Lecture Series on
Hardware for Deep Learning

Part 5:
The Deep Learning Acceleration Landscape

Dr. Adam Teman
EnICS Labs, Bar-Ilan University

21 May 2020

Emerging Nanoscaled
Integrated Circuits and Systems Labs

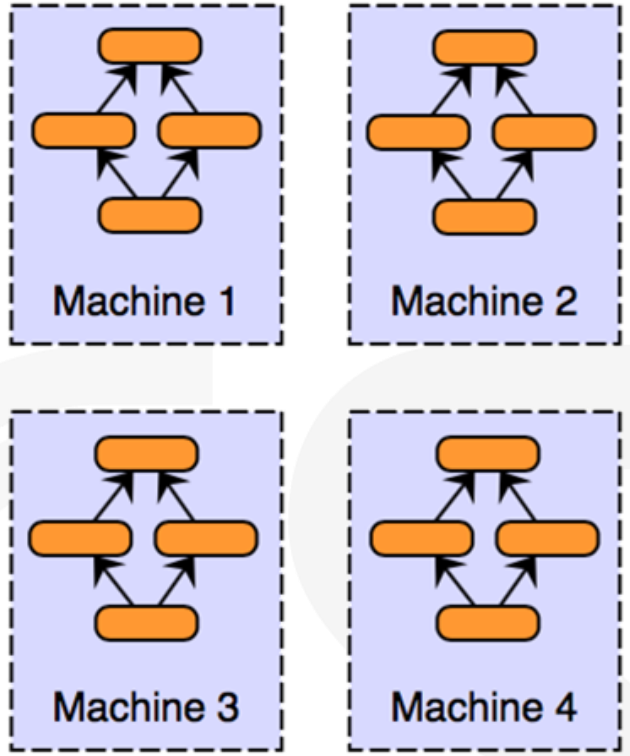
Tradition of Excellence
Bar-Ilan University
אוניברסיטת בר-אילן

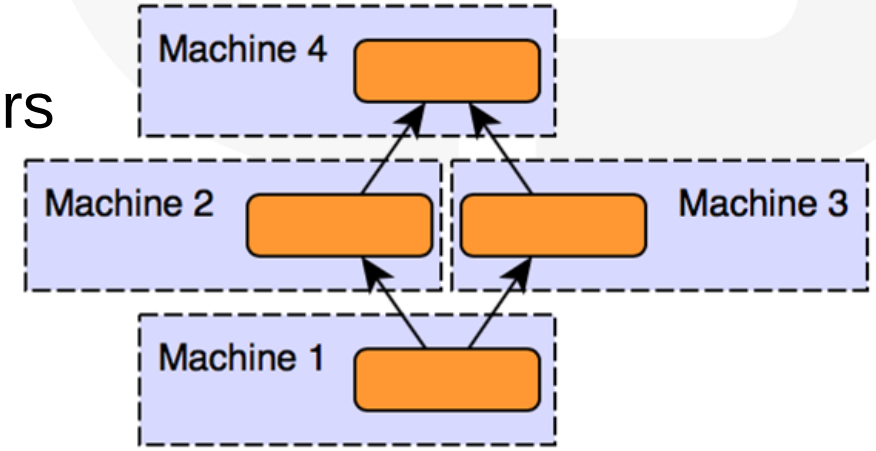
Outline

Parallelism in Deep Learning

Types of Parallelism

- **Data parallelism**
  - Different machines have a complete copy of the model
  - Each machine gets a different portion of the data
- **Model parallelism**
  - Each machine gets a different part of the model
  - For example, each machine gets a layer in a network
- **Hyper Parameter parallelism**
  - Models are trained with different hyper parameters on different machines

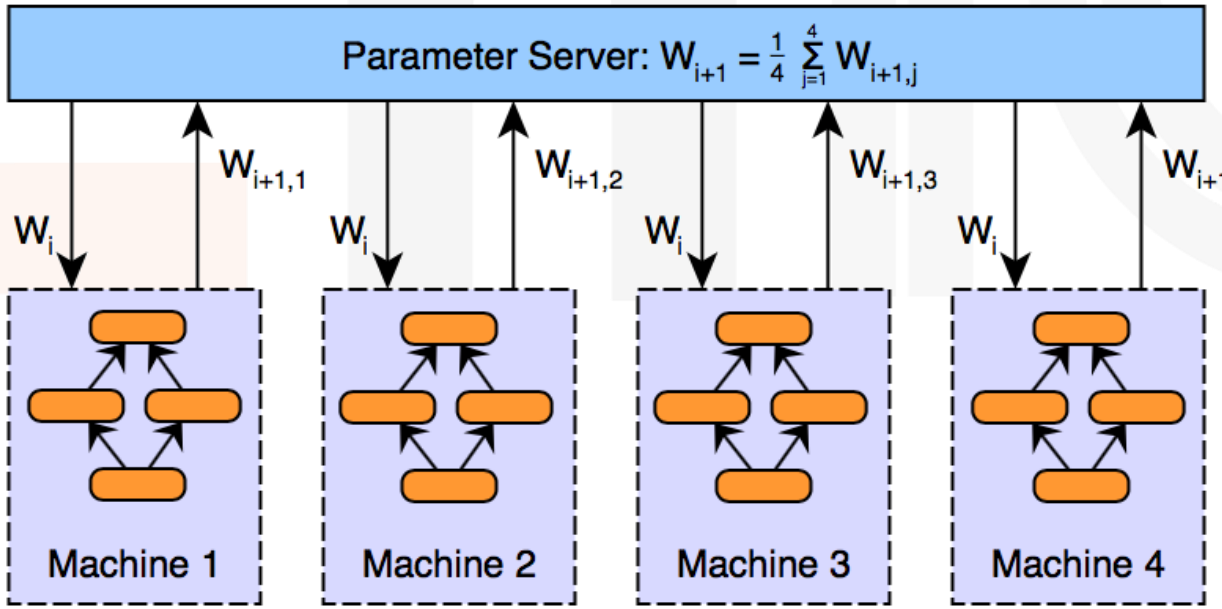




Source: Xiandong QI

Data Parallelism

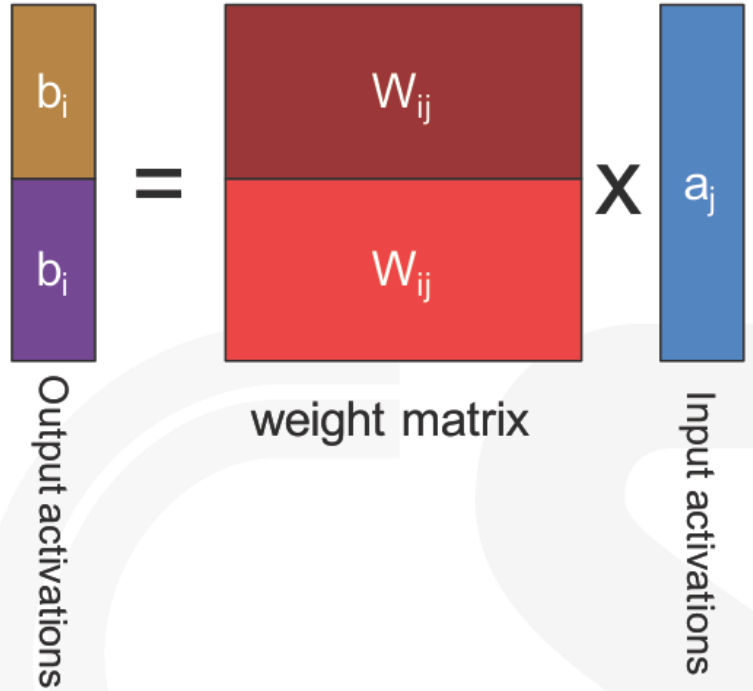
- **Parameter Averaging:**
  - Initialize the network parameters randomly based on the model configuration
  - Distribute a copy of the current parameters to each worker
  - Train each worker on a subset of the data
  - Set the global parameters to the average the parameters from each worker

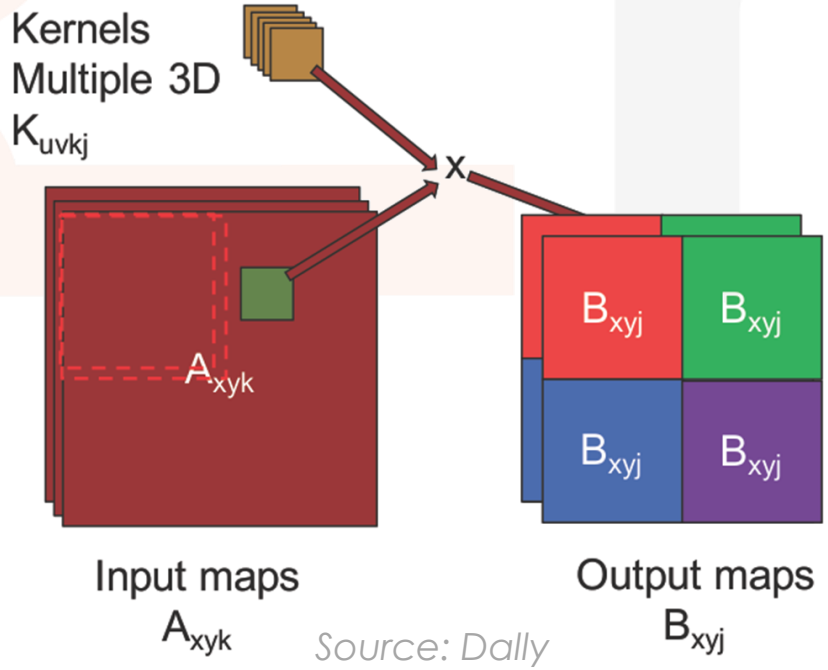


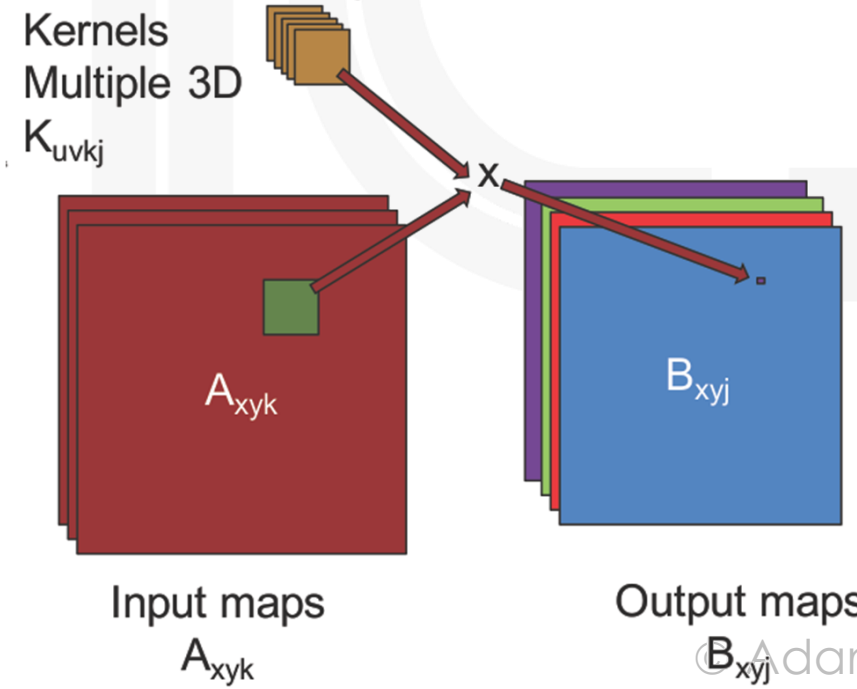
Source: Xiandong QI

Model Parallelism

- **Model-Parallel Convolution**
  - By output region
  - By output map
- **Model-Parallel Fully Connected Layers**
  - e.g., 16M independent multiplies in one FC layer



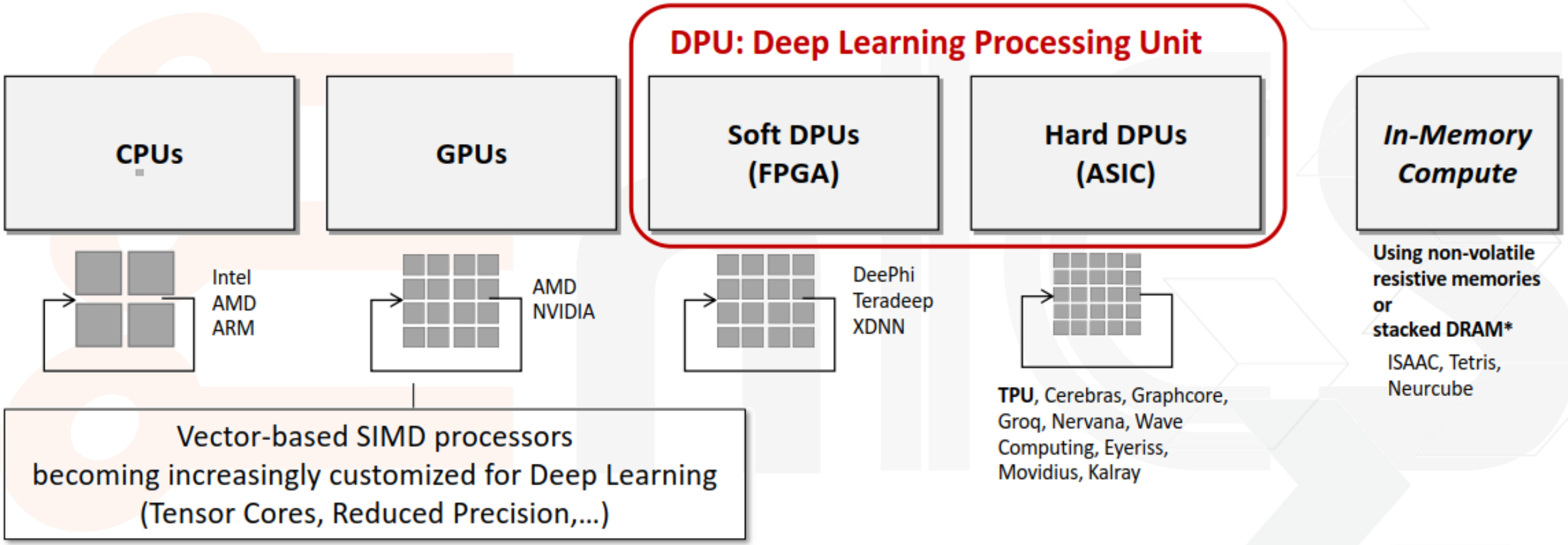




Source: Dally

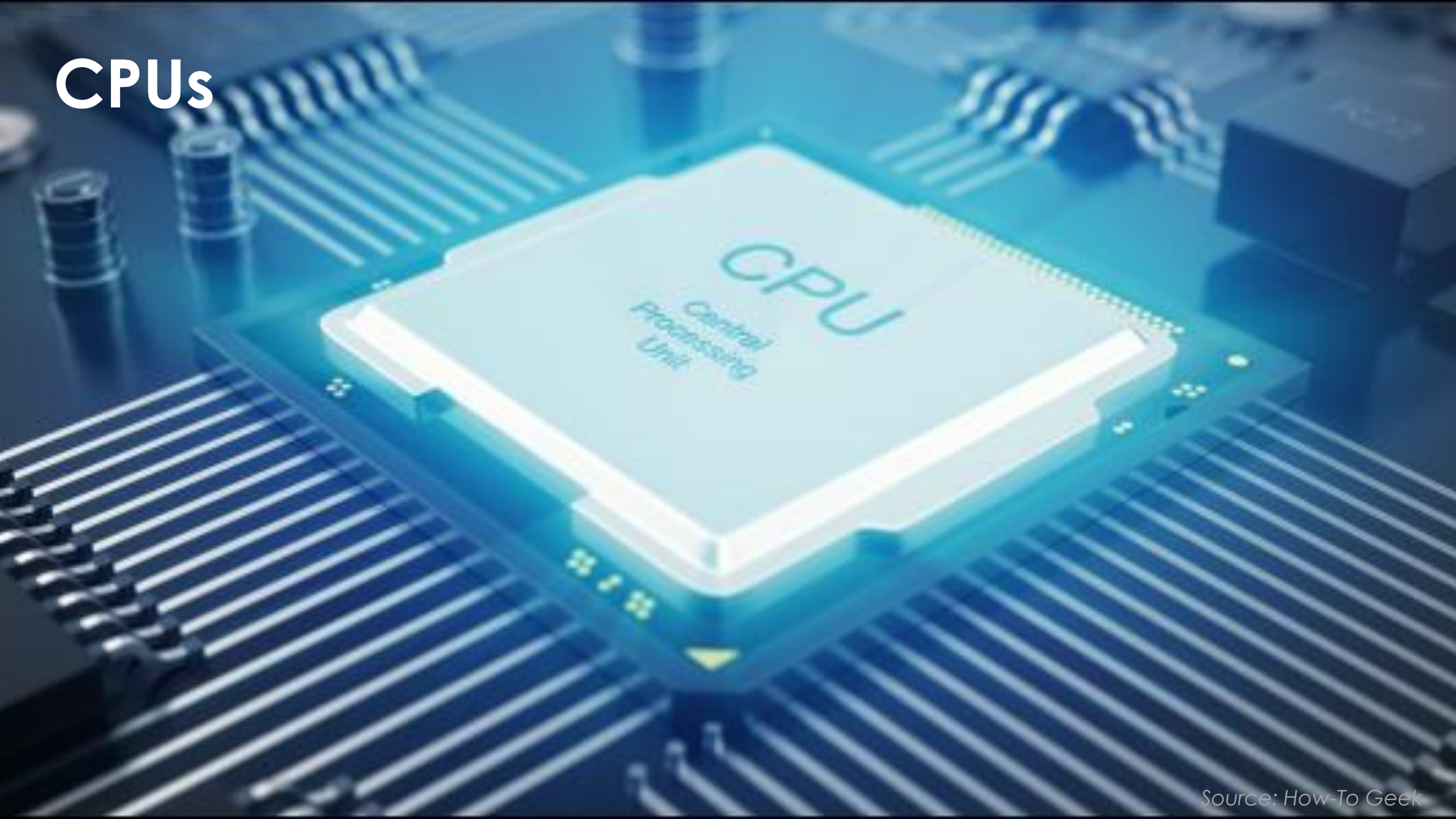
Traditional Programmable Hardware

Spectrum of Architectures for Deep Learning



Source: Blott, Xilinx

CPUs

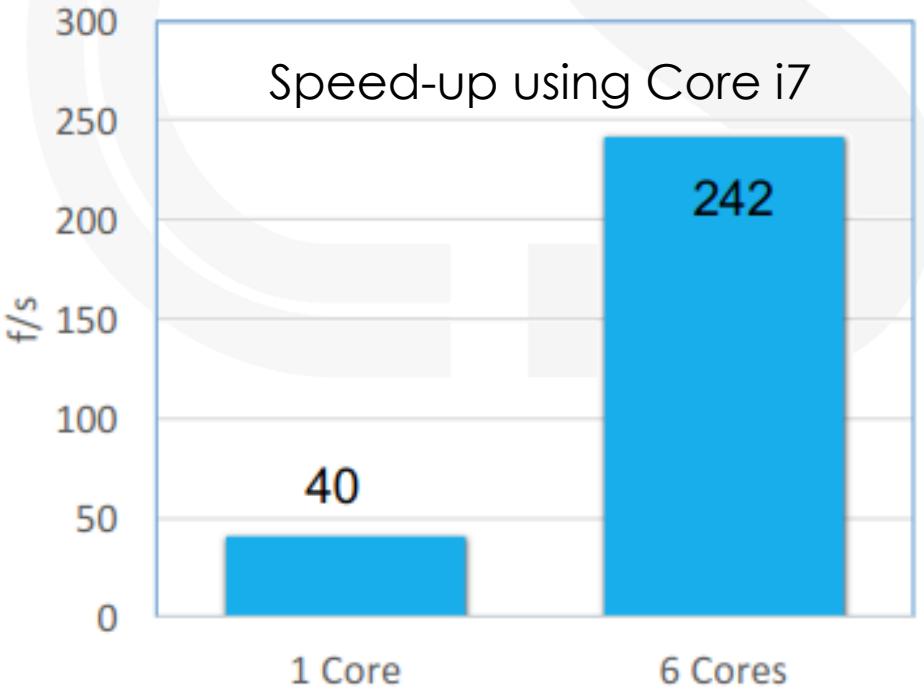


Source: How-To Geek

Using CPUs for Deep Learning

- **CPUs are (generally) optimized for single-threaded performance**
  - Desktop core i3-i9 have 2-10 cores
  - High end Intel/AMD CPUs can have up to 32 cores/64 threads
  - Server Intel Xeon/AMD EPYC have up to 64 cores/128 threads
- **Intel Xeon Scalable processors**
  - 28 cores/56 threads per socket – up to 8 sockets
  - Up to 1.5 TB DDR4
  - Price (Xeon Platinum 8180): $10K
- **AVX-512 Instructions**
  - Vector Neural Network Instructions (AVX512 VNNI)
  - Brain Floating-Point Format

