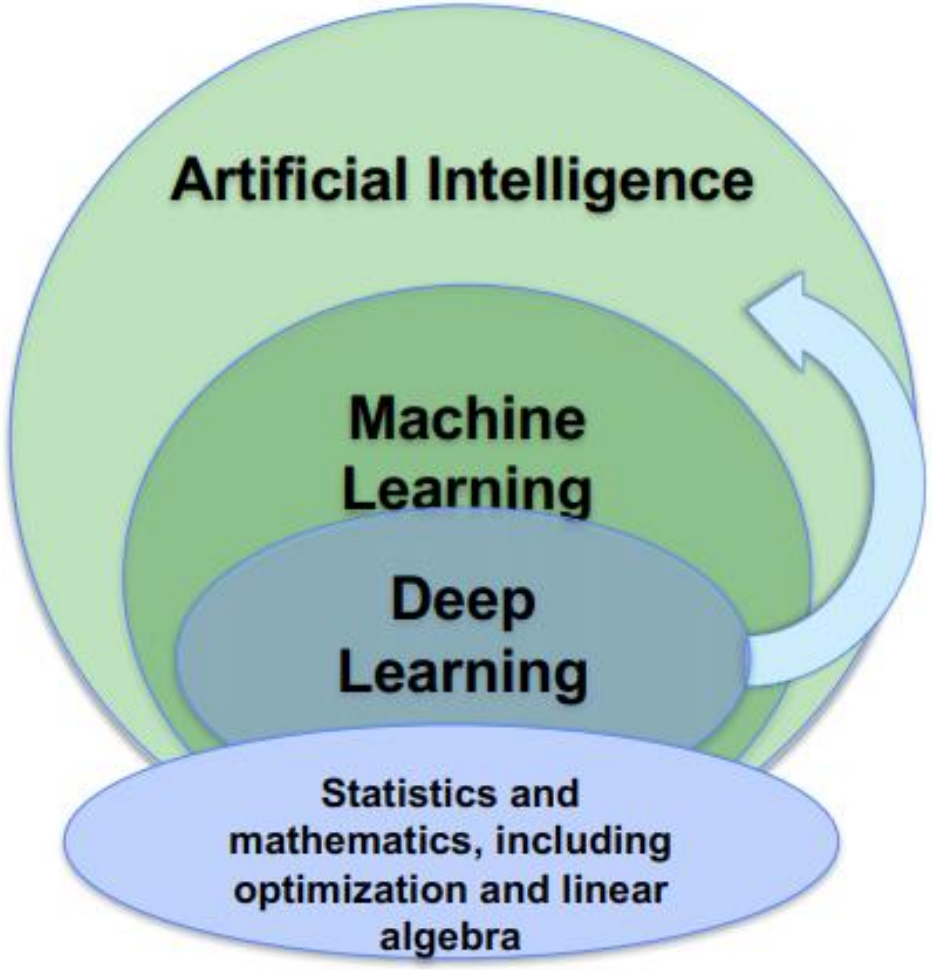
- An example: 4K HDR Video (Sample-3) Decoding



Conclusion: hardware acceleration helps decode and render videos without occupying much CPU without scarifying quality

source: https://www.5kplayer.com/video-music-player/paper-hardware-acceleration-vs-software-decoding.htm

Artificial Intelligence, Machine Learning and Deep Learning



Big Data Processing

Sophisticated Algorithms

High Performance Machines

Slide Credit: Prof. Yonghong Yan @ University of South Carolina

Example: Deep Neural Networks

- Inpired by neuron of the brain
- Computes non-linear “activiation” function of the weighted sum of input values
- Neurons arranged in layers

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

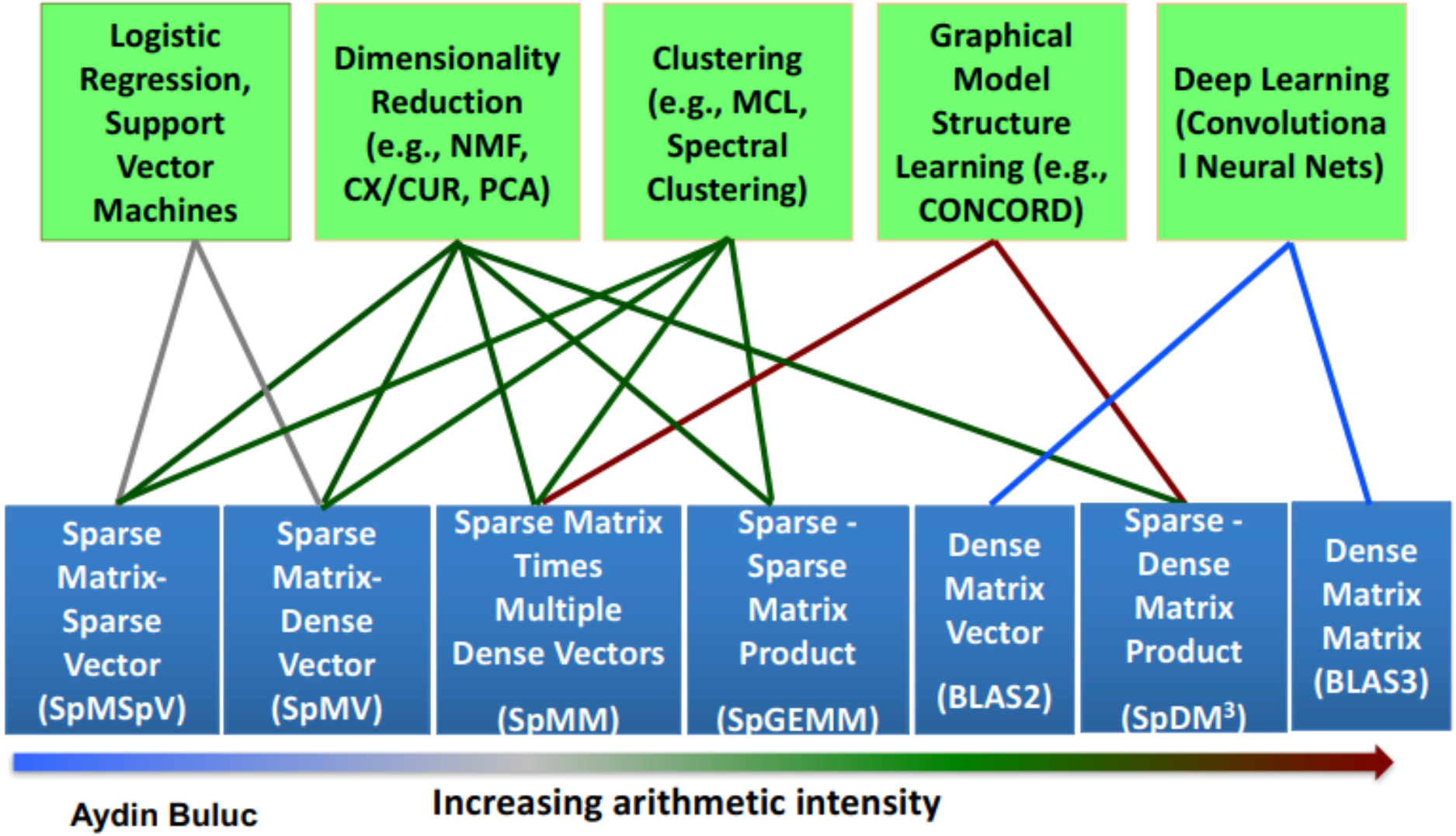
Example: Deep Neural Networks

- Most practioners will choose an existing design
  - Topology
  - Data type
- Training (learning):
  - Calculate weights using backpropagation algorithm
  - Supervised learning: stochastic graduate descent

Type of data	Problem area	Size of benchmark's training set	DNN architecture	Hardware	Training time
text [1]	Word prediction (word2vec)	100 billion words (Wikipedia)	2-layer skip gram	1 NVIDIA Titan X GPU	6.2 hours
audio [2]	Speech recognition	2000 hours (Fisher Corpus)	11-layer RNN	1 NVIDIA K1200 GPU	3.5 days
images [3]	Image classification	1 million images (ImageNet)	22-layer CNN	1 NVIDIA K20 GPU	3 weeks
video [4]	activity recognition	1 million videos (Sports-1M)	8-layer CNN	10 NVIDIA GPUs	1 month