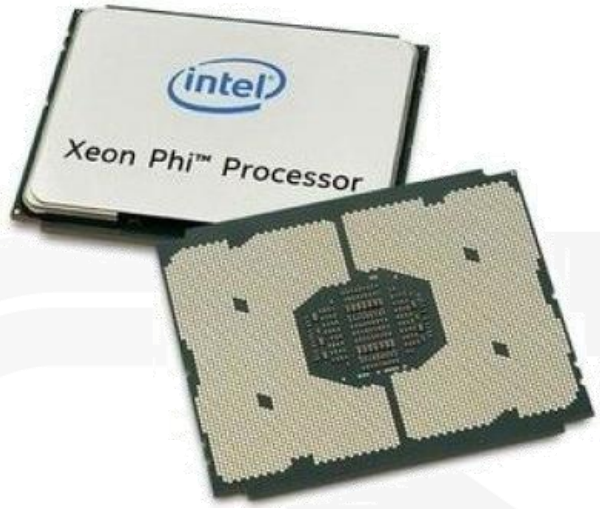
DLBoost

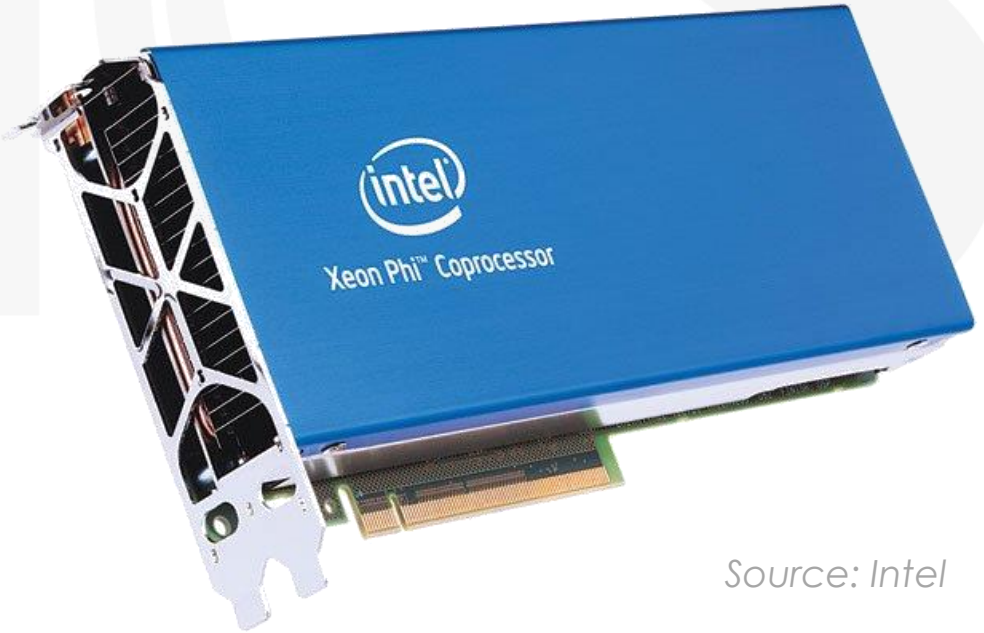


Source: NVIDIA

Xeon-Phi

- **The Intel attempt to make a heavily multi-core processor**
- **Xeon-Phi 7290F**
  - 72 cores, 4 threads per core (=288 threads)
  - Peak performance: 3456 GFLOPS DP (FP64)
  - Power: 260W
  - Price: $3.3K
- **The next generation was called “Knights Hill”**
  - Delayed due to problems with 10nm process
  - Eventually cancelled in Nov. 2017

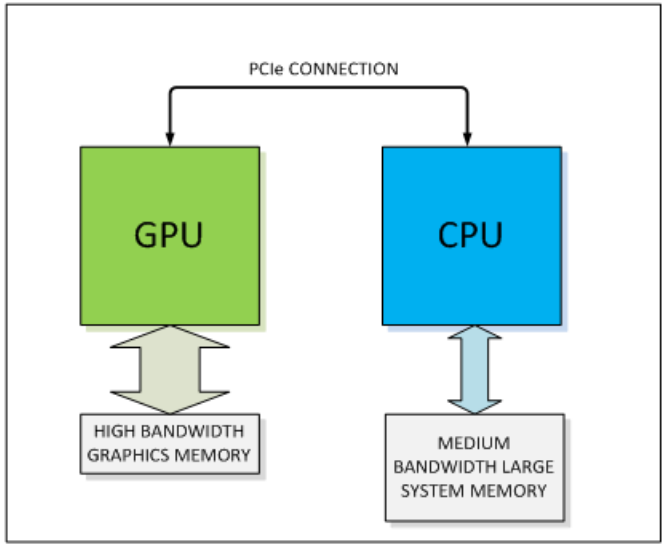


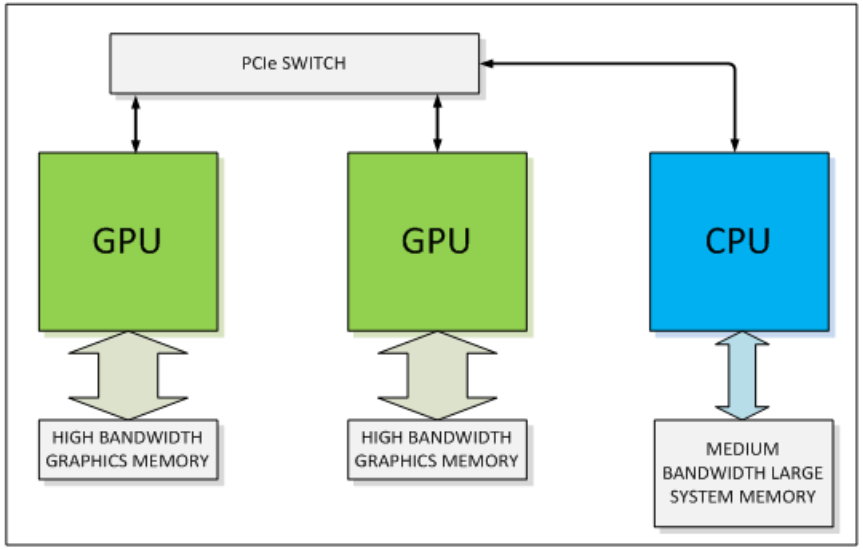


Source: Intel

CPU Interconnect: PCI Express (PCIe)

- **PCIe used to communicate between host CPU and GPU**
  - Need to support as many lanes as possible to communicate with many GPUs
  - Xeons have 64 PCIe v.3 lanes
- **PCIe v.3**
  - 985 MB/s per 1 lane, so 15.75 GB/s for x16 links.
- **PCIe v.4**
  - 31.51 GB/s for x16
- **PCIe v.5**
  - Expected: 63 GB/s for x16.





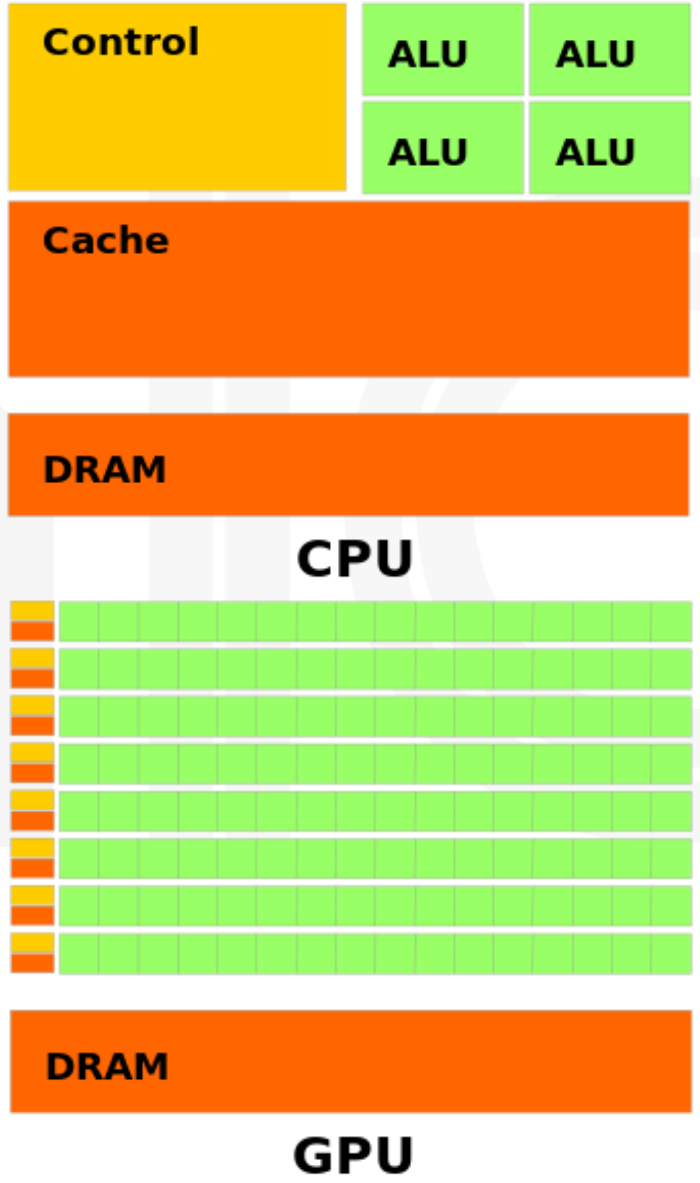
Source: NVIDIA

GPUs

Source: NVIDIA

CPU vs GPU Architectures

- **CPU**
  - Few Cores
  - Lots of Cache
  - Handful of Threads
  - Independent Processes
- **GPU**
  - Hundreds of Cores
  - Thousands of Threads
  - Single Process Execution



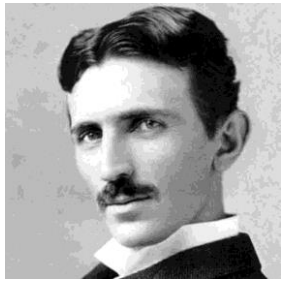
Source: wikipedia

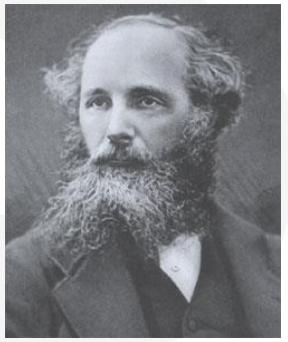
GPGPUs for Deep Learning

- **What GPUs are out there? Which ones are for deep learning?**
  - Okay, so this is *really confusing*…
- **GPUs were originally only for graphics/gaming and there are three major players:**
  - NVidia GeForce for consumers and Quadro for workstations
  - AMD Radeon (formerly ATI) for consumer and Radeon Pro for workstations
  - Intel HD Graphics, integrated with CPUs and soon Intel Xe standalone
- **But then people started using NVIDIA cards for deep learning**
  - And they introduced the **Tesla** series for training
  - And AMD followed with the Radeon Instinct, but much later
  - And the Tegra SoC series started being marketed for edge inference

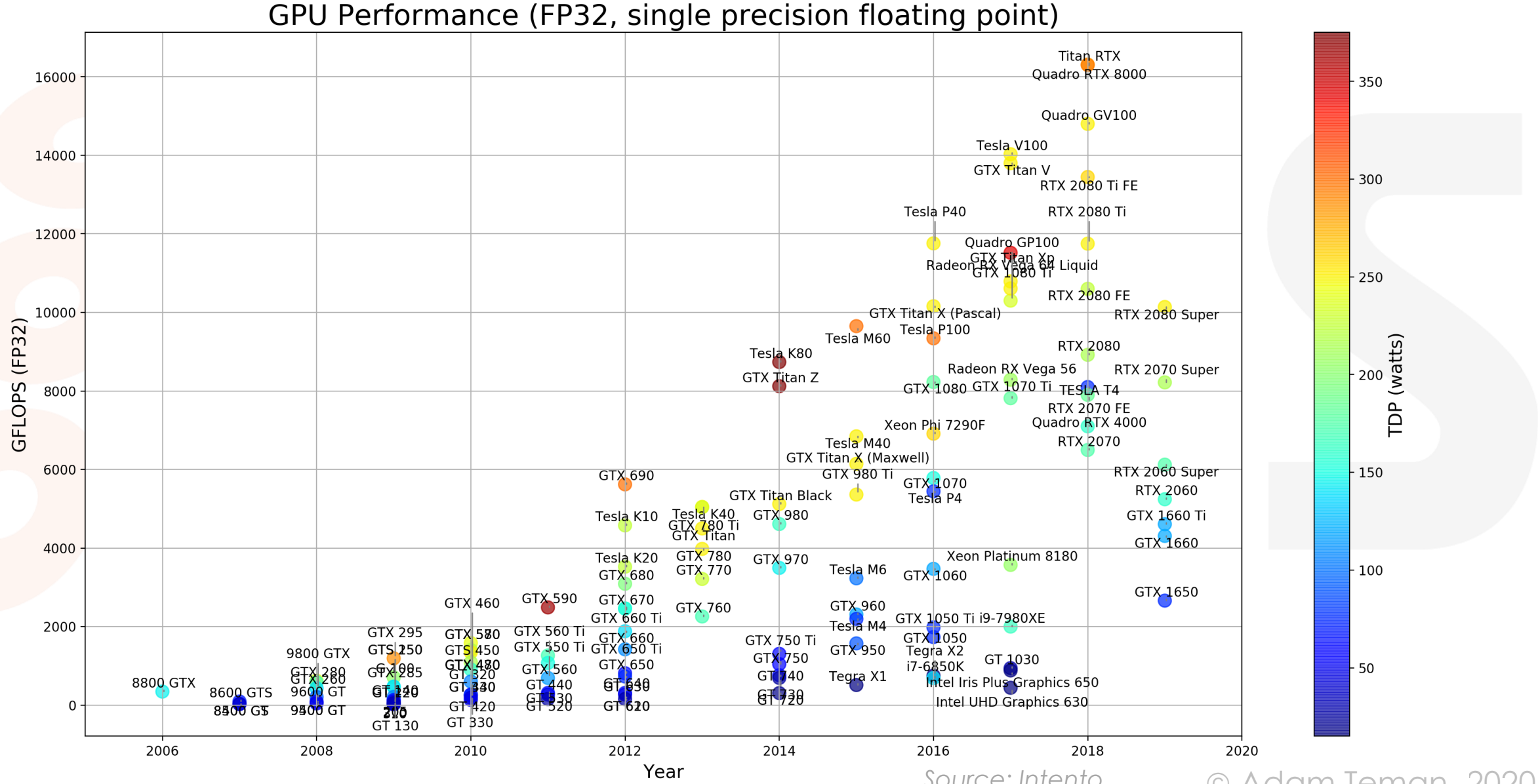
NVIDIA MicroArchitectures

- **NVIDIA Microarchitecture**
  - In addition to the “series” (GeForce, Quadro, Tesla), NVIDIA names their products according to the *generation* of the microarchitecture.
  - These are named after great scientists: Tesla, Fermi, Kepler, Maxwell
  - And recently they are DL oriented: Pascal, Volta, Turing, Ampere
- **Wait, but you said that Tesla was a series…**
  - Yes, it’s both a series and a microarchitecture
  - So, Tesla P100 uses the Pascal architecture with a GP100 chip
  - And the Tesla V100 usesthe Volta architecture with the GV100 chip
- **It gets even more confusing if you try to figure out which is best**
  - For example, recent Quadro RTX and Titan RTX (Turing architecture) are really good, but they’re not marketed for deep learning training…





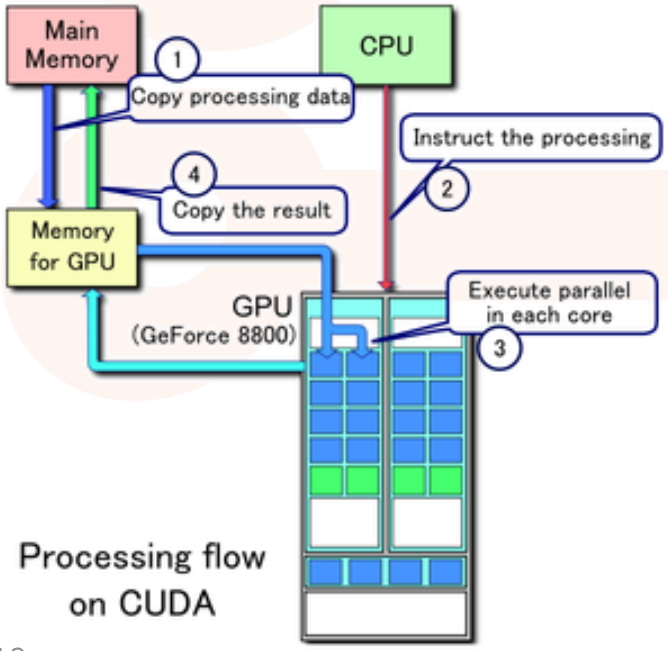

FP32 Performance of NVidia GPUs



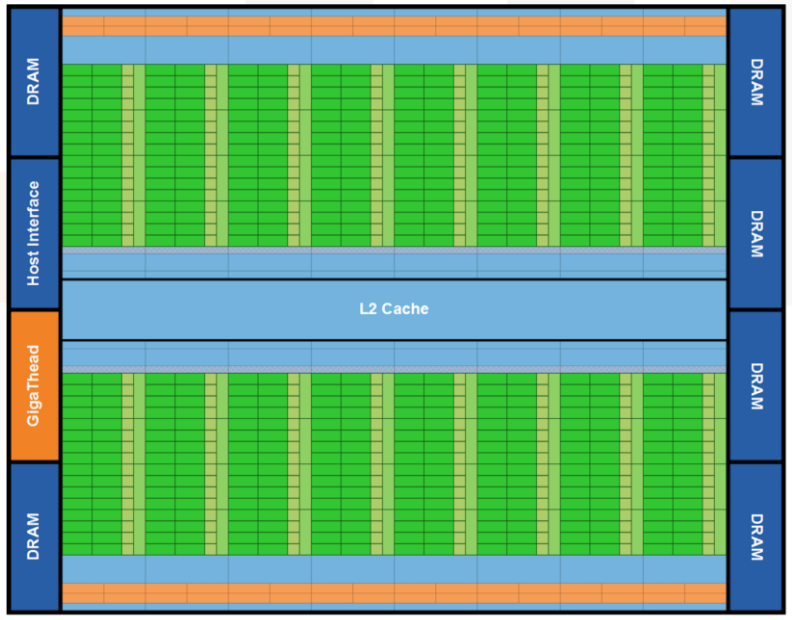
Source: Intento

GPUs in a nutshell

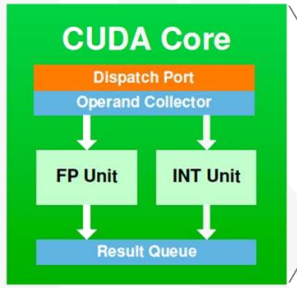
- **GPUs perform "Single Instruction Multiple Data" (SIMD) operations.**
  - NVIDIA has several "Streaming Multiprocessors" (SMs)
  - Each SM has many Compute Unified Device Architecture (CUDA) cores

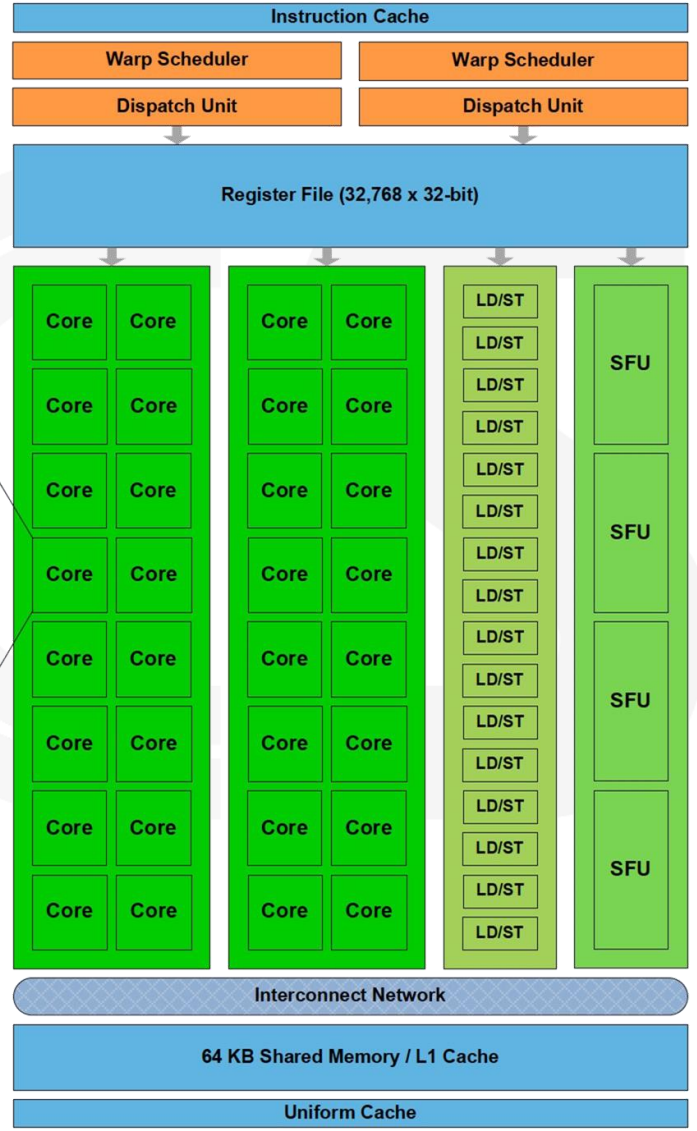


Source: wikipedia



Fermi Architecture with 16 SMs

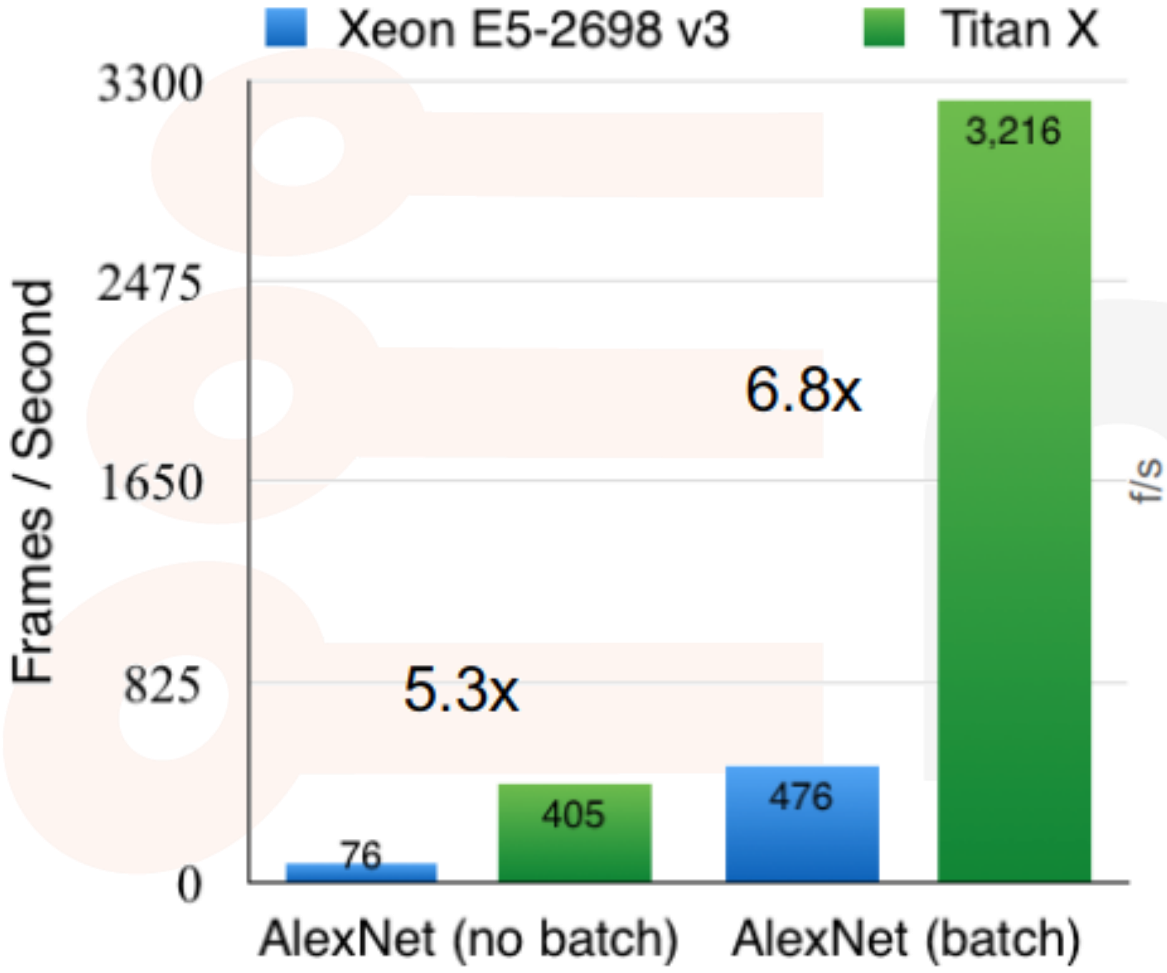




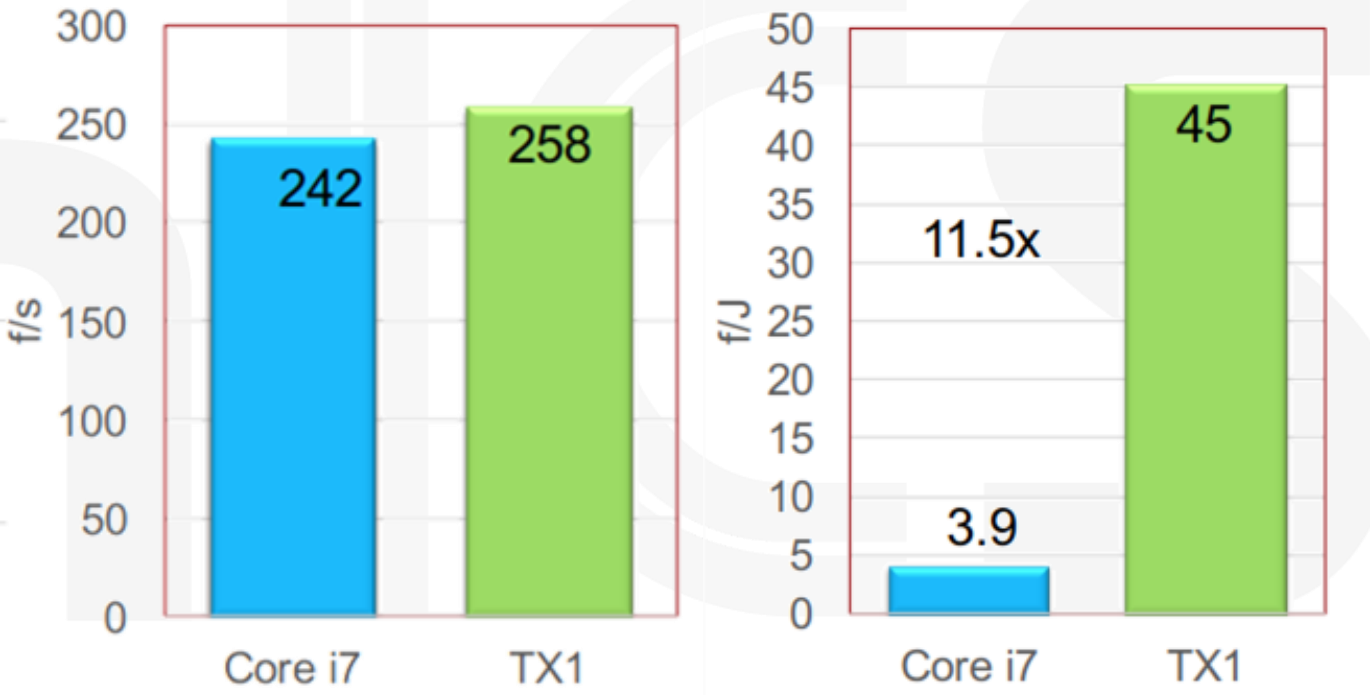
Source: NVidia

Fermi Streaming Multiprocessor

Example speedup and efficiency



Titan X (Maxwell) vs Xeon



Tegra SoC vs Core i7

Source: NVidia

Volta Architecture

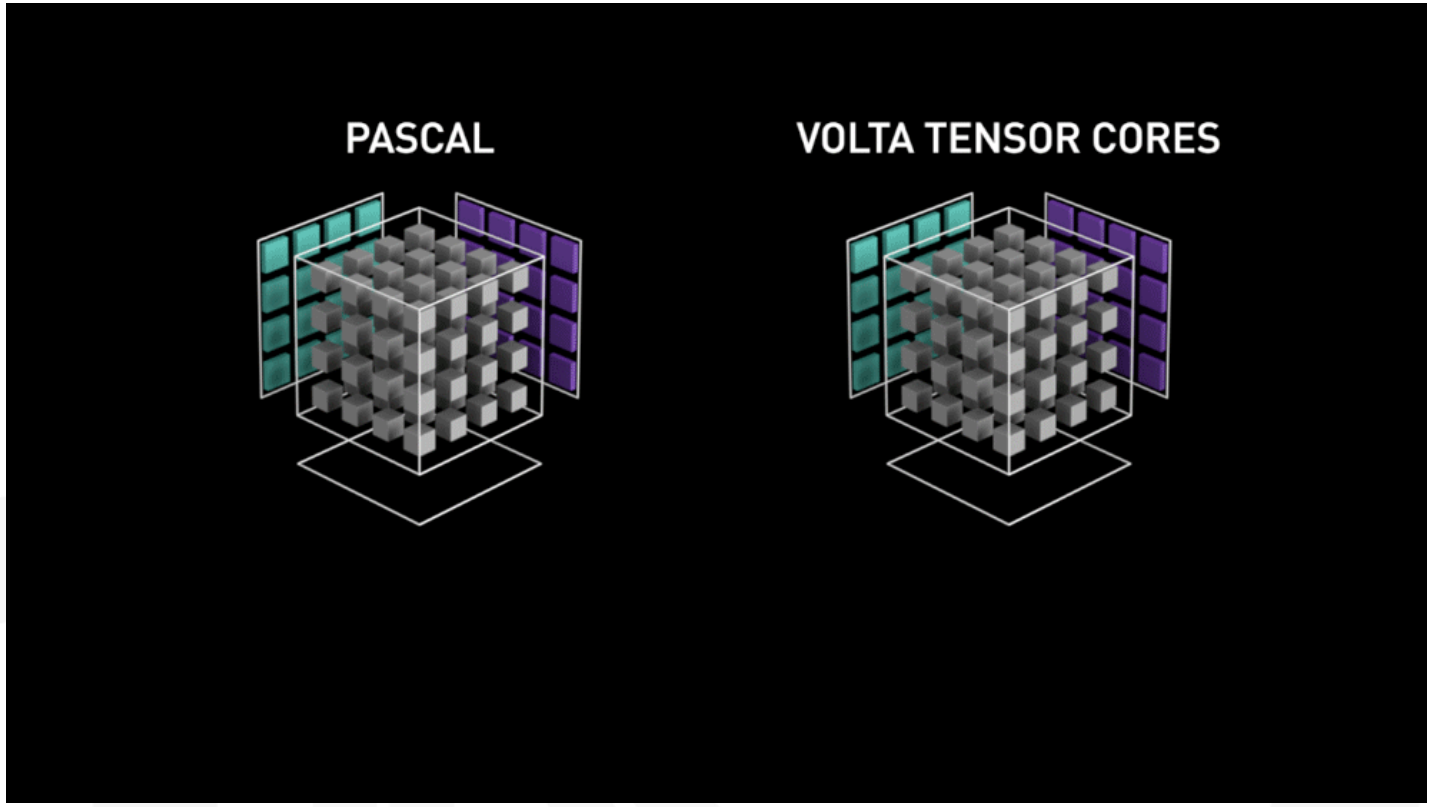
Volta GV100 Full GPU with 84 SM Units

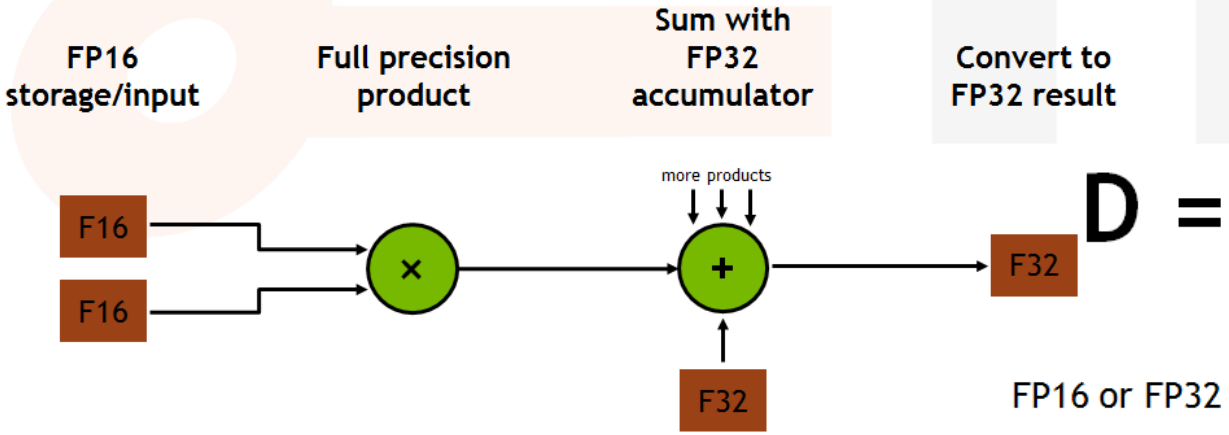
Quarter of a Volta SM

Source: NVidia

Tensor Cores

- Released with Volta architecture
- Processes 4x4x4 MAC in single operation
- Takes two FP16 inputs and accumulates in FP32