- Inferrence: use neural network for classification

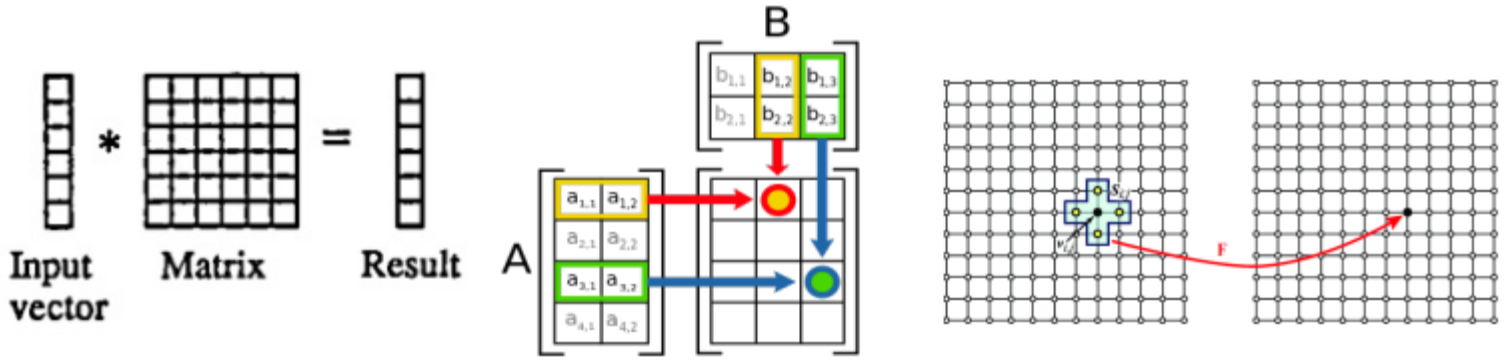
Machine Learning Mapping to Linear Algebra



Slide Credit: Prof. Yonghong Yan @ University of South Carolina

Quick Summary

- Need high-efficient (performance and power) implementation for dense matrix operations
  - Matrix-vector, matrix-matrix multiplication, and stencil



- **Other non-linear functions**
  - ReLU, Sigmoid, tanh, etc

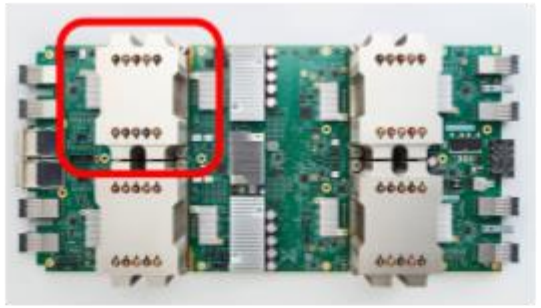
Google Tensor Processing Units (TPUs)

- Custom-designed machine learning ASIC that powers Google products like Alpha Go, Translate, Photos, Search, Assistant, and Gmail.
- Heart: 256 x 256 8-bit matrix multiply-add unit, Large software-managed scratchpad

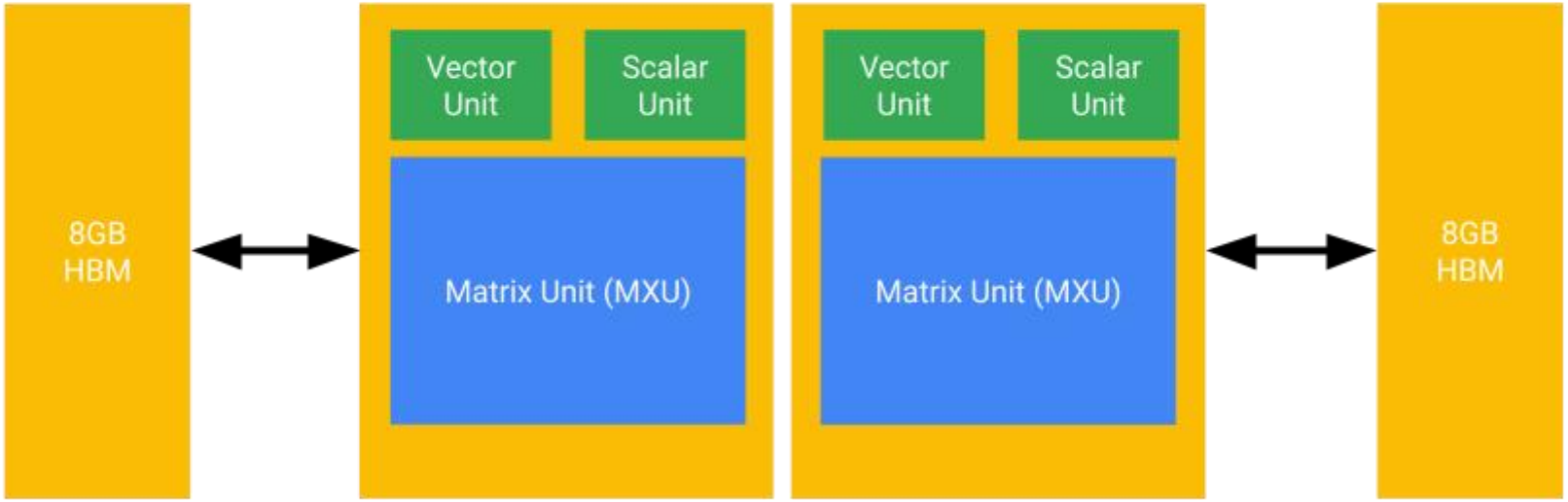
Source: Google

Example: Google Tensor Processing Units (TPUs)

TPU v2 processor layout

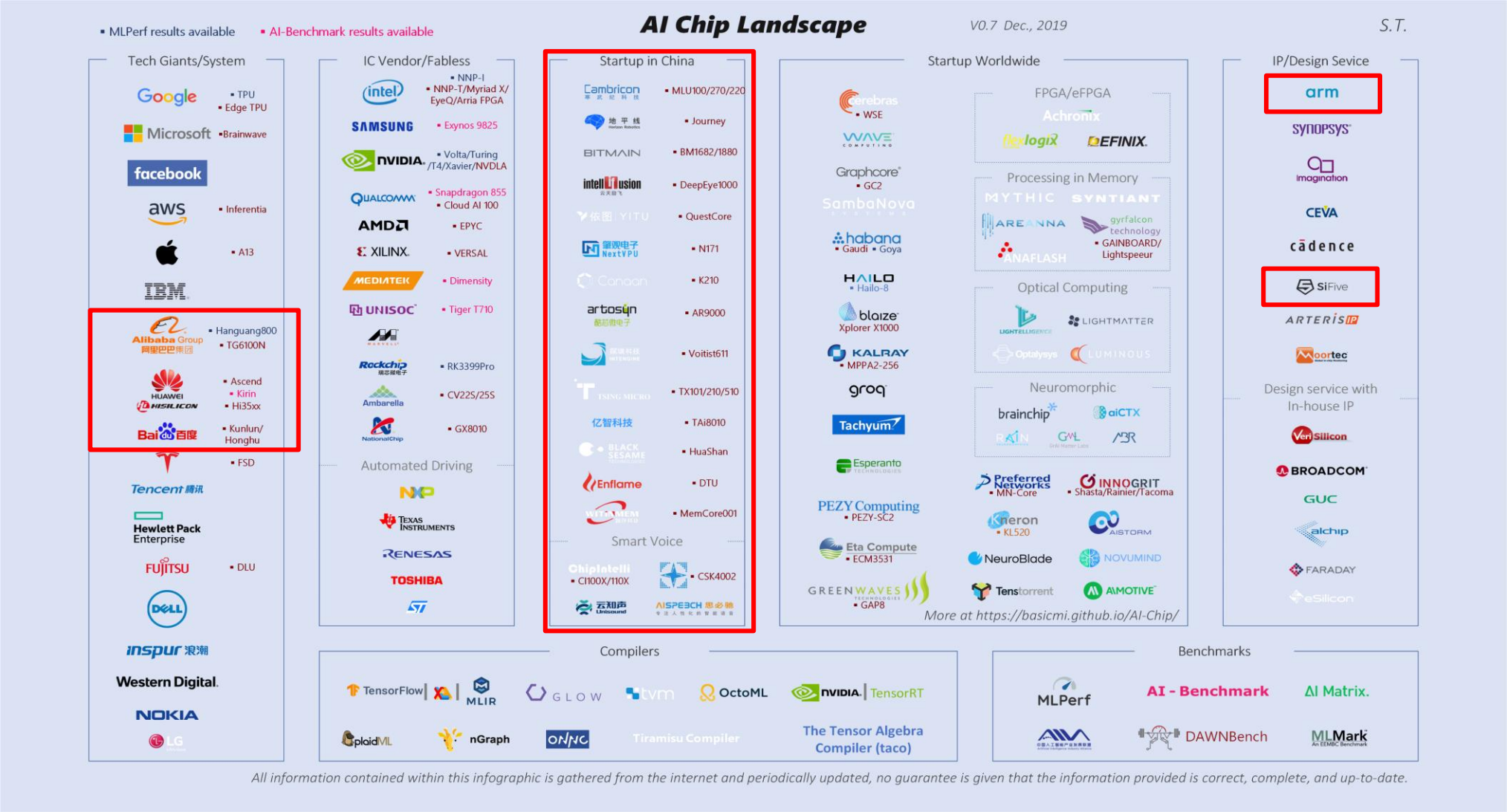


- Neural network accelerator
- Fast matrix multiply machine



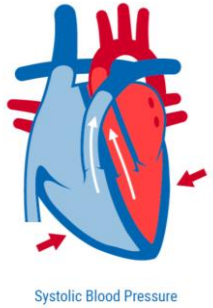
Source: Google

AI Chip Landscape (since TPU)

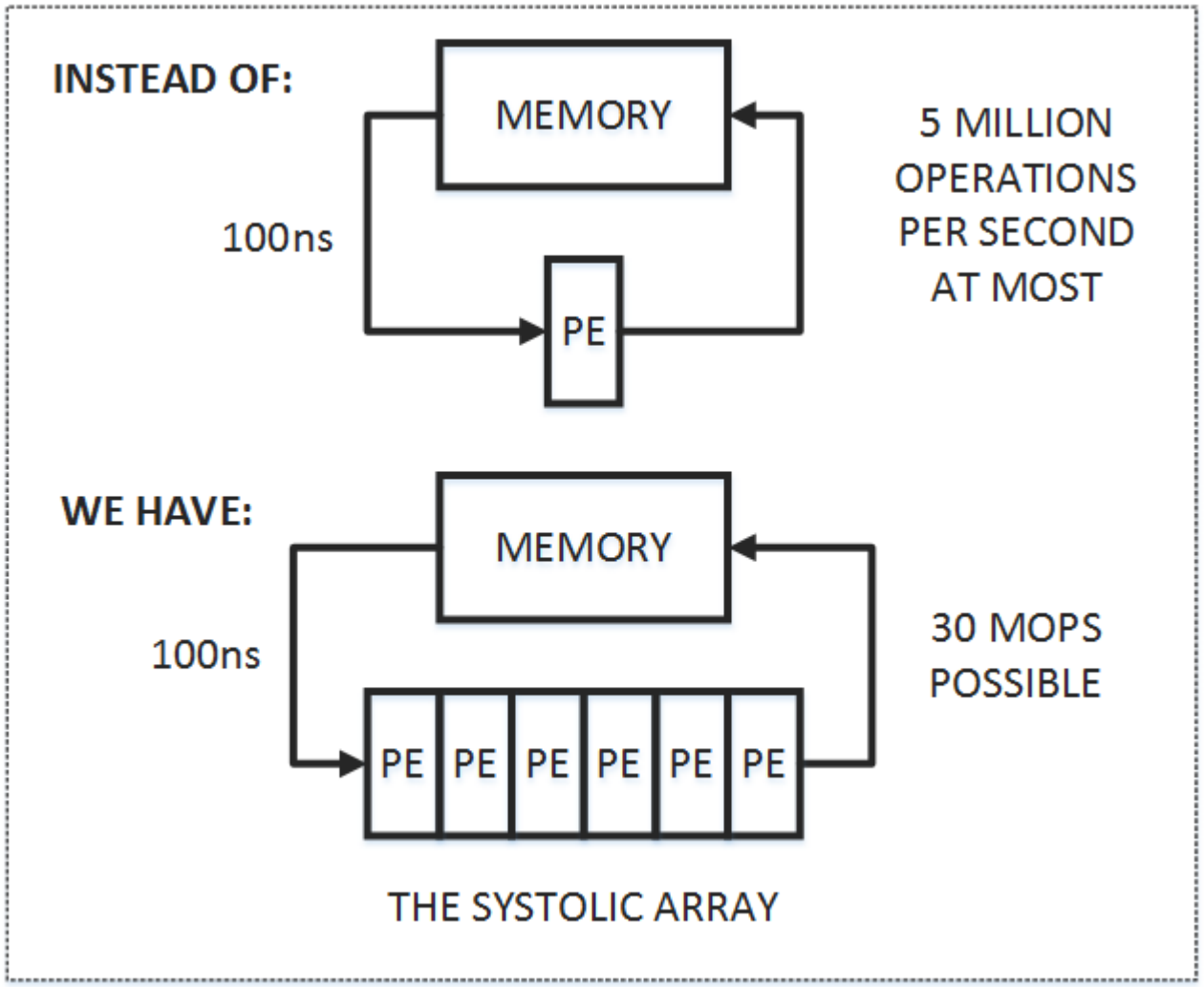


Source: https://github.com/basicmi/AI-Chip

Systolic Array



Memory: Heart
PEs: Cell



- Goal: design an accelerator that has
  - Simple, regular design (keep # unique parts small and regular)
  - High concurrency → high performance
  - Balanced computation and I/O (memory) bandwidth
- Idea: Replace a single processing element (PE) with a regular array of PEs and carefully orchestrate flow of data between the PEs
  - such that they collectively transform a piece of input data before outputting it to memory
- Benefit: Maximizes computation done on a single piece of data element brought from memory

Systolic Computation Example

- Convolution
  - Used in filtering, pattern matching, correlation, polynomial evaluation, etc …
  - Many image processing tasks

Given the sequence of weights $\{w_1, w_2, \ldots, w_k\}$
and the input sequence $\{x_1, x_2, \ldots, x_n\}$,

compute the result sequence $\{y_1, y_2, \ldots, y_{n+1-k}\}$
defined by

$$y_i = w_1 x_i + w_2 x_{i+1} + \ldots + w_k x_{i+k-1}$$

Systolic Computation Example: Convolution

- y1 = w1x1 + w2x2 + w3x3
- y2 = w1x2 + w2x3 + w3x4
- y3 = w1x3 + w2x4 + w3x5

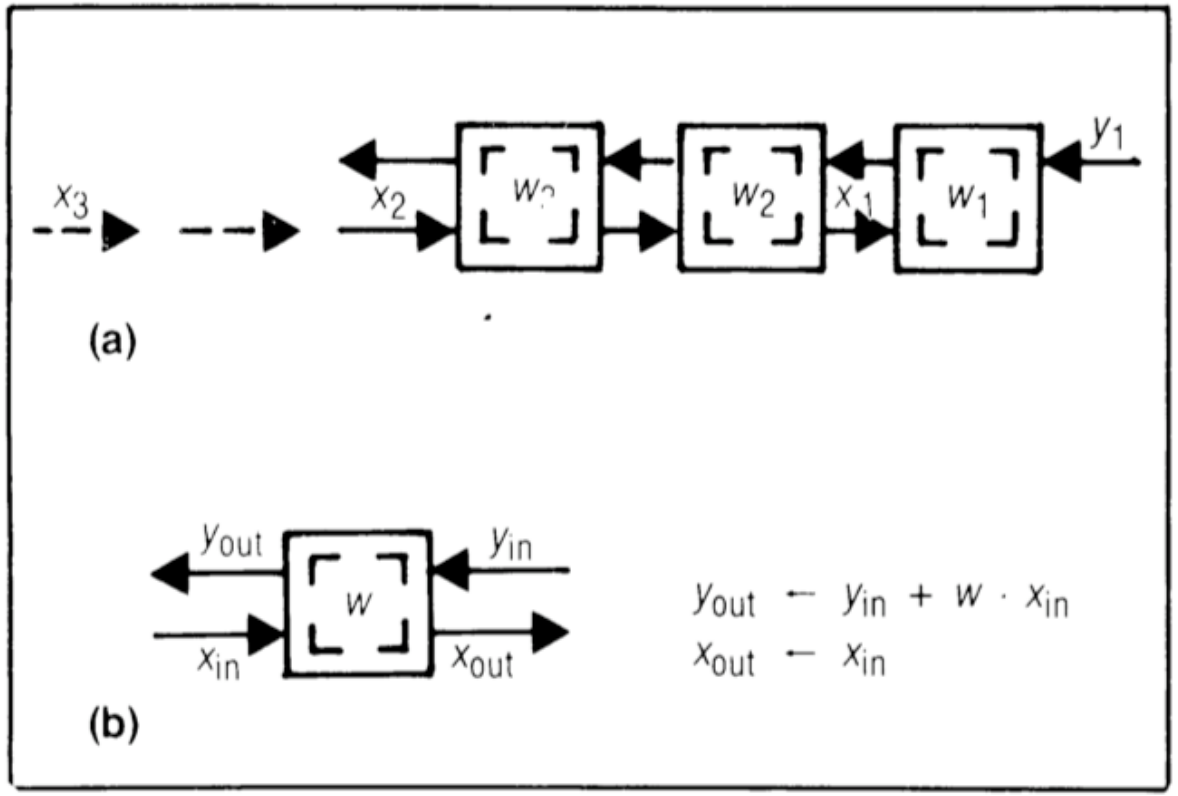


Figure 8. Design W1: systolic convolution array (a) and cell (b) where w_i's stay and x_i's and y_i's move systolically in opposite directions.