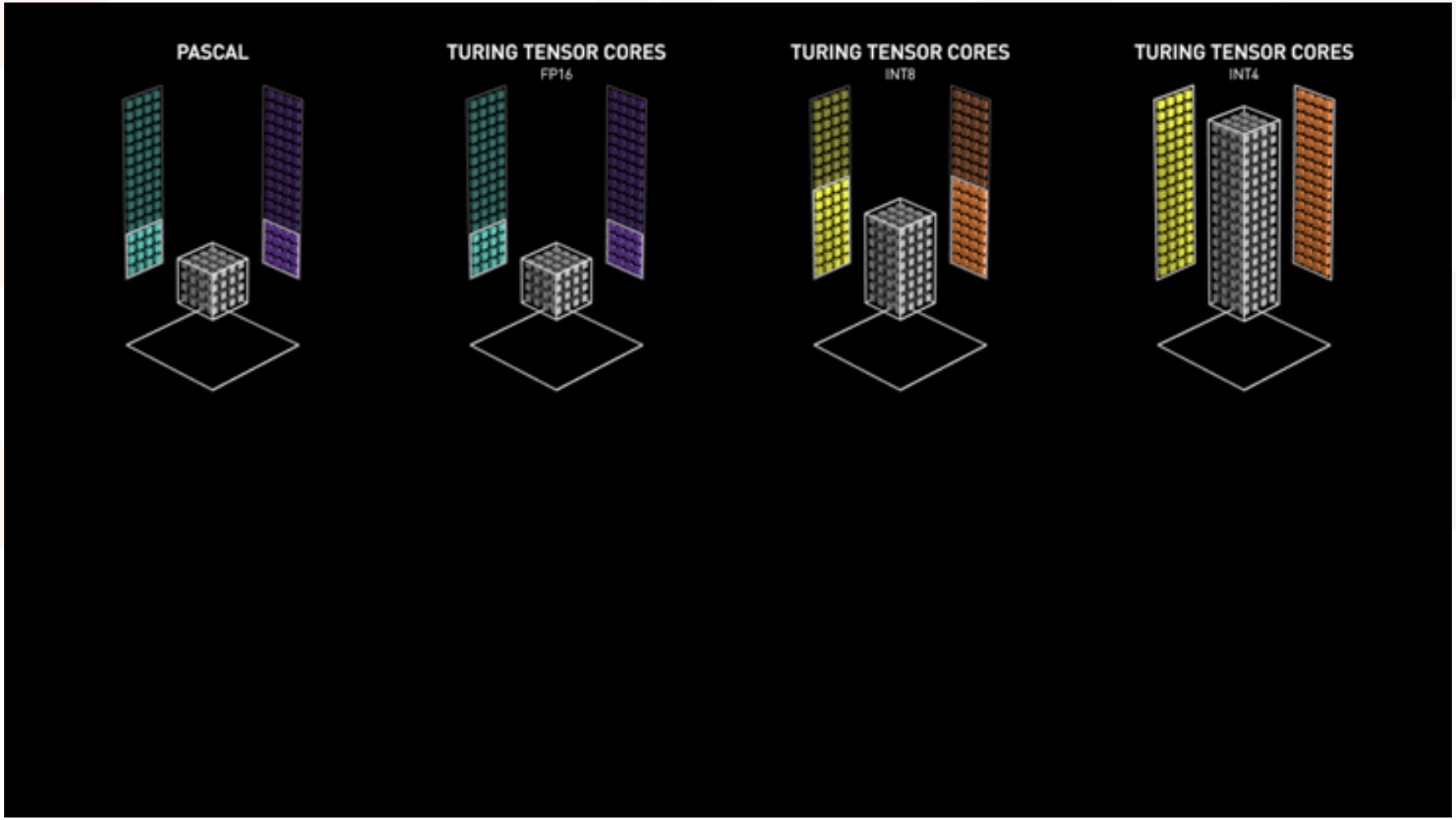
$$D = \begin{pmatrix} A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} \\ A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix} \begin{pmatrix} B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} \\ B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} \\ B_{2,0} & B_{2,1} & B_{2,2} & B_{2,3} \\ B_{3,0} & B_{3,1} & B_{3,2} & B_{3,3} \end{pmatrix} + \begin{pmatrix} C_{0,0} & C_{0,1} & C_{0,2} & C_{0,3} \\ C_{1,0} & C_{1,1} & C_{1,2} & C_{1,3} \\ C_{2,0} & C_{2,1} & C_{2,2} & C_{2,3} \\ C_{3,0} & C_{3,1} & C_{3,2} & C_{3,3} \end{pmatrix}$$

FP16 or FP32 — FP16 — FP16 — FP16 or FP32

Source: NVidia

Turing Tensor Cores

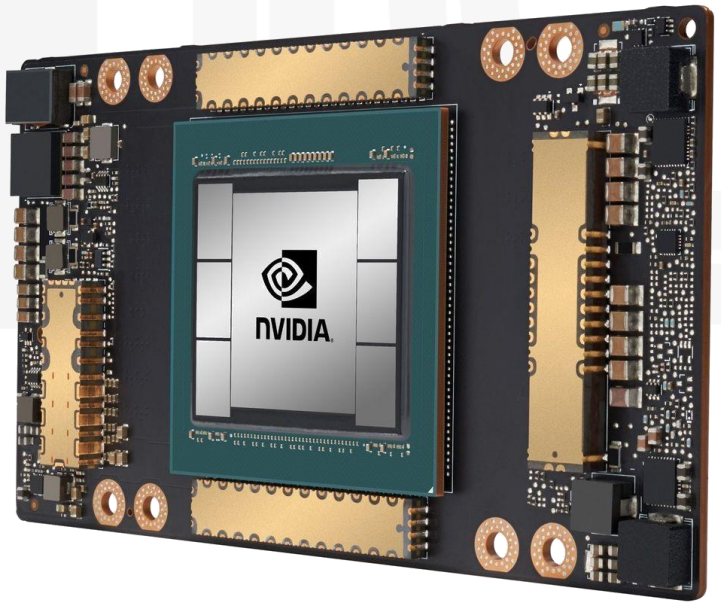
- The Turing architecture introduced new Tensor Cores that support INT8 and INT4 precision
- Included in the Tesla T4 cards



Source: NVidia

Hot off the Press: Ampere

- **Ampere GA100**
  - Up to 128 SMs / 8192 CUDA Cores
  - 54 B Transistors, 7nm Process, 826 mm^2
  - 6 stacks of HBM2 (5120-bit) → 40GB DRAM
  - Boost clock: 1.41 GHz, Power: 400W
  - TF32 format – 19 bits:
    precision of FP16 with exponent of FP32
  - 312 TFLOPs for TF32 calculations
    20X faster than the V100@FP32
  - 19.5 TFLOPs@FP64
    2.5X faster than the V100@FP64



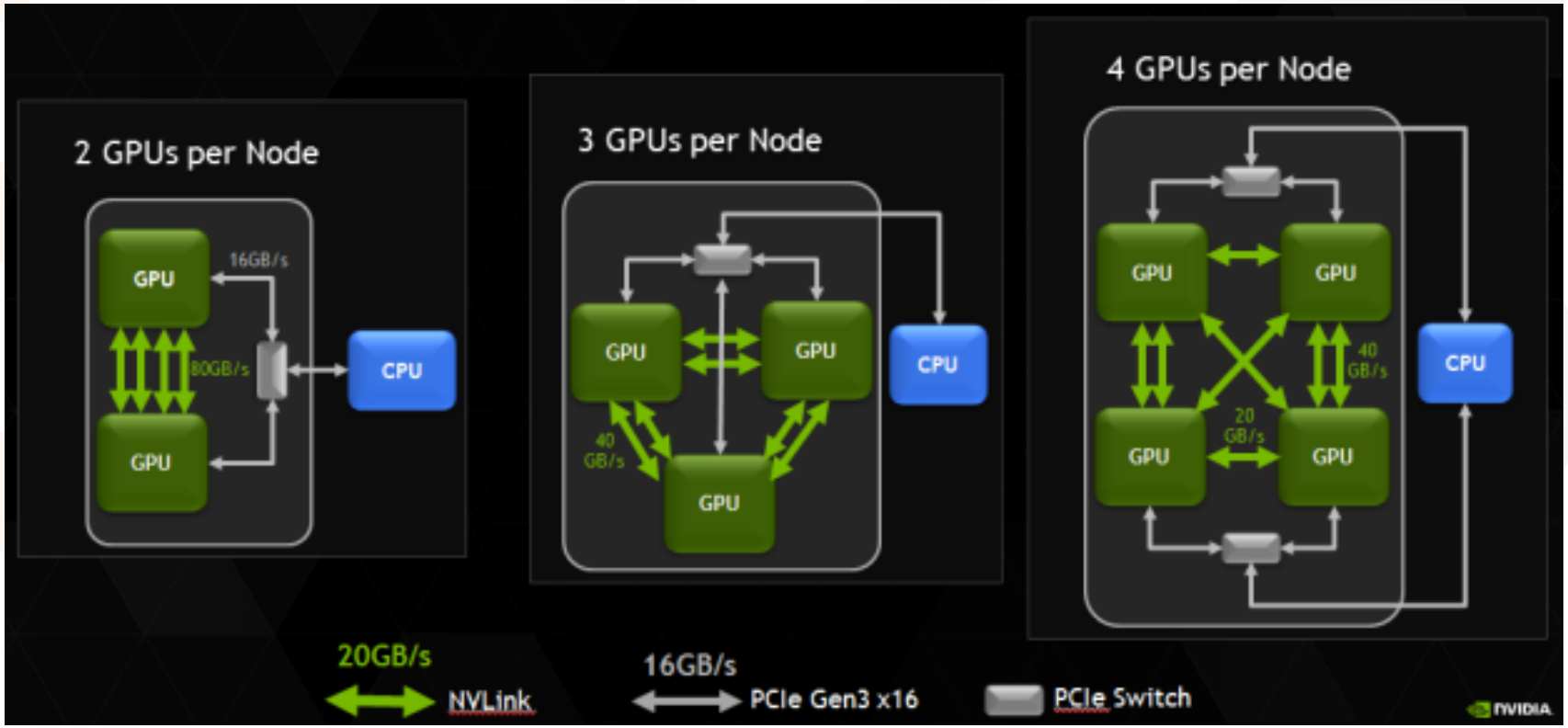
Source: NVidia

Data Center GPU	NVIDIA Tesla P100	NVIDIA Tesla V100	NVIDIA A100
SMs	56	80	108
FP32 Cores / GPU	3584	5120	6912
FP64 Cores / GPU	1792	2560	3456
INT32 Cores / GPU	NA	5120	6912
Tensor Cores / GPU	NA	640	432
GPU Boost Clock	1480 MHz	1530 MHz	1410 MHz
Peak TF32 Tensor TFLOPS	NA	NA	156/312
Peak FP16 TFLOPS	21.2	31.4	78
Peak FP64 TFLOPS	5.3	7.8	9.7
Memory Interface	4096-bit HBM2	4096-bit HBM2	5120-bit HBM2
Memory Size	16 GB	32 GB / 16 GB	40 GB
Memory Data Rate	703 MHz DDR	877.5 MHz DDR	1215 MHz DDR
Memory Bandwidth	720 GB/sec	900 GB/sec	1.6 TB/sec
L2 Cache Size	4096 KB	6144 KB	40960 KB
TDP	300 Watts	300 Watts	400 Watts
Transistors	15.3 billion	21.1 billion	54.2 billion
GPU Die Size	610 mm^2	815 mm^2	826 mm^2
TSMC Manufacturing Process	16 nm FinFET+	12 nm FFN	7 nm N7

Source: NVidia

NVIDIA Interconnect: NVLink

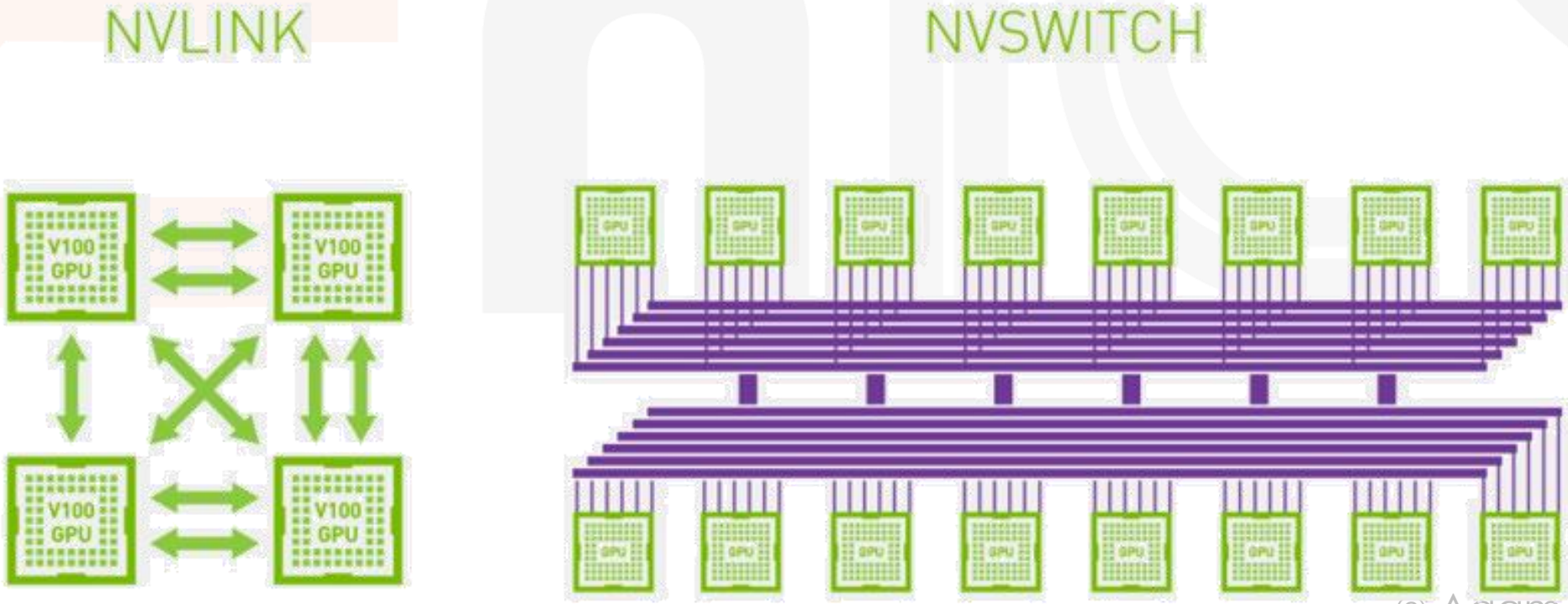
- **NVIDIA NVLink is an energy-efficient, high-bandwidth path between the GPU and the CPU.**
- **NVLink 1.0**
  - 80 GB/s
- **NVLink 2.0**
  - 150 GB/s
- **New NVLink**
  - 600 GB/s
- **Infinity Fabric**
  - A similar technology by AMD



Source: NVidia

NVLink

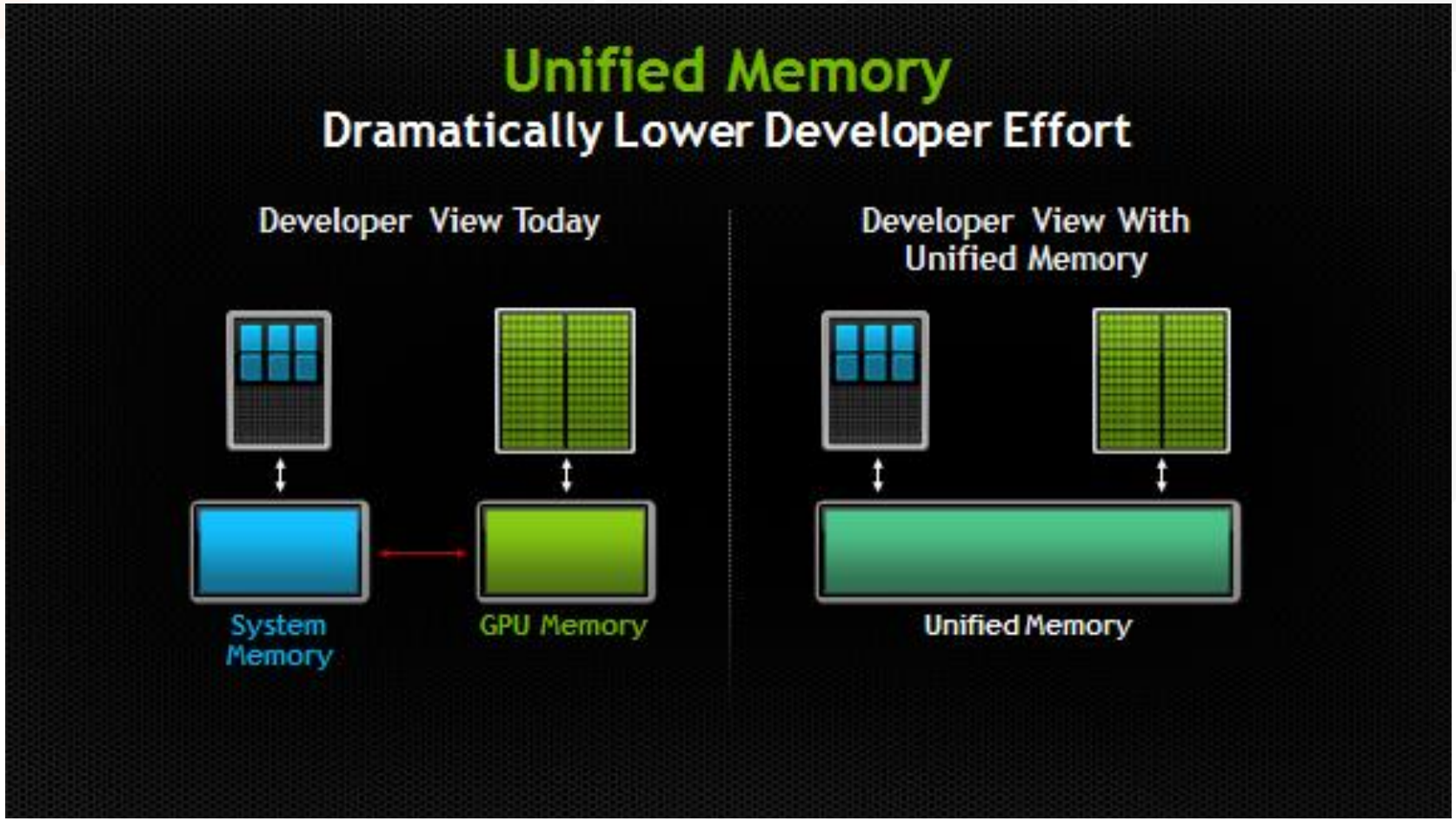
- **Separate cards can joined using NVLink**
- **NVSwitch: The Fully Connected NVLink**
  - Six NVSwitch with the newly announced NVLink → 4.8 TB/s