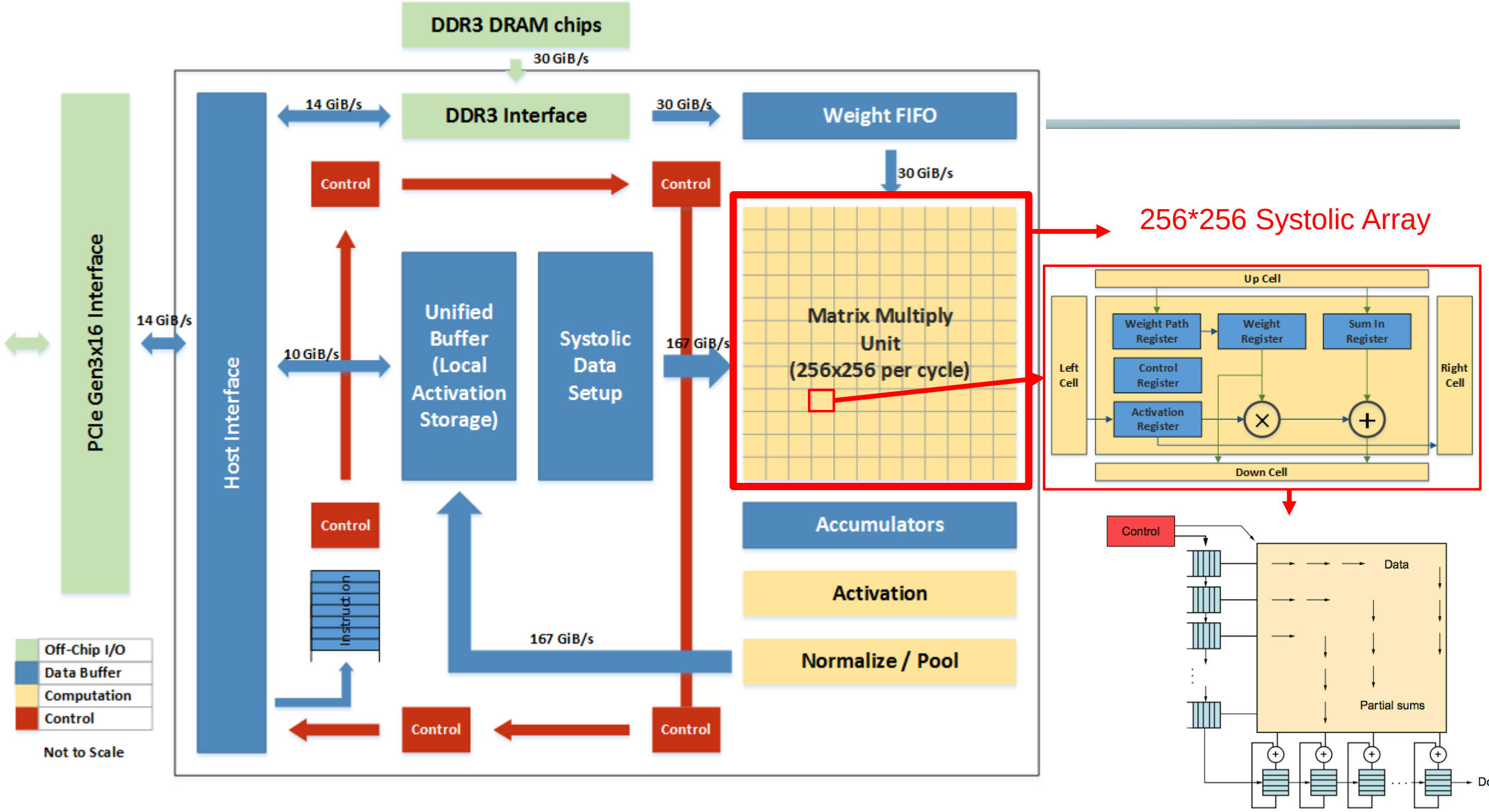


Figure 7.14 Systolic data flow of the Matrix Multiply Unit.