Source: NVidia

Unified Memory

- **"one of the most dramatic programming model improvements in the history of the CUDA platform"**
  - A pool of managed memory shared between the CPU and GPU
  - Not yet supported by DL frameworks



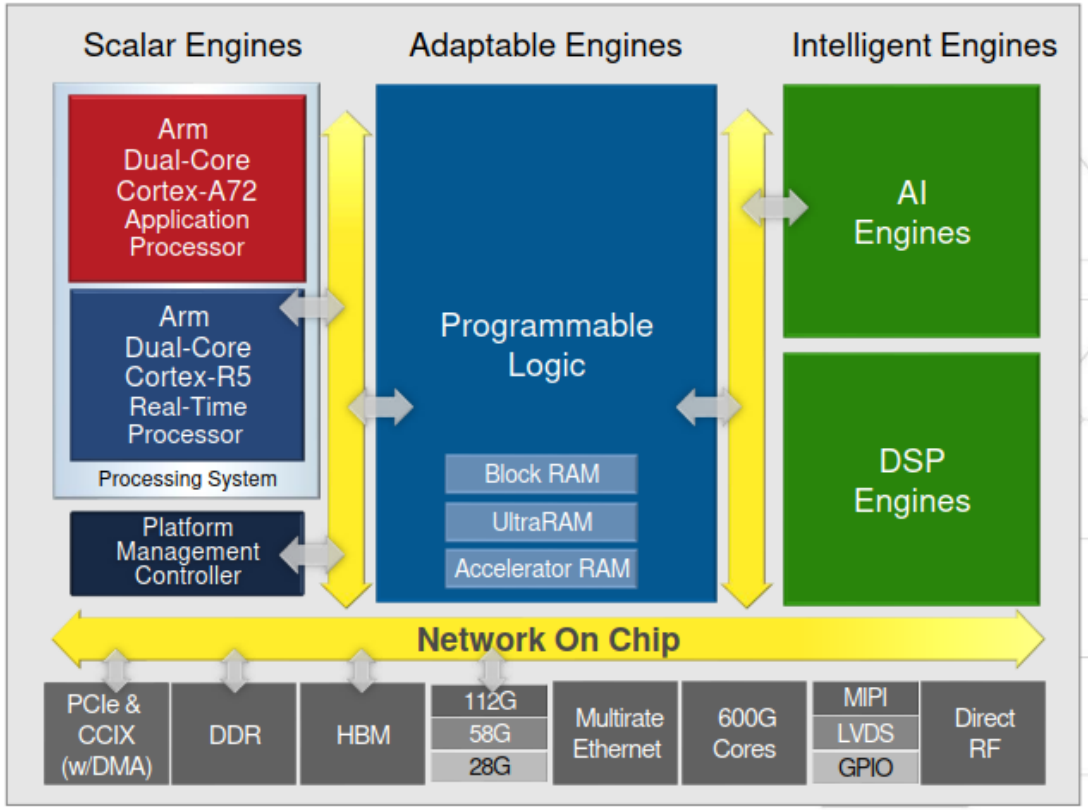
Source: NVidia

FPGAs

Source: Xiliinx

Don't forget the FPGAs

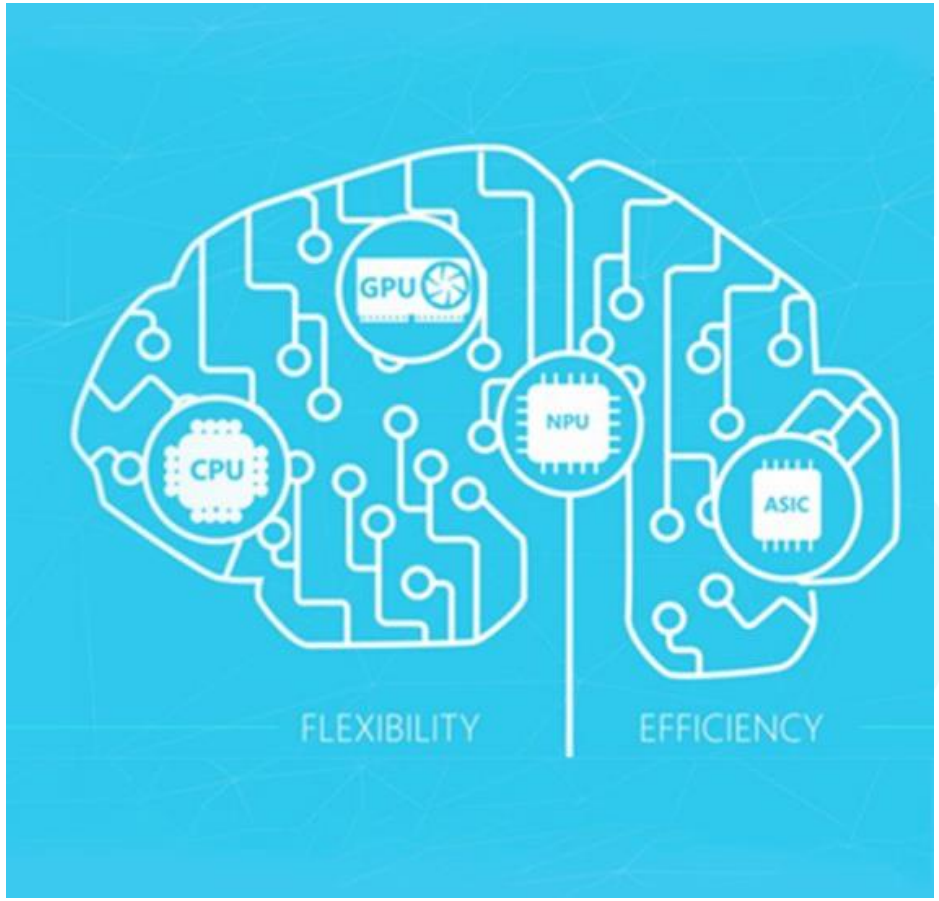
- **Xilinx Versal AI Core**
  - Branded an "ACAP" (Adaptive Compute Acceleration Platform)
- **Specs**
  - 7nm FF, 37B transistors, 885Mb SRAM
  - 400 AI engine cores,
- **Includes**
  - DSP and AI Compute Engines
  - NoC and Memory (DDR and HBM)
  - High Speed Interfaces
  - PCIe (Gen4) & CCIX (Gen5), Ethernet 600 Gbps
  - SerDes and RF



Source: Xilinx

Microsoft Project Brainwave

- **What is Project Brainwave?**
  - A deep learning platform for real-time AI inference in the cloud and on the edge
  - Based on "*Soft Neural Processing Units*"
  - Implemented on FPGAs
  - Augments CPUs with with an interconnected and configurable compute layer composed of programmable silicon.
- **On the cloud with Azure Machine Learning**
  - Distribute models onto Intel Arria 10 boards
- **At the edge with Azure Stack Edge**
  - Microsoft ships you a cloud-managed device with built in FPGA



Source: Microsoft

Specialized Deep Learning Hardware Platforms

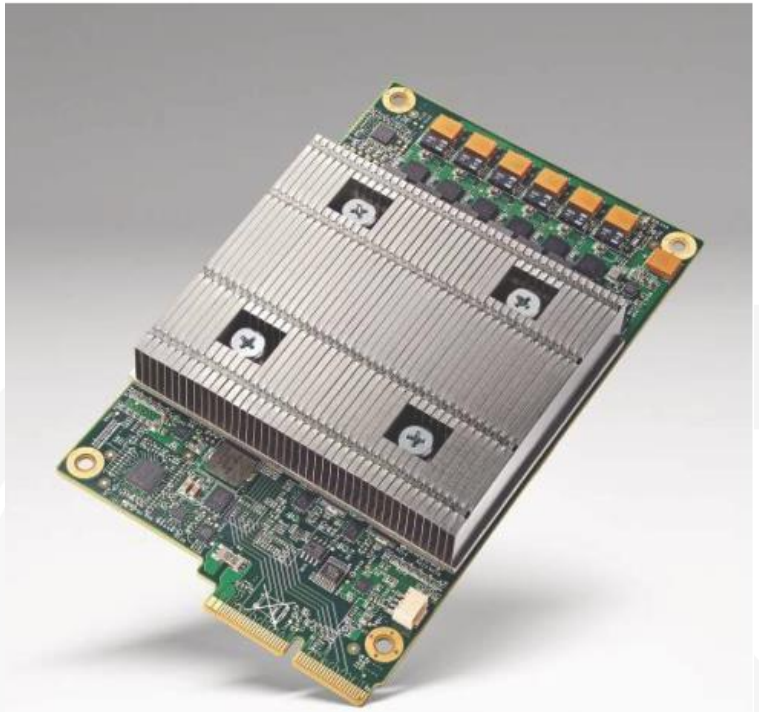
Emerging Nanoscaled Integrated Circuits and Systems Labs

Bar-Ilan University
אוניברסיטת בר-אילן

Google TPU

Source: Google

Google TPU

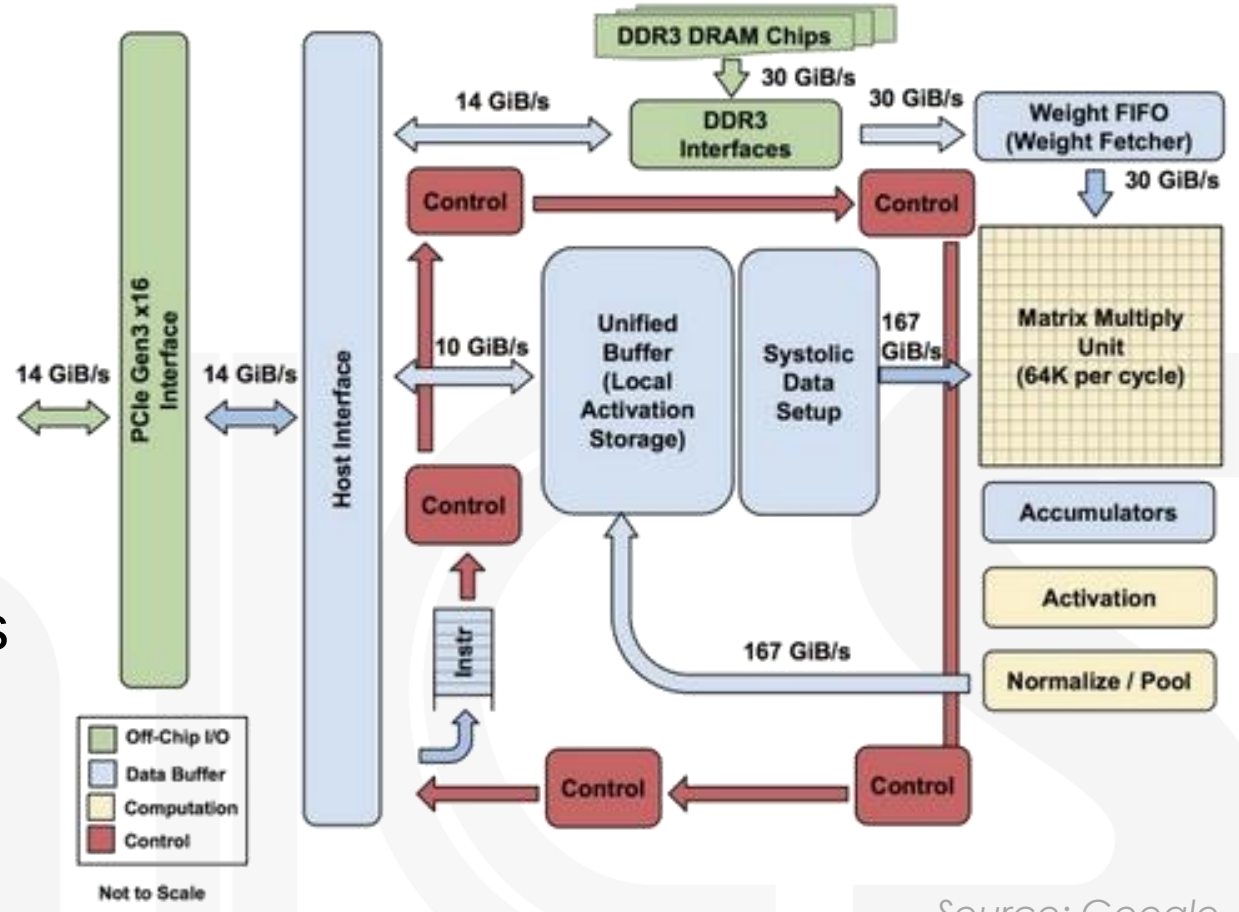


Source: Google

- **Tensor Processing Unit (TPU)**
- **Released in 2014 at Google I/O**
  - Developed in just 15 months due to fast growing computational demands of neural networks
  - Project led by Norm Jouppi
  - Fits in a SATA hard disk slot for drop-in installation
  - 331 mm^2 die, 28nm, 700 MHz, 40W
  - Die area: 35% memory, 24% matrix multiply unit, 41% remaining area for logic.
  - 8 GB DDR3-2133 DRAM accessible via two ports at 34 GB/s.
  - Communicate with Host through PCIe-3 x 16 (12.5 GB/s).
- **Uses 8-bit quantization for inference**
  - A TPU contains 65,536 8-bit integer multipliers (~25X more than a GPU)

TPU v1 Architecture

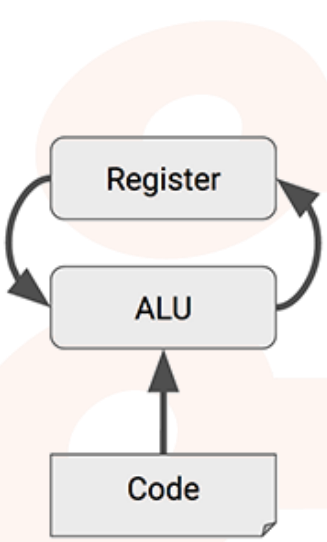
- **Matrix Multiplier Unit (MXU):**
  - 65,536 8-bit multiply-and-add units for matrix operations
- **Unified Buffer (UB):**
  - 24MB of SRAM that work as registers
- **Activation Unit (AU):**
  - Hardwired activation functions
- **Special ISA:**
  - *Read_Host_Memory*: Read data from memory
  - *Read_Weights*: Read weights from memory
  - *MatrixMultiply/Convolve*: Multiply or convolve with the data and weights, accumulate the results
  - *Activate*: Apply activation functions
  - *Write_Host_Memory*: Write result to memory

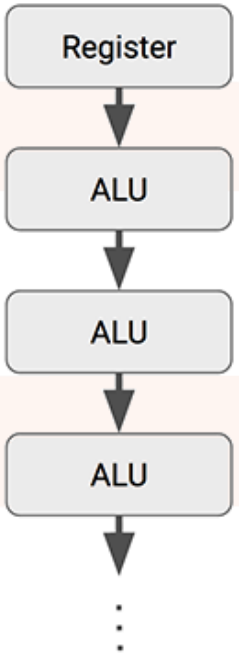


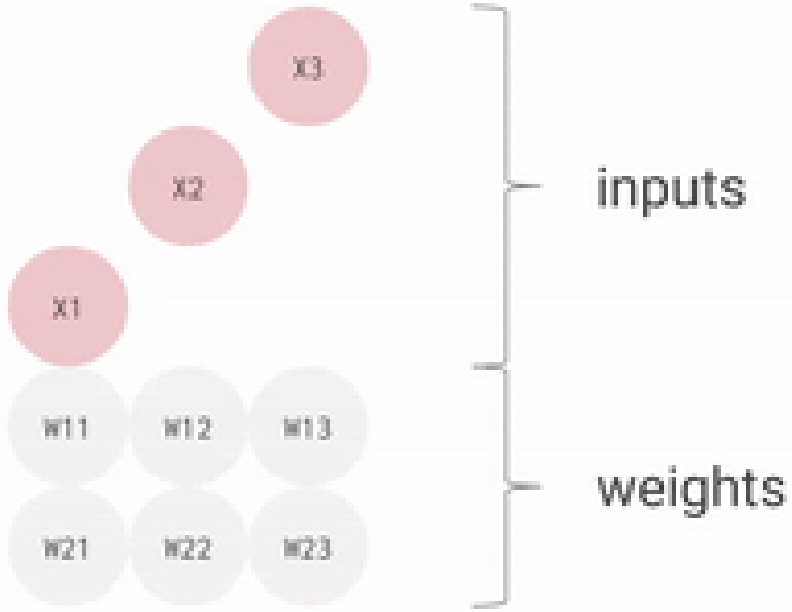
Source: Google

Systolic Array

- **Read each value once and use it multiple times without storing it back to a register**







Source: Google

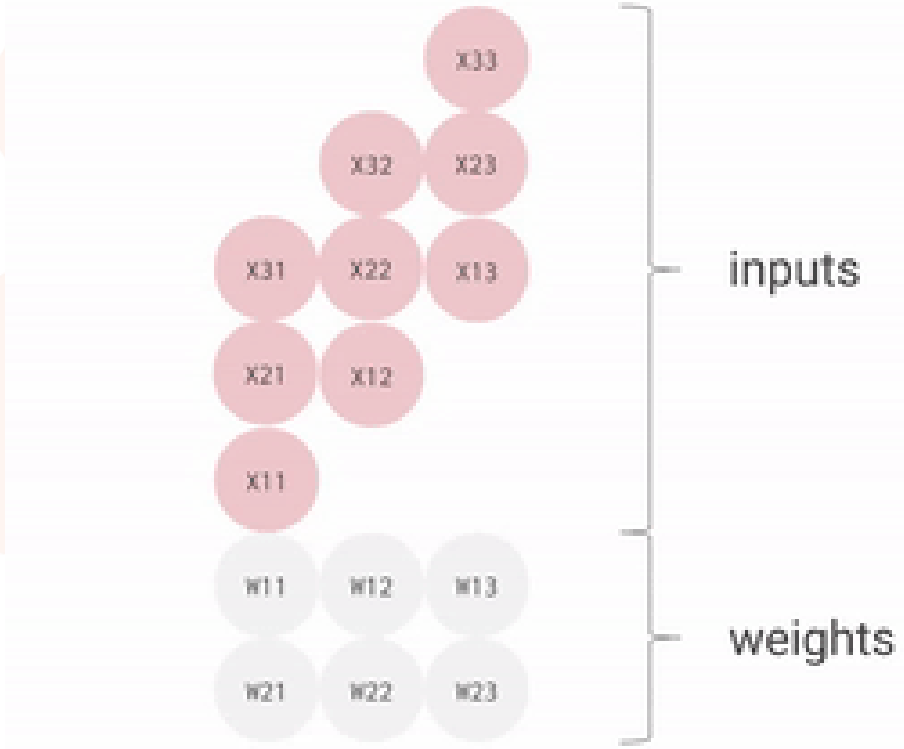
CPU

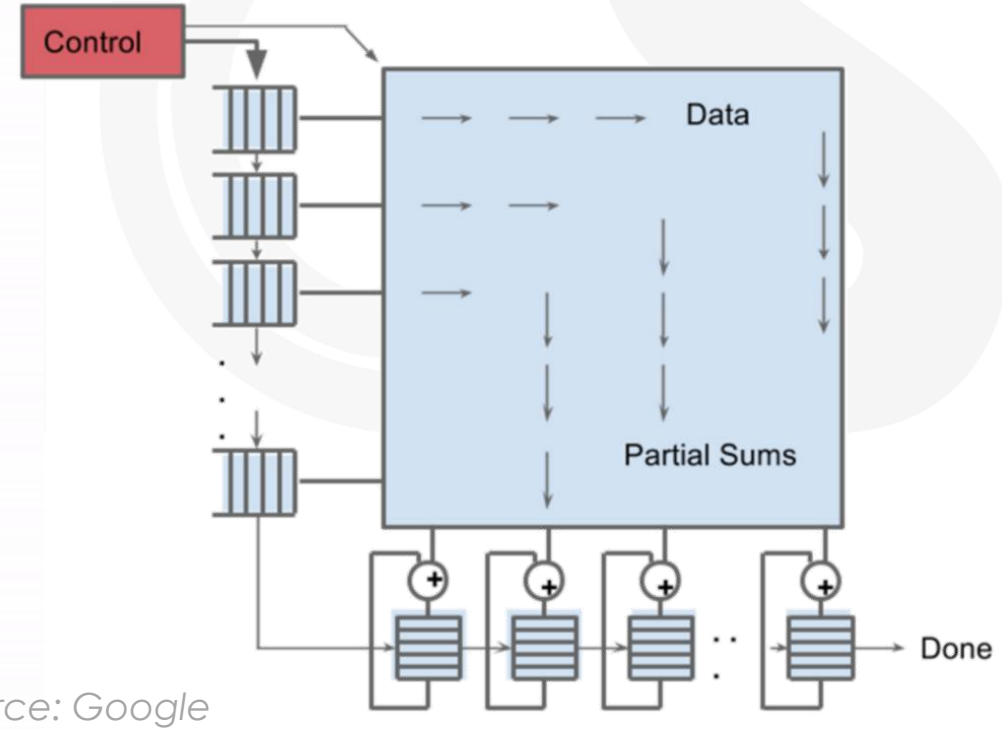
Systolic Array

Multiplying an input vector by a weight matrix with a systolic array

Systolic Array

- **The TPU Systolic Array contains 256x256 ALUs**
  - → 65,536 8-bit MACs every cycle
  - @700MHz → 92 TOPS (46x10^{12} MACs per second)

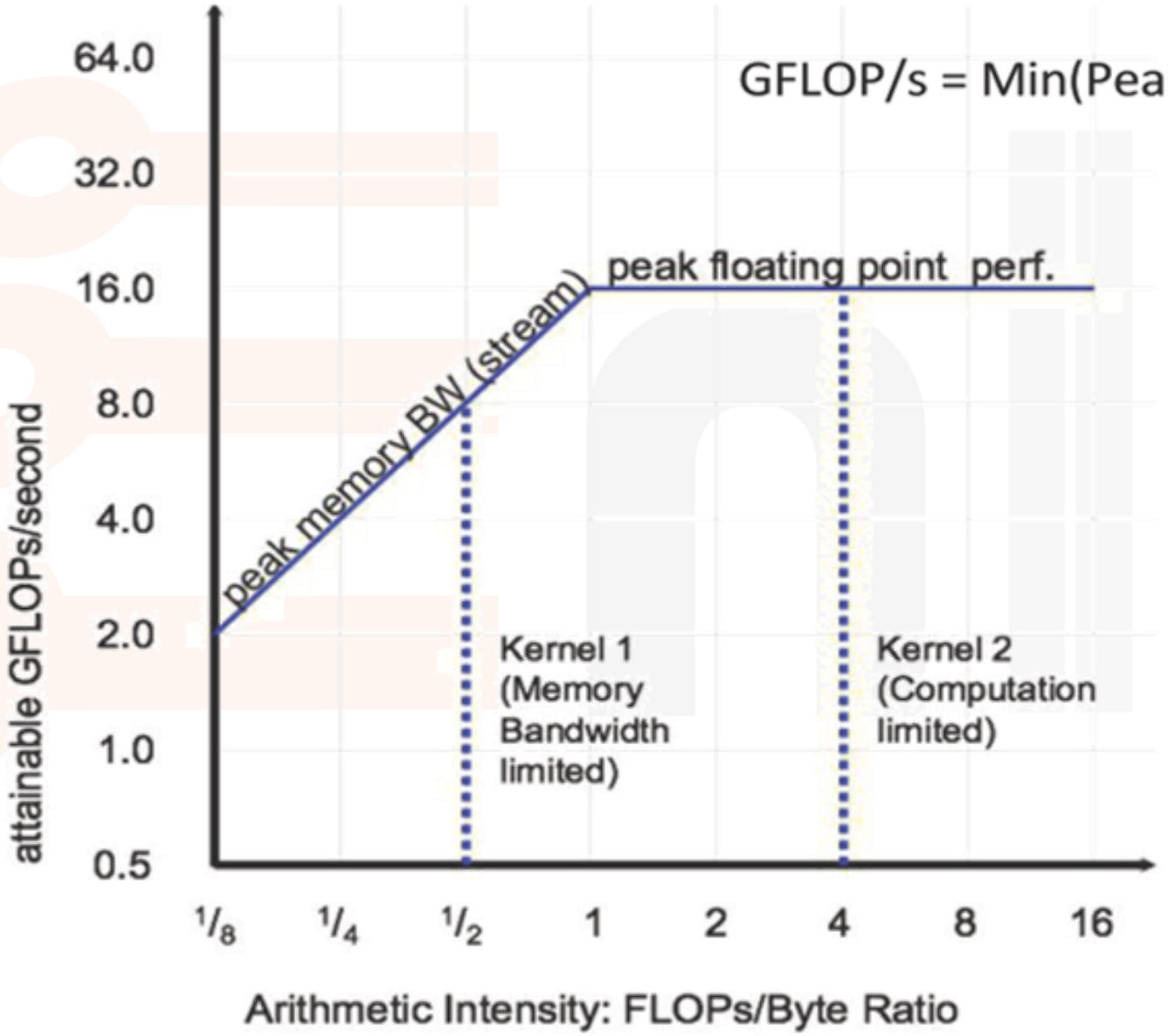




Source: Google

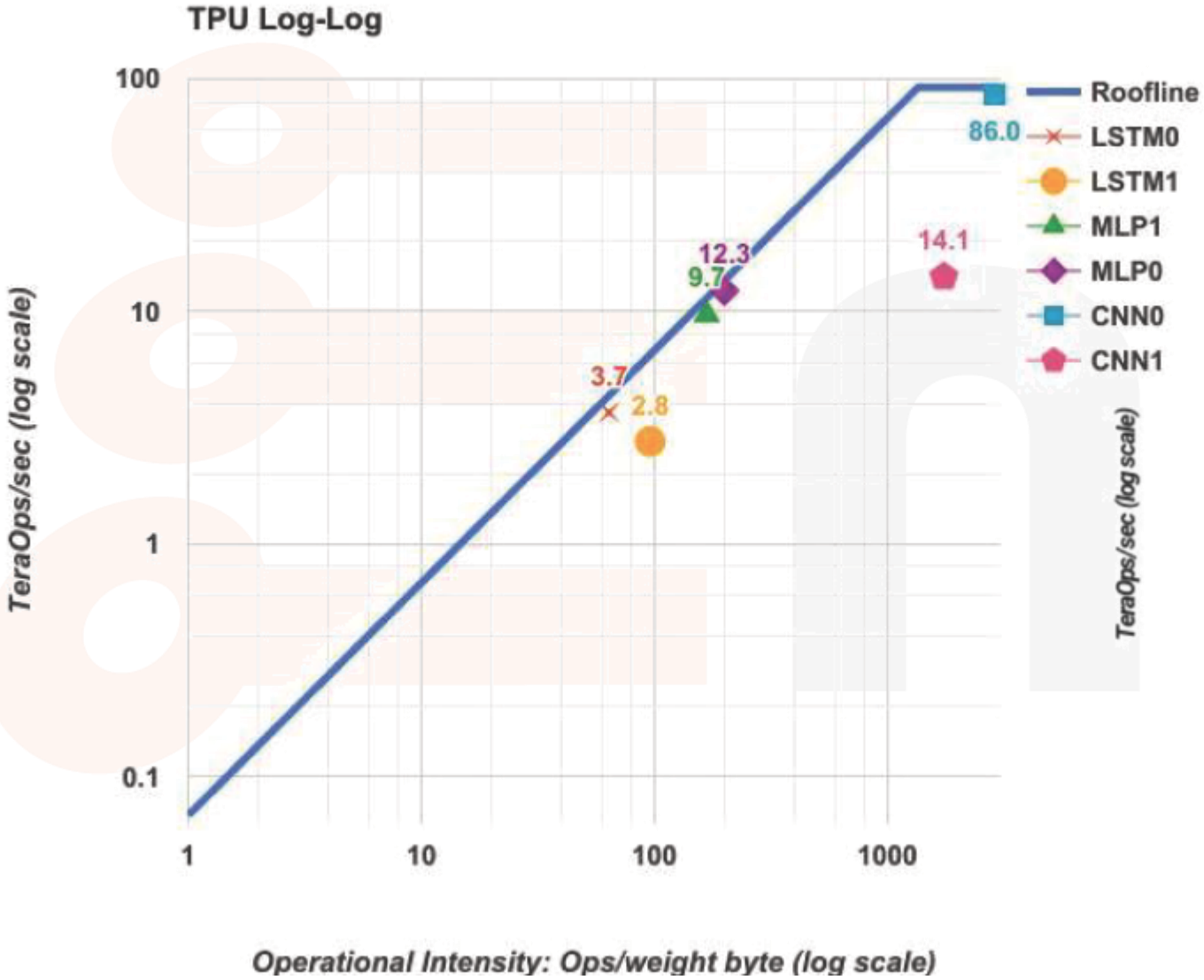
Multiplying an **input matrix** by a **weight matrix** with a systolic array

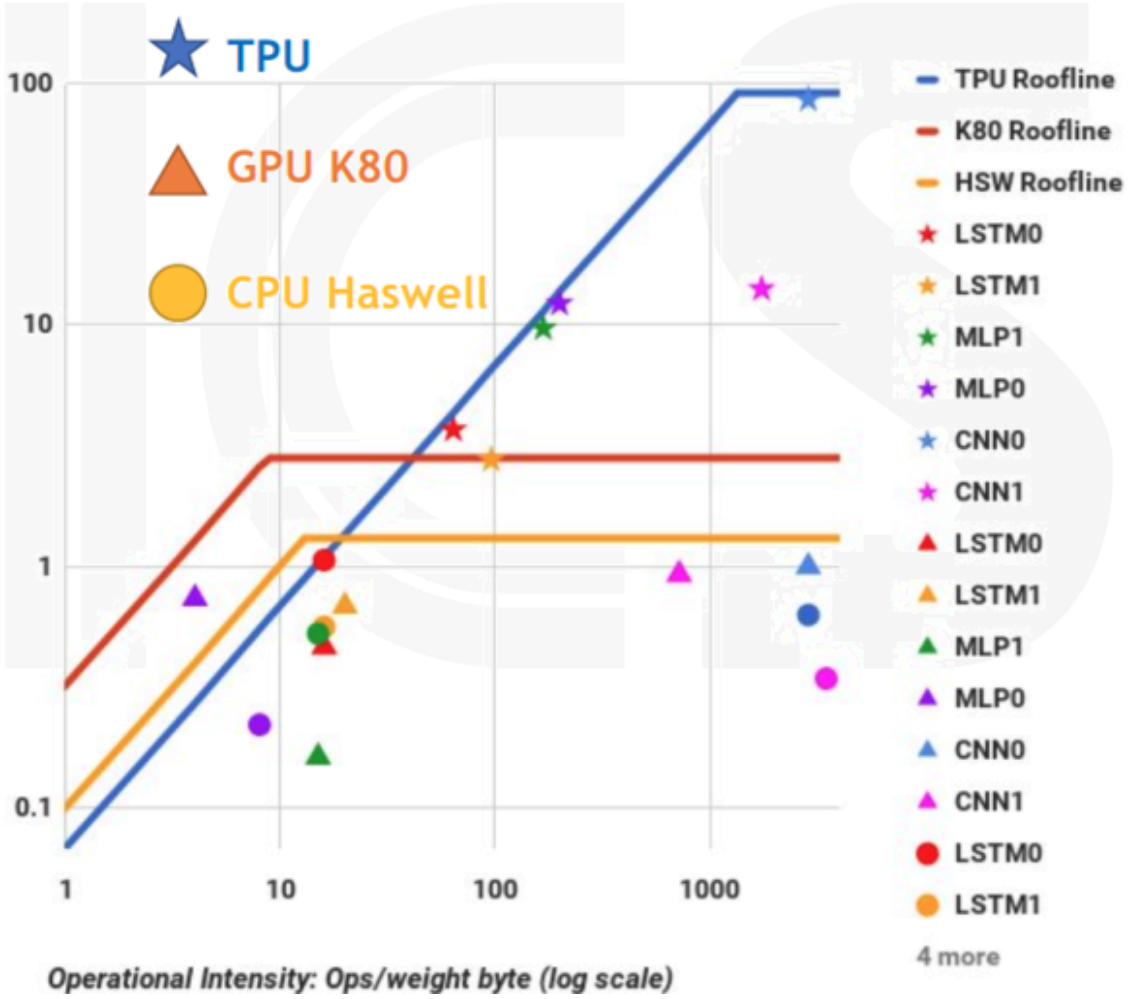
Roofline Model



Source: Patterson

TPU Roofline

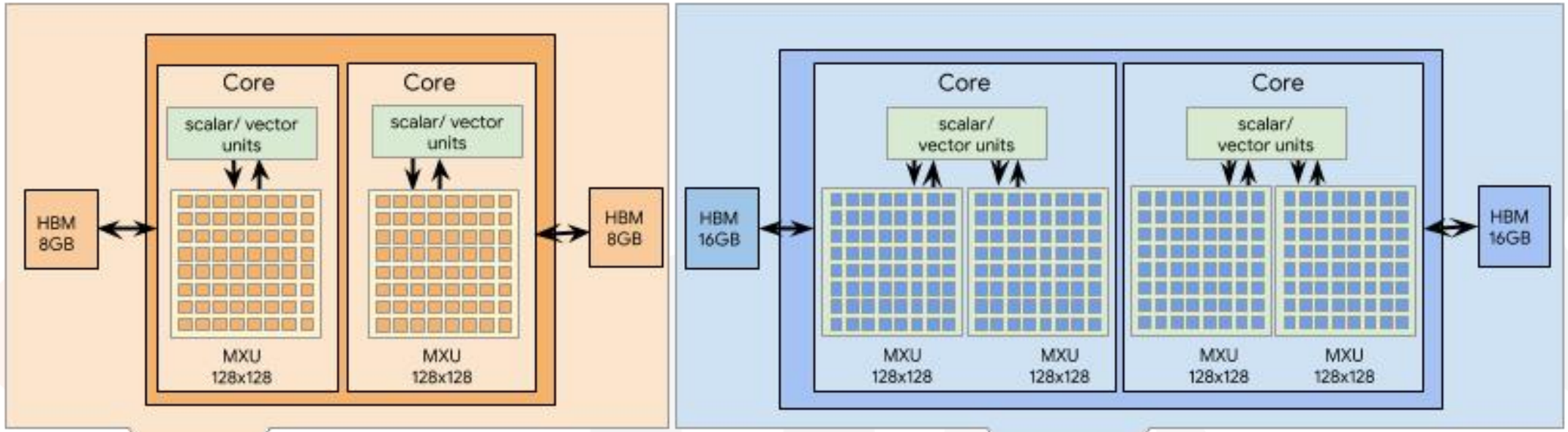


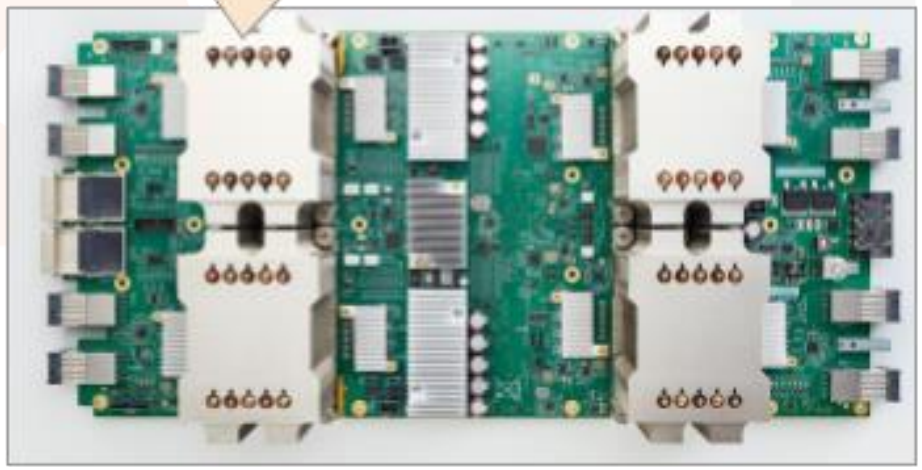


TPU v2/v3: Cloud TPUs

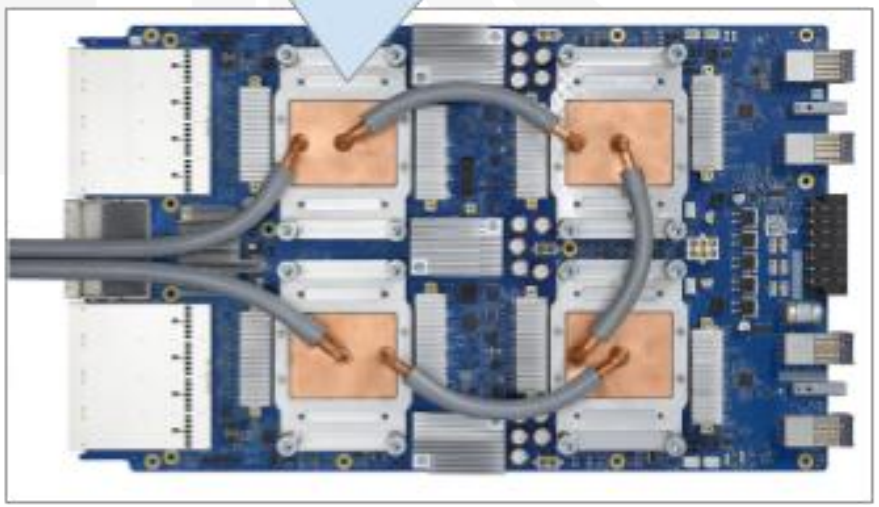
- **Original TPU was for inference and was memory bandwidth limited**
- **TPU v2 (2017) and v3 (2018) for inference and training**
  - Each chip has 2 cores, each board has 4 chips
  - “Pods” of several boards can be connected through high speed interconnect.
- **TPU v2**
  - 8 GiB of HBM for each TPU core
  - One MXU for each TPU core
  - Up to 512 total TPU cores and 4 TiB of total memory in a TPU Pod
- **TPU v3**
  - 16 GiB of HBM for each TPU core
  - Two MXUs for each TPU core
  - Up to 2048 total TPU cores and 32 TiB of total memory in a TPU Pod
- **Each TPU core has scalar, vector and matrix units (MXU)**
  - MXU – 16K (128x128) MACs/cycle @bfloat16 precision with FP32 accumulate