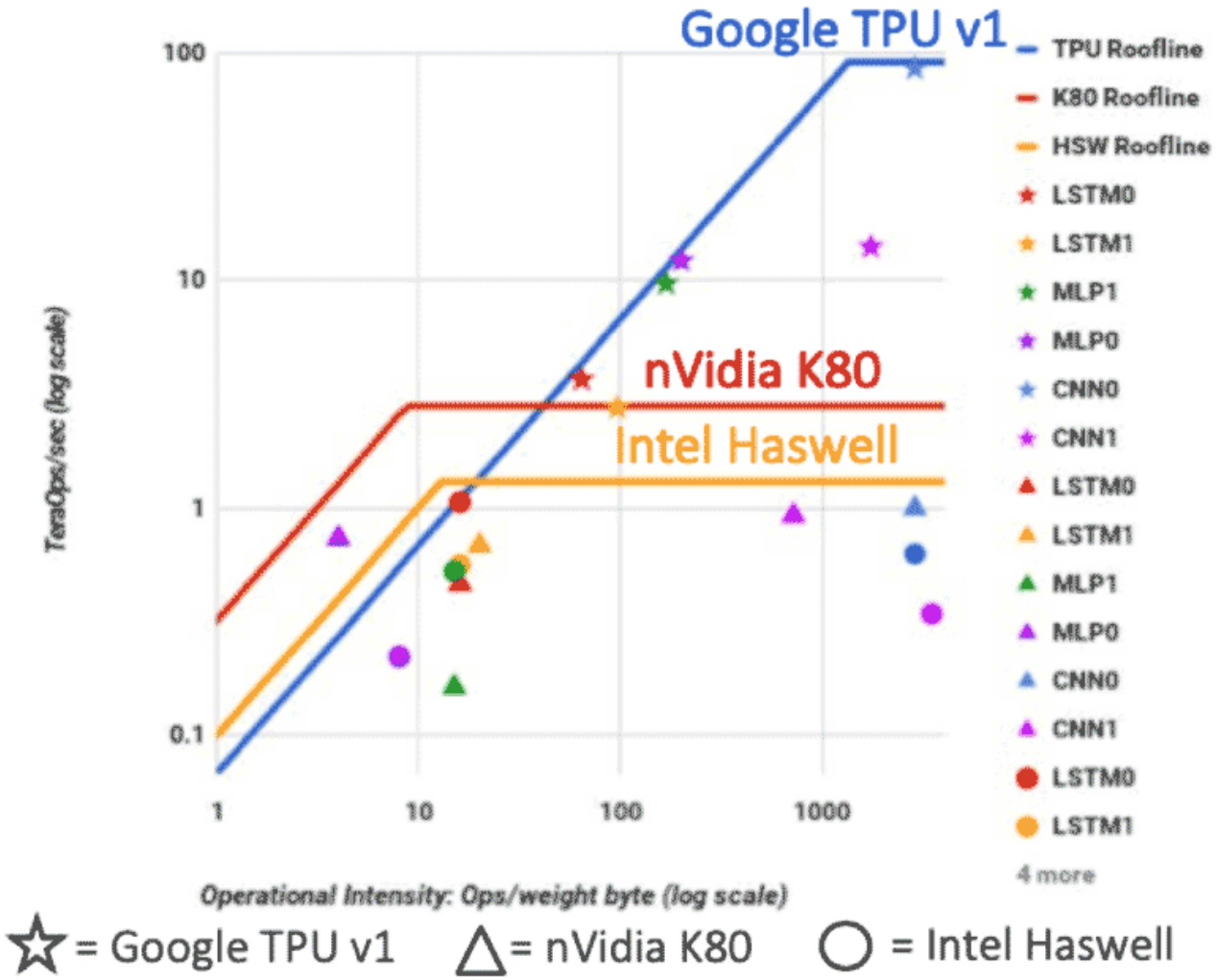
TPU Details

- TPU was designed to be a coprocessor on the PCIe I/O bus
  - **Plugged into existing servers and simplify hardware design and debugging,**
- Host server sends instructions over the PCIe bus directly to the TPU I-buffer for it to execute
  - **TPU is closer in spirit to an FPU (floating-point unit) coprocessor than it is to a GPU, which fetches instructions from its memory.**
- The internal blocks are typically connected together by 256-bytewide (2048-bits) paths.
- Matrix Multiply Unit contains 256x256 ALUs that can perform 8-bit multiply-and-adds on signed or unsigned integers.
  - **The 16-bit products are collected in the 4 MiB of 32-bit Accumulators below the matrix unit.**
  - **It reads and writes 256 values per clock cycle and can perform either a matrix multiply or a convolution. The nonlinear functions are calculated by the Activation hardware.**
- The weights are staged through an on-chip Weight FIFO that reads from an off-chip 8 GiB DRAM called Weight Memory (for inference, weights are read-only;
- The intermediate results are held in the 24 MiB on-chip Unified Buffer, which can serve as inputs to the Matrix Multiply Unit.
- A programmable DMA controller transfers data to or from CPU Host memory and the Unified Buffer.

TPU ISA

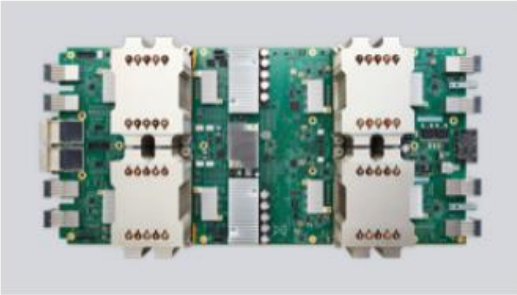
- TPU is CISC tradition, CPI are typically 10-20
- No program counter, no branch instructions
- About a dozen instructions, five key ones:
  - `Read_Host_Memory`: Reads data from the CPU memory into the unified buffer
  - `Read_Weights`: Reads weights from the Weight Memory into the Weight FIFO as input to the Matrix Unit
  - `MatrixMatrixMultiply/Convolve`: Perform a MM multiply, a MV multiply, an element-wise MM, an element-wise MV, or a convolution from the Unified Buffer into the accumulators; Takes a variable-sized B*256 input, multiplies it by a 256x256 constant input, and produces a B*256 output, taking B pipelined cycles to complete
  - `Activate`: Computes activation function, those nonlinear function of the artificial neuron, with options for ReLU, Sigmoid, tanh, and so on. Its inputs are the Accumulators, and its output is the Unified Buffer.
  - `Write_Host_Memory`: Writes data from unified buffer into host memory

TPU Roofline



- Not computation bounded!
- Increasing memory bandwidth (memory) has the biggest impact!

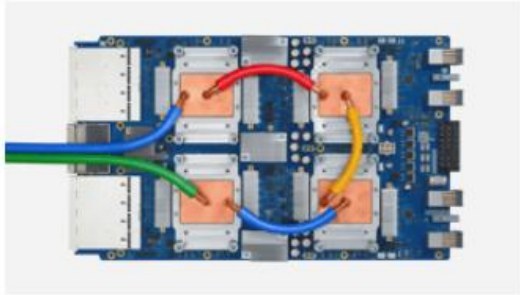
TPU Evolution (from inference to training)



Cloud TPU v2

180 teraflops

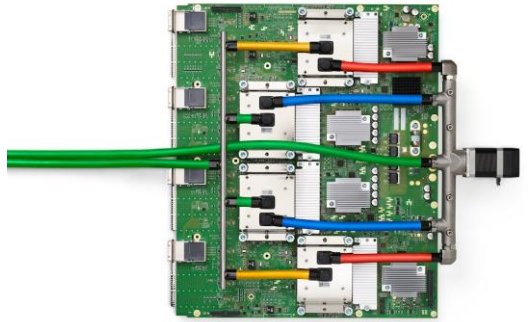
64 GB High Bandwidth Memory (HBM)



Cloud TPU v3

420 teraflops

128 GB HBM



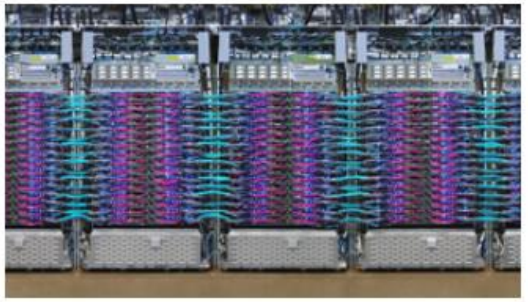
TPU v4

Cloud TPU v2 Pod (beta)

11.5 petaflops

4 TB HBM

2-D toroidal mesh network



Cloud TPU v3 Pod (beta)

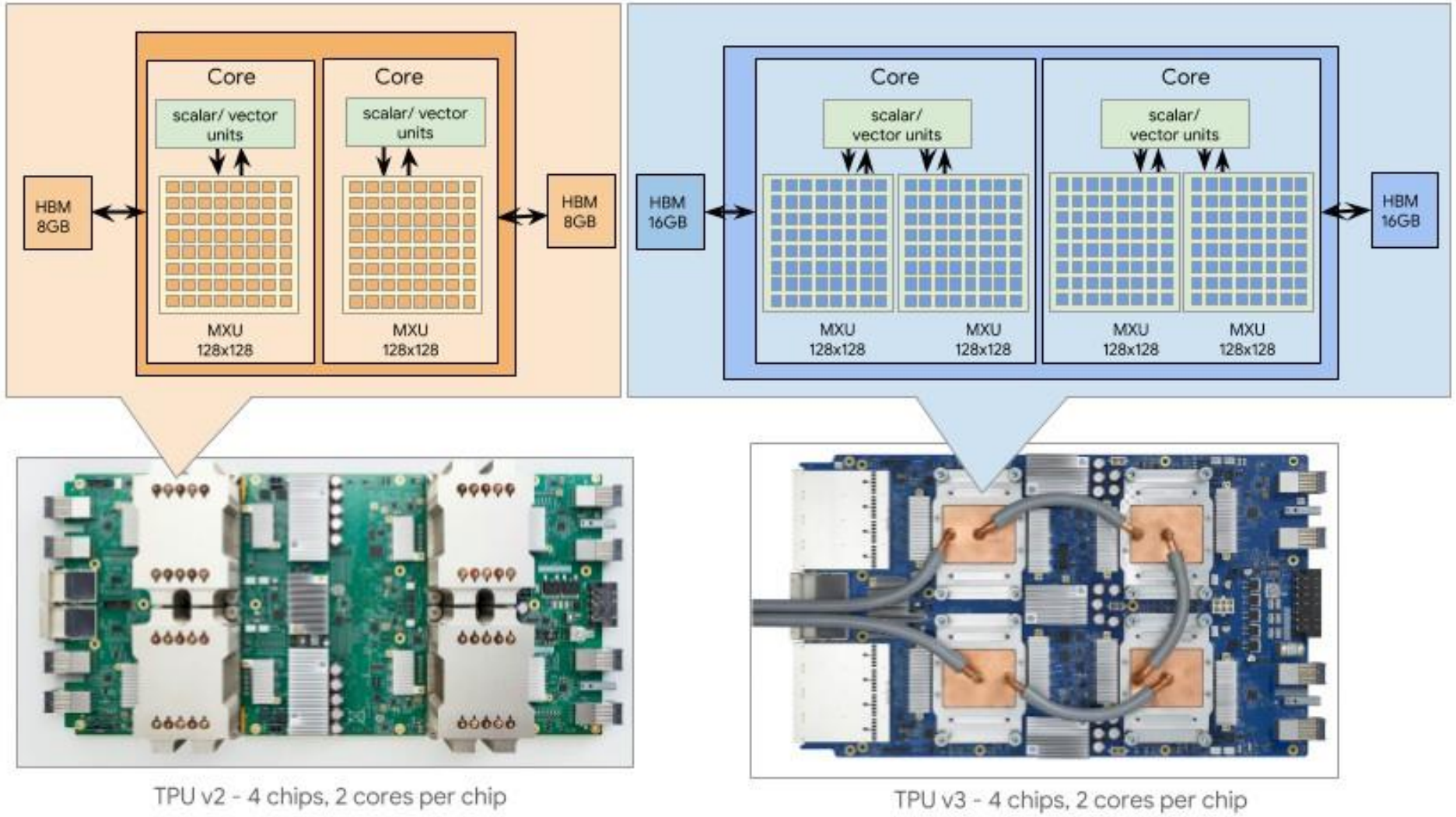
100+ petaflops

32 TB HBM

2-D toroidal mesh network

source: Google

TPU V2 vs. V3



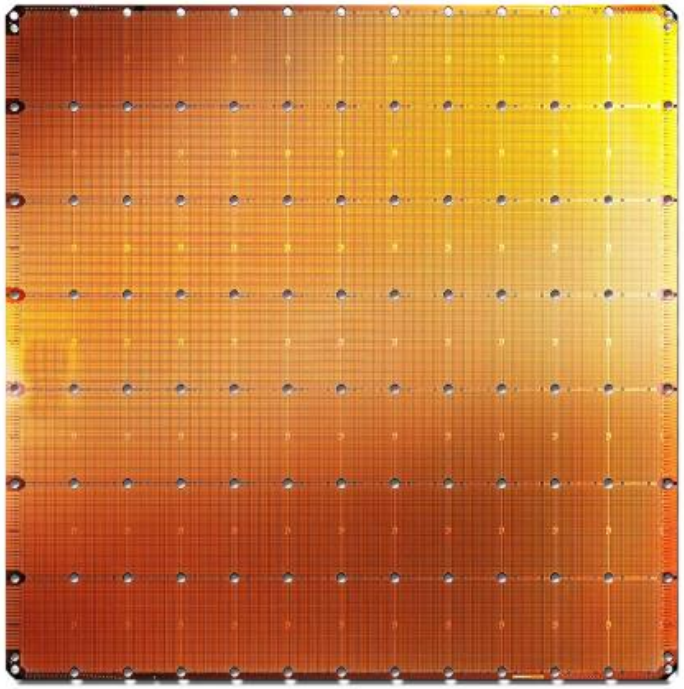
source: Google

TPU Summary

- Use dedicated memories
  - 24 MiB dedicated buffer, 4 MiB accumulator buffers
- Invest resources in arithmetic units and dedicated memories
  - 60% of the memory and 250X the arithmetic units of a server-class CPU
- Use the easiest form of parallelism that matches the domain
  - Exploits 2D SIMD parallelism using systolic array
- Reduce the data size and type needed for the domain
  - Primarily uses 8-bit integers
- Use a domain-specific programming language
  - Uses TensorFlow

OTHER AI ACCELERATORS

Cerebras's Wafer Scale Engine (for training)



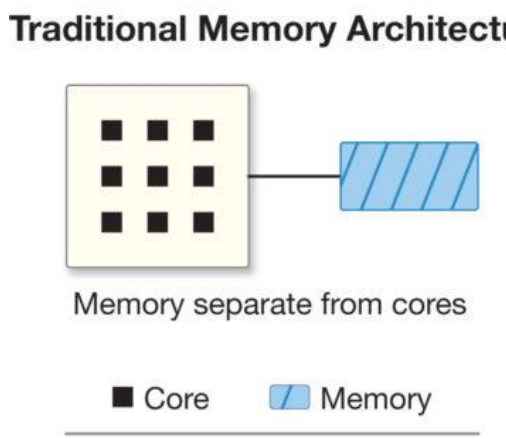
- The largest ML accelerator chip (2019)
- 400,000 cores

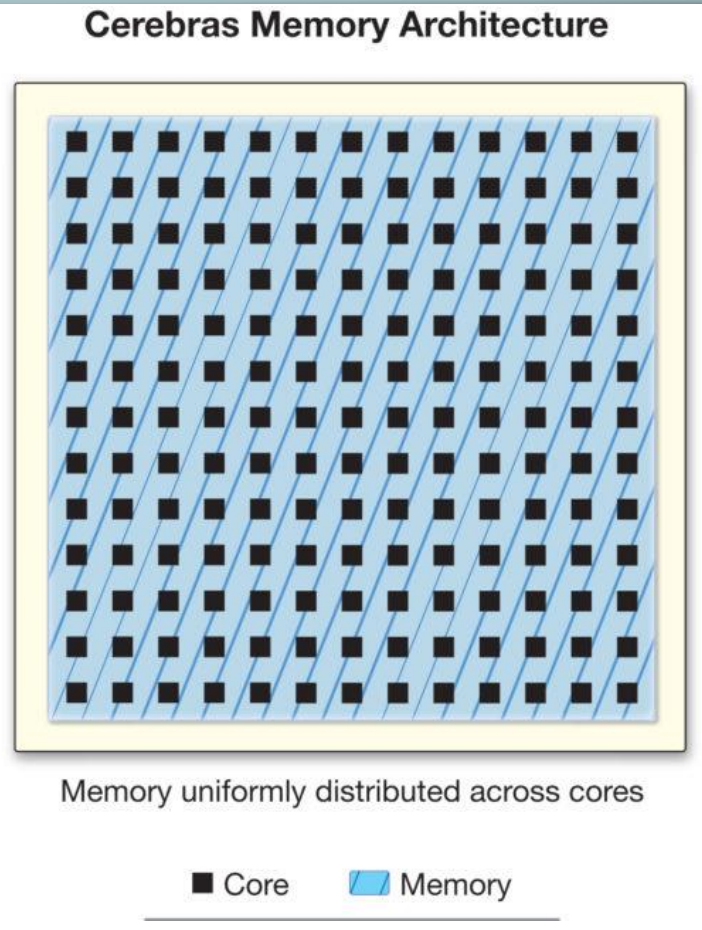
Cerebras WSE
1.2 Trillion transistors
46,225 mm^2

Largest GPU
21.1 Billion transistors
815 mm^2
NVIDIA TITAN V

source: Prof. Onur Mutlu, Digital Design and Computer Architecture
https://www.anandtech.com/show/14758/hot-chips-31-live-blogs-cerebras-wafer-scale-deep-learning

Cerebras's Wafer Scale Engine



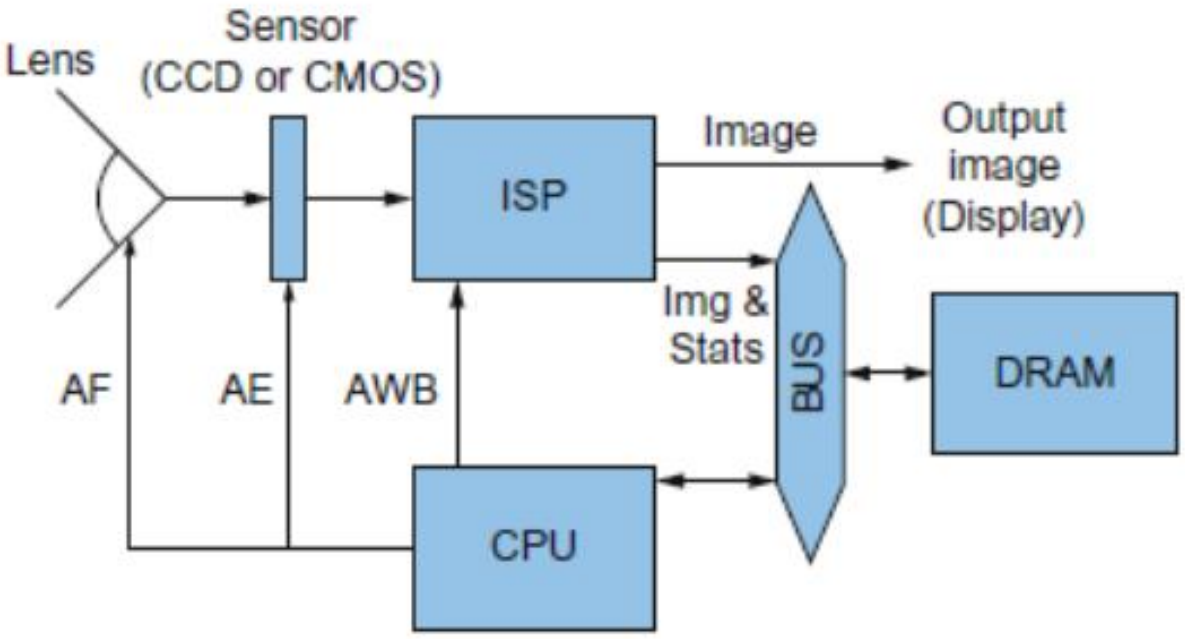


18 Gigabytes of on chip memory and 9.6 Petabytes of memory bandwidth — respectively, 3,000x and 10,000x more than is available on the leading GPU!