TPU v2/v3: Cloud TPUs





TPU v2 - 4 chips, 2 cores per chip

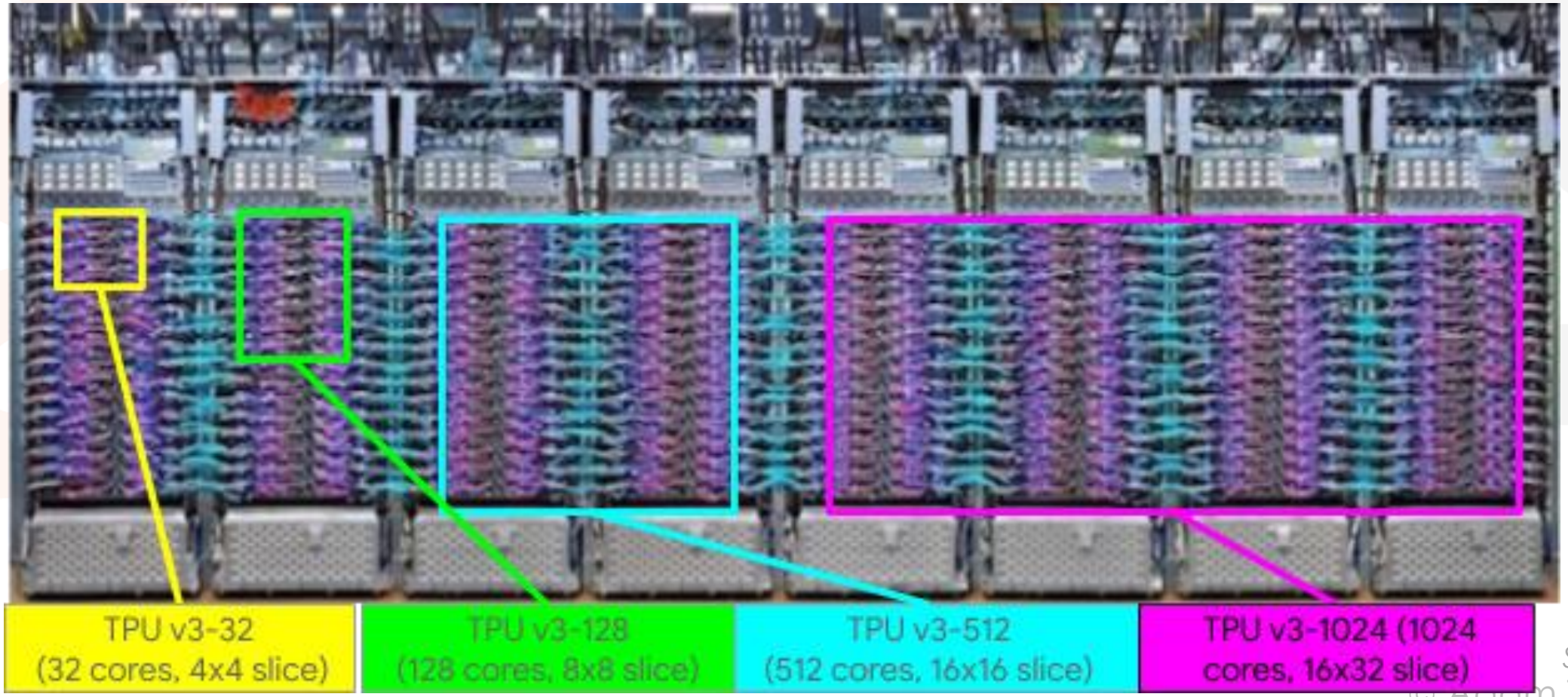


TPU v3 - 4 chips, 2 cores per chip

Source: Google

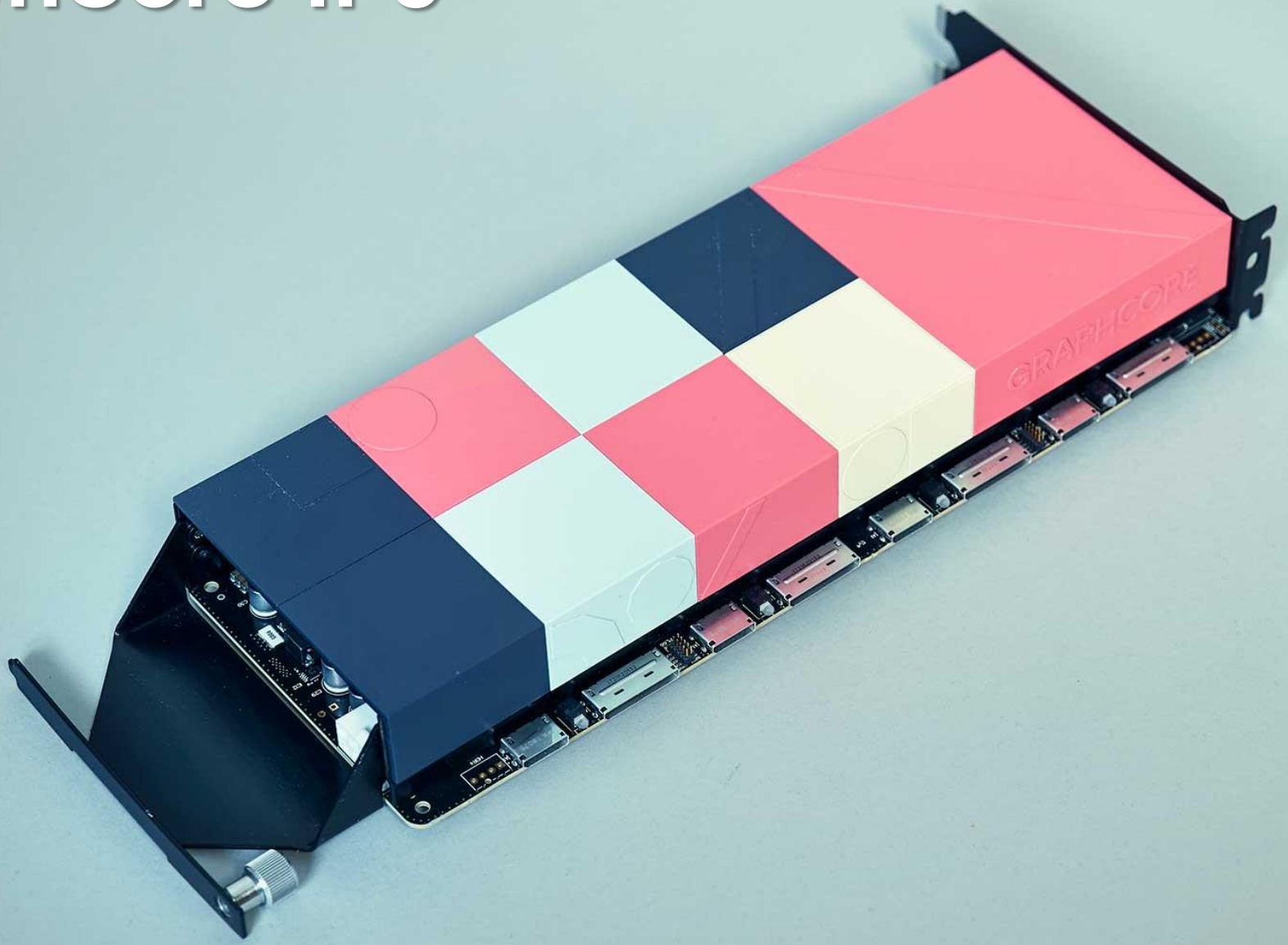
Cloud TPU v3 Pod

- **Maximum configuration: 256 devices**
  - >100 pflops, 32TB TPU Memory (HBM)



Source: Google

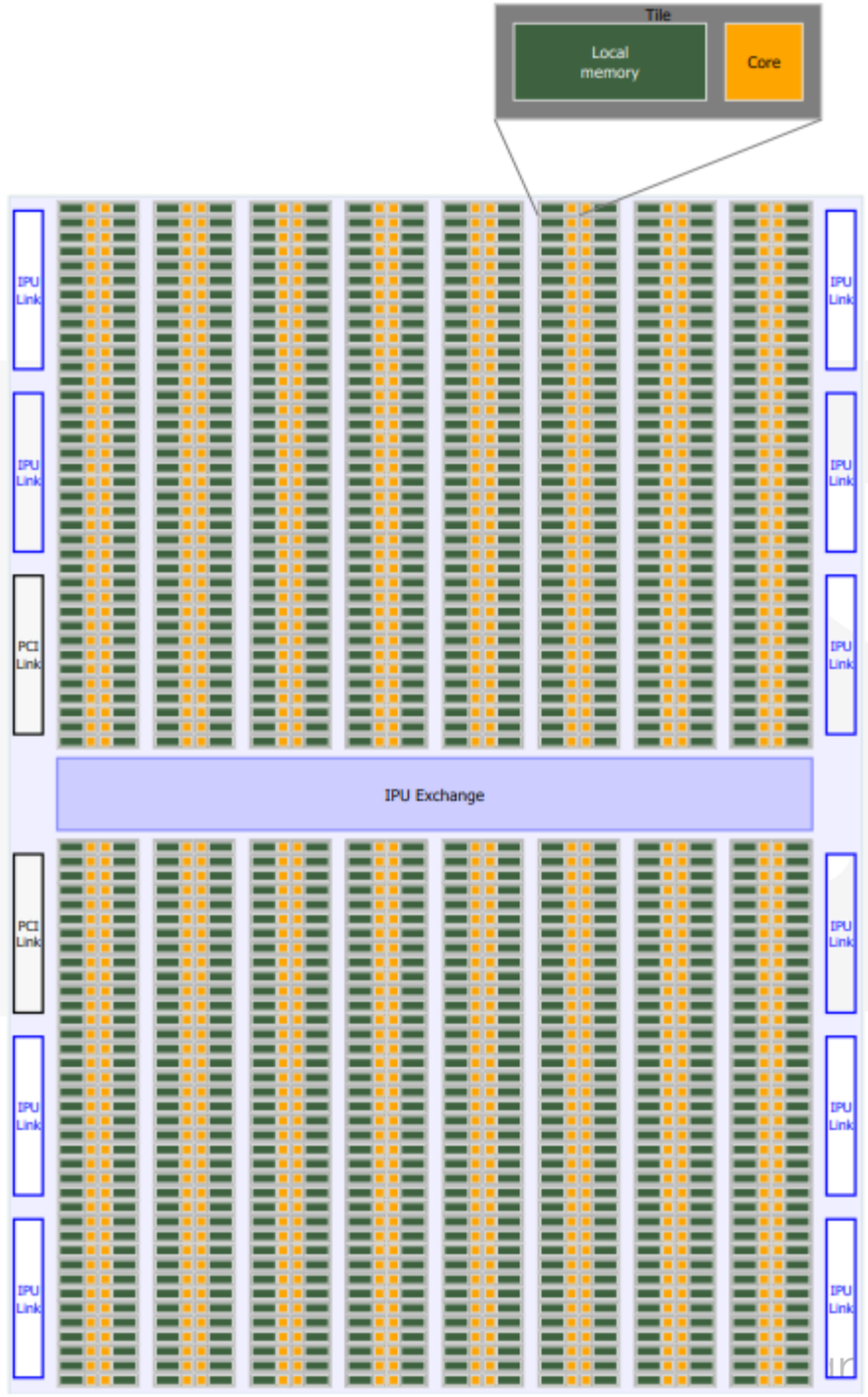
Graphcore IPU



Source: Graphcore

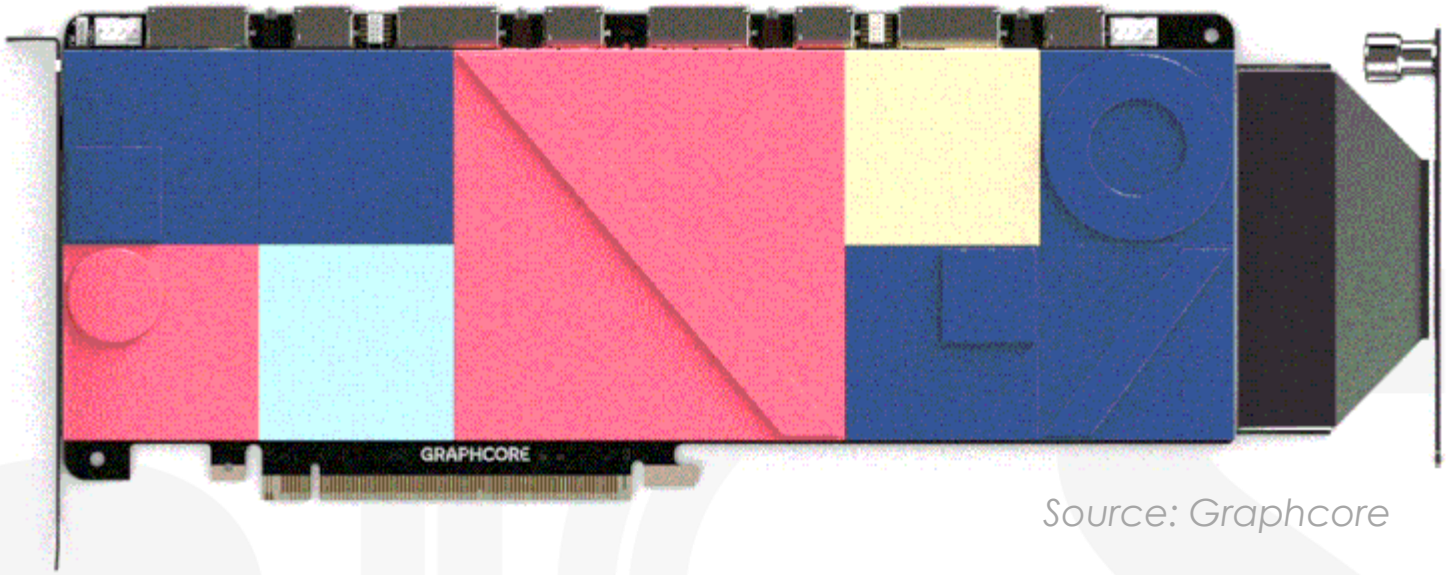
Graphcore IPU

- **Fine-grained Parallelism**
  - Only true MIMD AI accelerator
  - Run individual distinct thread on small block
- **Tiles (PEs)**
  - One computing core
  - 256 KiB local memory
  - Total of 1216 tiles on each IPU
- **IPUs have no shared memory!**
- **Interconnect**
  - Exchange: on-chip interconnect
  - IPU Link: communication between IPUs
  - Two PCIe links to communicate with CPU



Source: Graphcore

Graphcore IPU



Source: Graphcore

- **Chip details:**
  - 16nm, 800mm^2 die
  - over 23 B transistors
  - 300 MB on chip memory, no DRAM
  - 150 W
- **Bulk Synchronous Parallel (BSP) Model**
  - Local computation phase: every process operates on local memory
  - Communication phase: processes exchange data (all-to-all)
  - Barrier synchronization phase: no process continues until all have finished
  - Parallel algorithms of arbitrary complexity can be described in the BSP model