https://cerebras.net/blog/cerebras-wafer-scale-engine-why-we-need-big-chips-for-deep-learning/

Pixel Visual Core

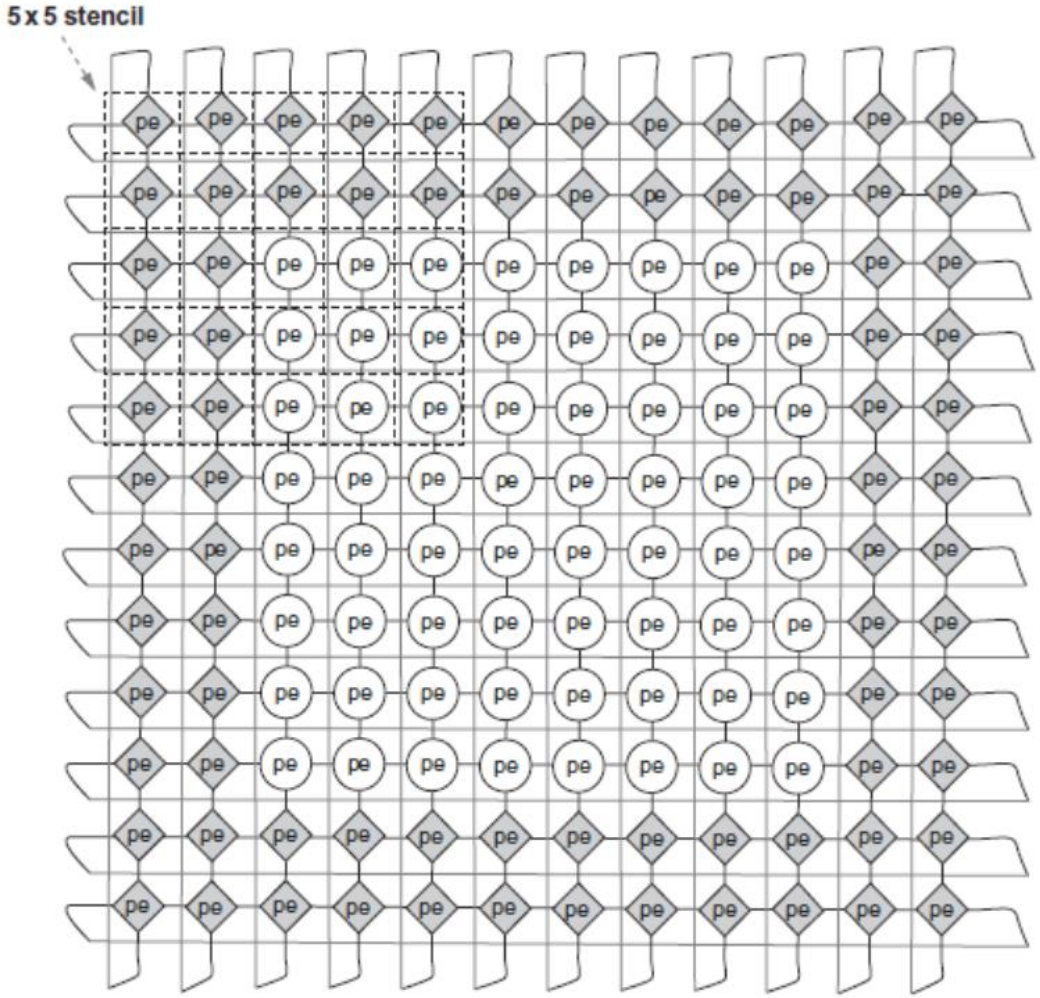
- **Pixel Visual Core**
  - ARM-based system in package (SiP) image processors designed by Google
  - Image Processing Unit (Image Signal Processor, ISP)

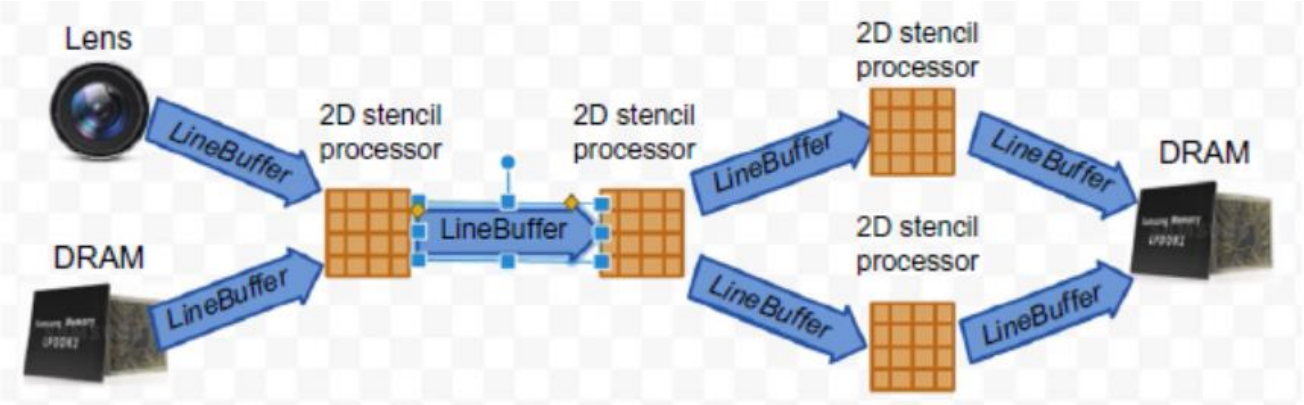


Pixel Visual Core

- Software written in Halide, a DSL
  - Compiled to virtual ISA
  - vISA is lowered to physical ISA using application specific parameters
  - pISA is VLSI
- Optimized for energy
  - Power Budget is 6 to 8 W for bursts of 10-20 seconds, dropping to tens of milliwatts when not in use
  - 8-bit DRAM access equivalent energy as 12,500 8-bit integer operations or 7 to 100 8-bit SRAM accesses
  - IEEE 754 operations require 22X to 150X of the cost of 8-bit integer operations
- Optimized for 2D access
  - 2D SIMD unit
  - On-chip SRAM structured using a square geometry

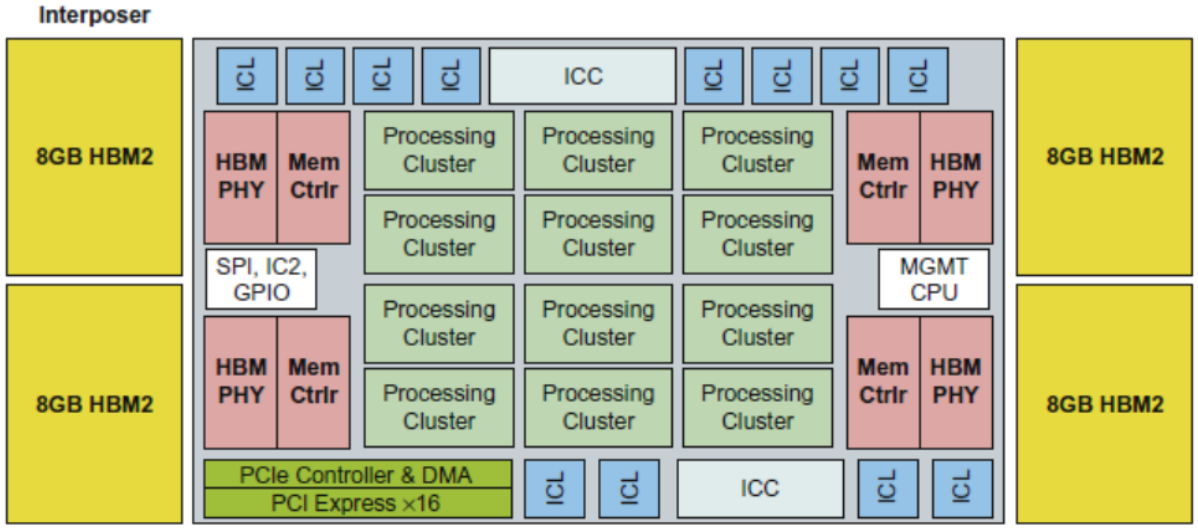
Pixel Visual Core





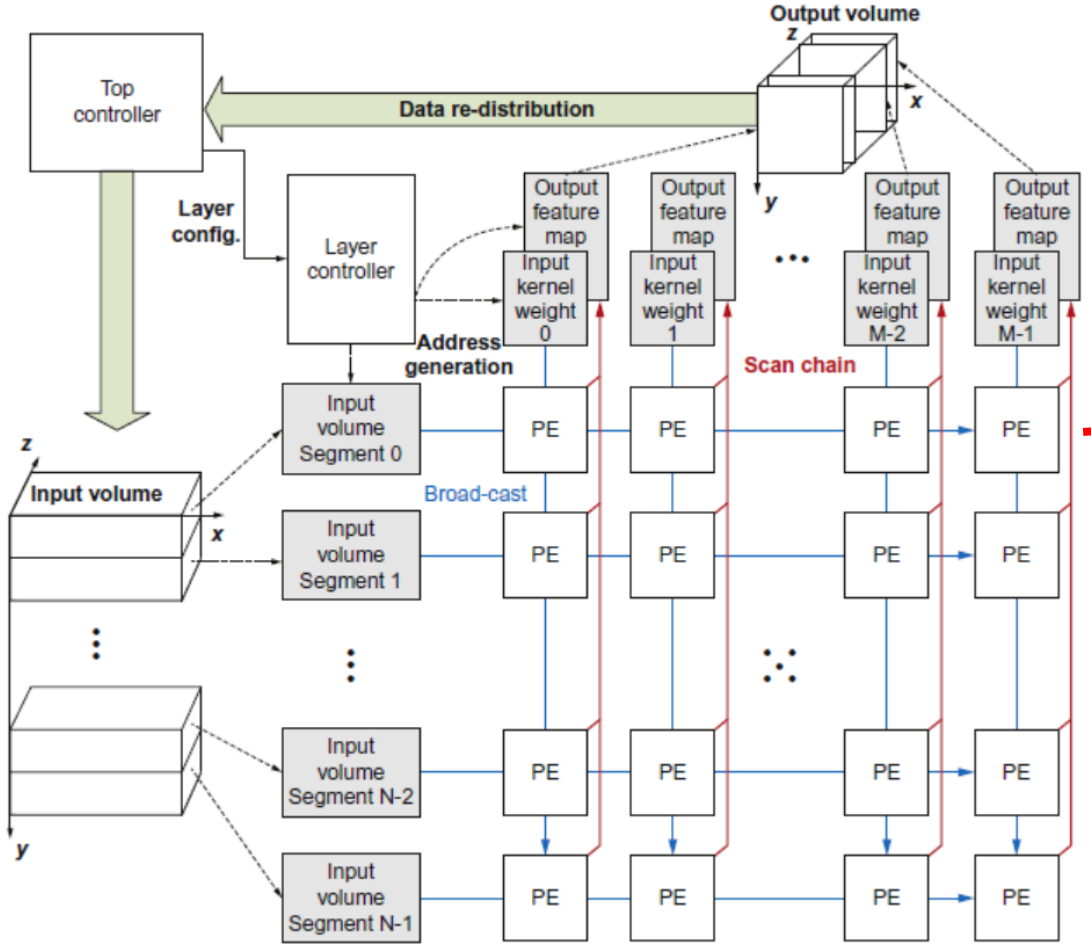
Intel Crest

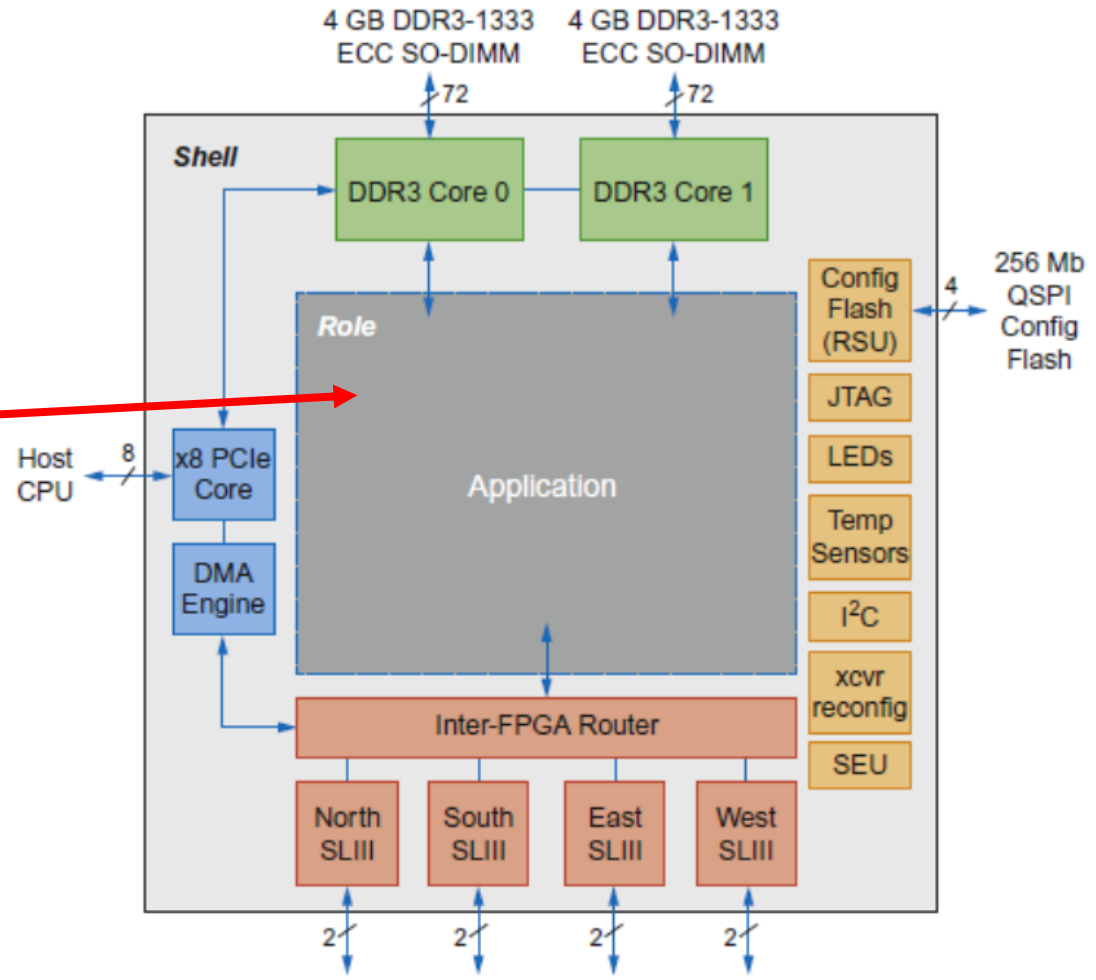
- DNN training
  - 16-bit fixed point
  - Operates on blocks of 32x32 matrices
  - SRAM + HBM2



Microsoft Catapult

- **CNN accelerator, mapped across multiple FPGAs**





Summary

- Processor is NOT the only way to compute
- Reconfigurable computers allows different tradeoffs among speed, flexibility, cost, power, etc
- FPGA offers fine-grain reconfigurability
- DSA and heterogeneous computing are trending
  - **Specialization, Customized Memories** -> Efficiency
  - **Parallelism** -> Speedup
  - **Co-Design:** The algorithm has to change
  - **Memory dominates**

Where are we Heading?

- ST2: Guest Lecture: Processing in Memory (PIM)

Acknowledgement

Slides in this topic are inspired in part by material developed and copyright by:

- ARM Courseware
- Nvidia Courseware
- Prof. Onur Mutlu @ ETH
- Prof. Joe Devietti @ Upenn, CIS 571
- Prof. Hakim Weatherspoon @ Cornell, CS 3410
- Prof. Krste Asanovic @ UCB, CS252
- Xinfei Guo @ JI, VE370 2021 SU

Action Items

- HW #5 due by 2:59AM July 20th, 2022 (Beijing Time)
- Final project
- Reading Materials
  - Ch. 7
  - David Patterson - Domain-Specific Architectures for Deep Neural Networks https://www.youtube.com/watch?v=FSwKCL8A9JQ&ab_channel=Matroid