Habana Labs

Source: Habana Labs

Habana Labs Gaudi and Goya

Purpose-Built for AI Inference

GOYA™

Available

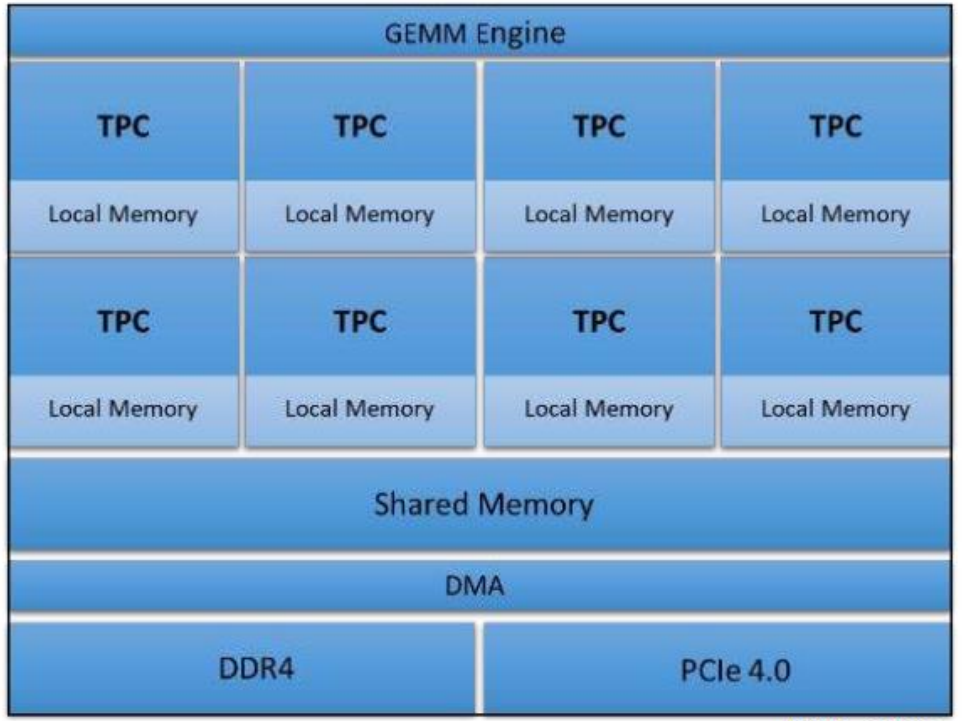
Purpose-Built for AI Training

GAUDI™

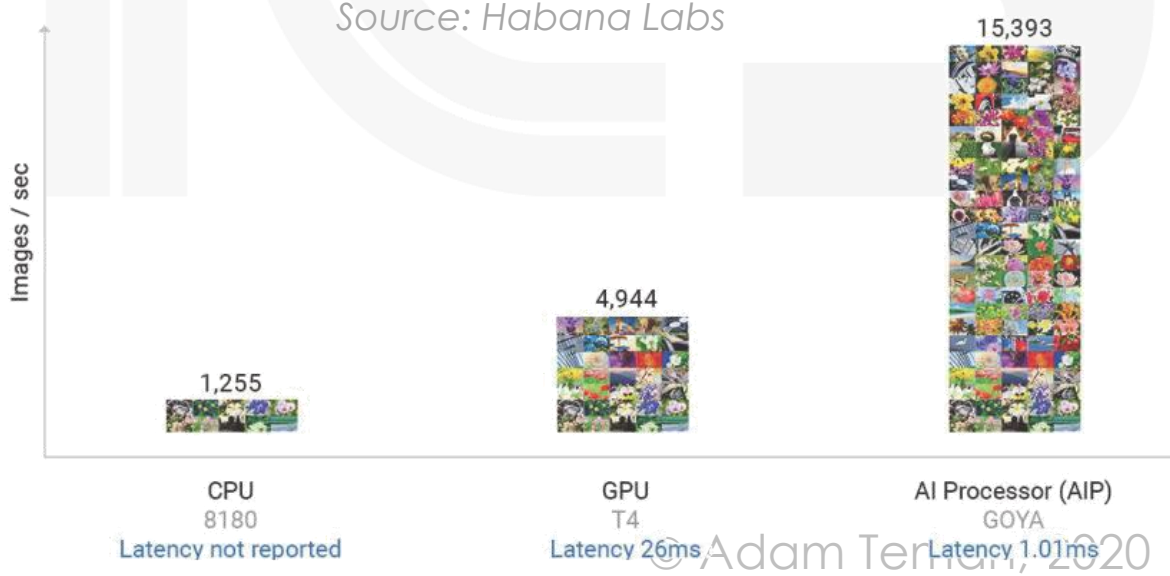
Sampling

Goya Inference Processor

- **GEMM (General matrix and multiply) Engine:**
  - Assumed to be Similar to Tensor Core
- **TPC (Tensor processing Core):**
  - C-programmable VLIW SIMD vector core
  - ISA and dedicated HW for special functions
- **GEMM, TPC and DMA engines use shared SRAM**
- **Mixed Precision data types:**
  - FP32, INT32, INT16, INT8
- **Interconnect**
  - 2 DDR4 channels 40GB/s BW, 16GB
  - PCIe Gen4.0 x16
- **Sophisticated software stack**

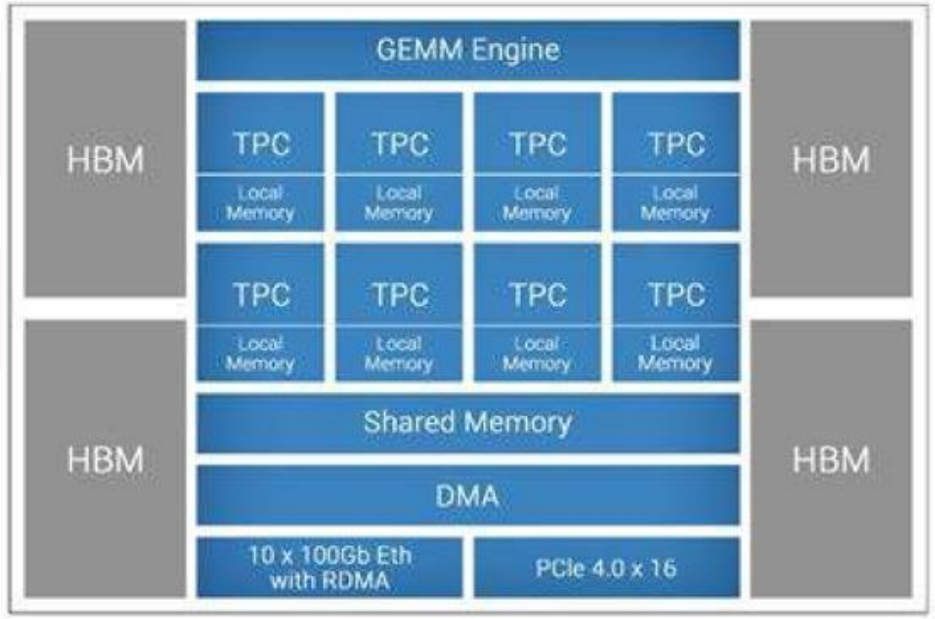


Source: Habana Labs

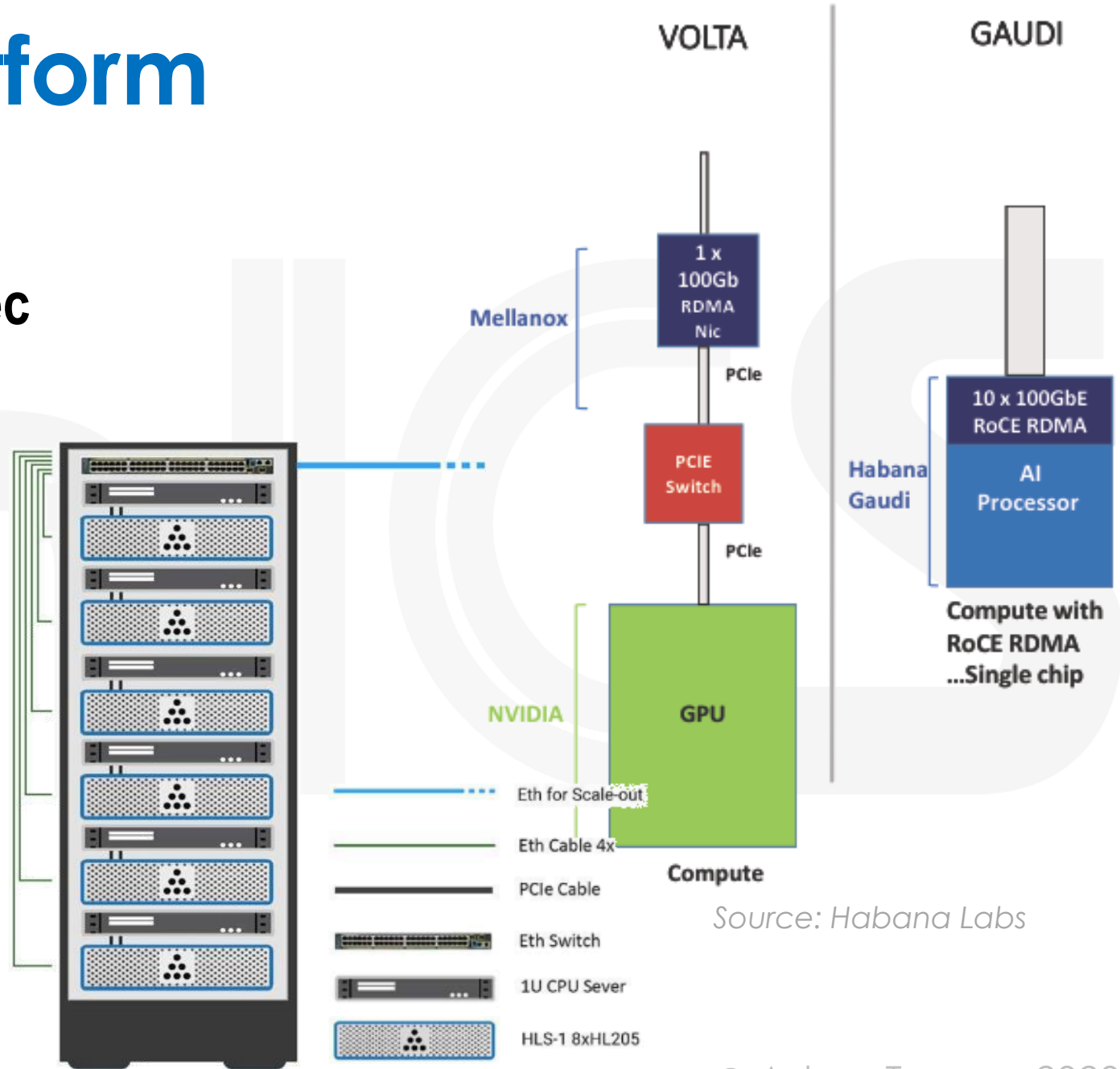


Gaudi Training Platform

- **Architecture similar to Goya**
- **4 HBMs: 2GT/s, 32 GB, BW 1 TB/sec**
- **Non-proprietary interconnect**
  - 10 ports of 100Gb Ethernet
  - Integrated RDMA over RoCE v2



TSMC – 16nm



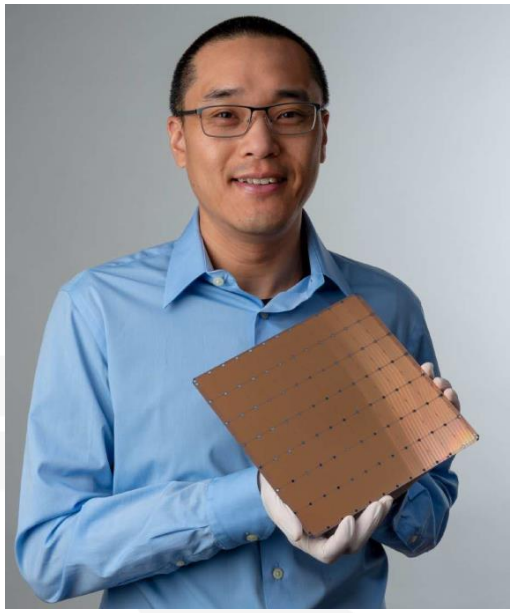
Source: Habana Labs

Cerebras WSE

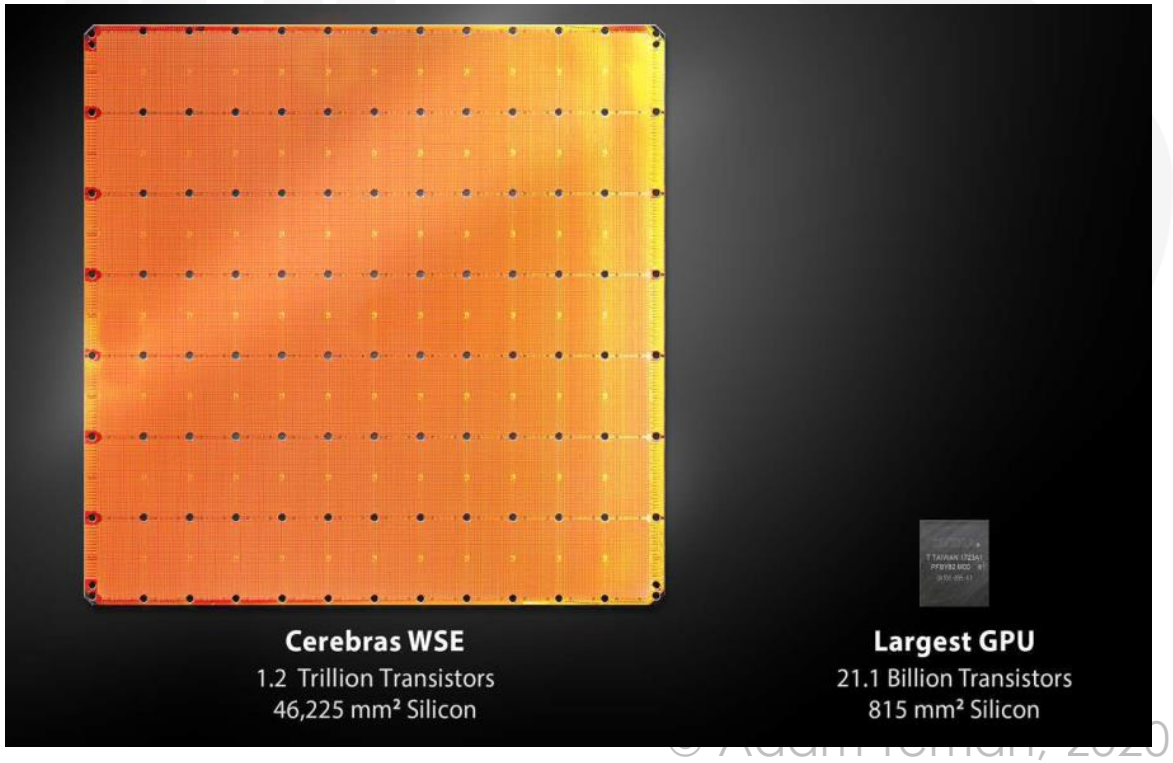
Source: Cerebras

Cerebras Wafer-Scale Engine

- **Largest Chip Ever Built**
  - 46,225 mm^2 (a total of 84 individual 510 mm^2 chips)
  - 1.2 trillion transistors on TSMC 16nm
  - 400,000 AI optimized cores (~0.1 mm^2 each)
  - 18GB distributed SRAM (47kB per core)
  - 9 PByte/s memory bandwidth
  - 100 Pbit/s fabric bandwidth
  - Speculated clock speed ~1 GHz
  - Speculated 5 kW power consumption
- **Each die:**
  - 5076 cores, 225 MB SRAM
  - 40 TFLOPS (FP32)

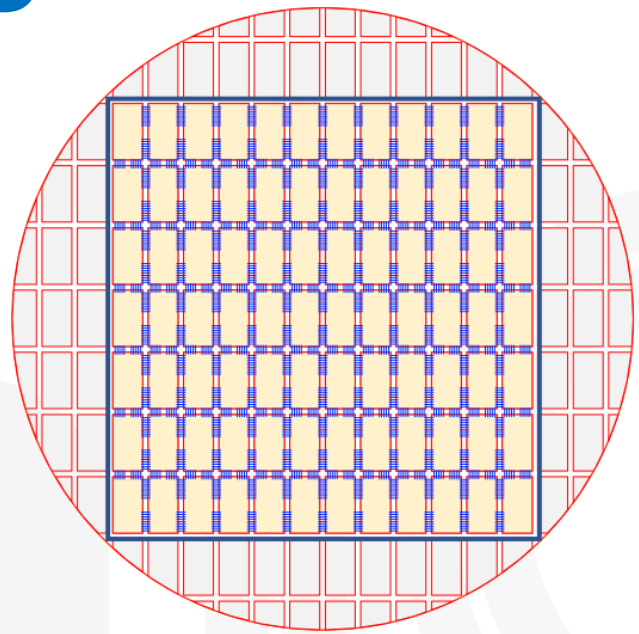


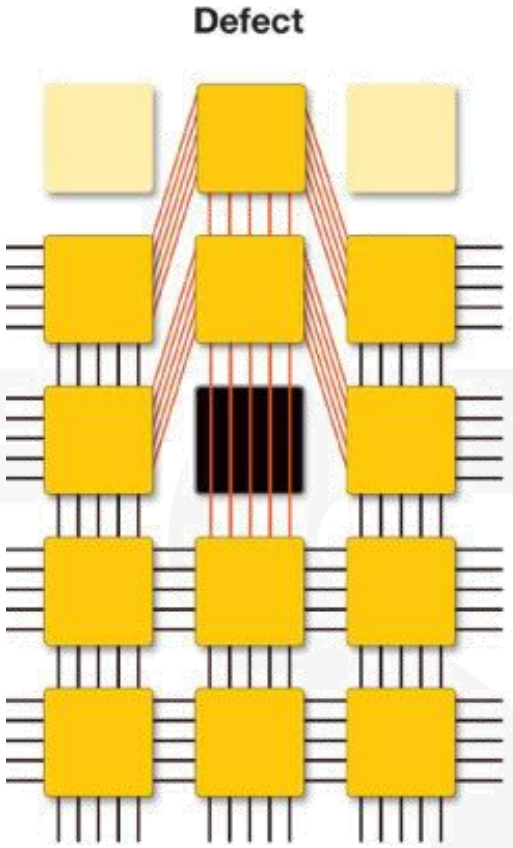
Source: Cerebras

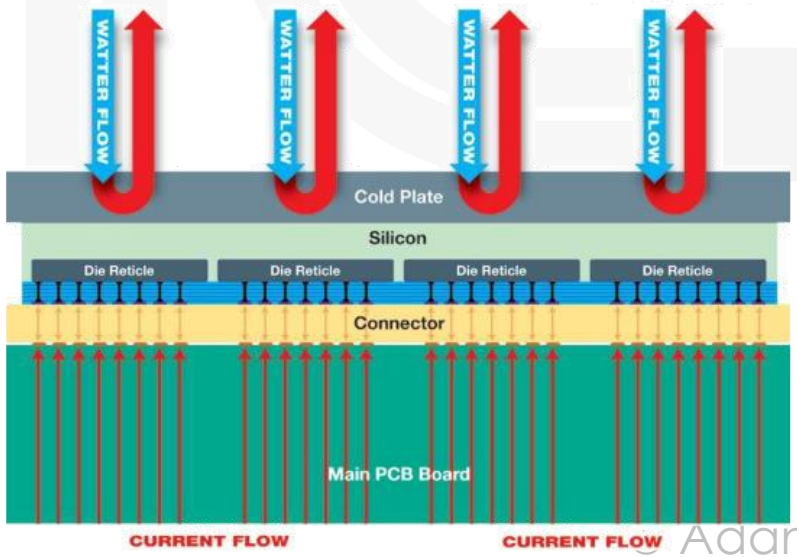


Addressing Challenges of WSE

- **Cross Die Connectivity**
  - Connect across scribe lines
  - Create homogenous array
- **Yield through Redundancy**
  - Uniform small core architecture
  - Redundant cores and links
- **Packaging, Assembly, Power and Cooling**
  - Custom connector from wafer to PCB
  - Custom packaging including machinery
  - Current flow up through PCB
  - Liquid cooling







Source: Cerebras

Hailo-8

Source: TechTime

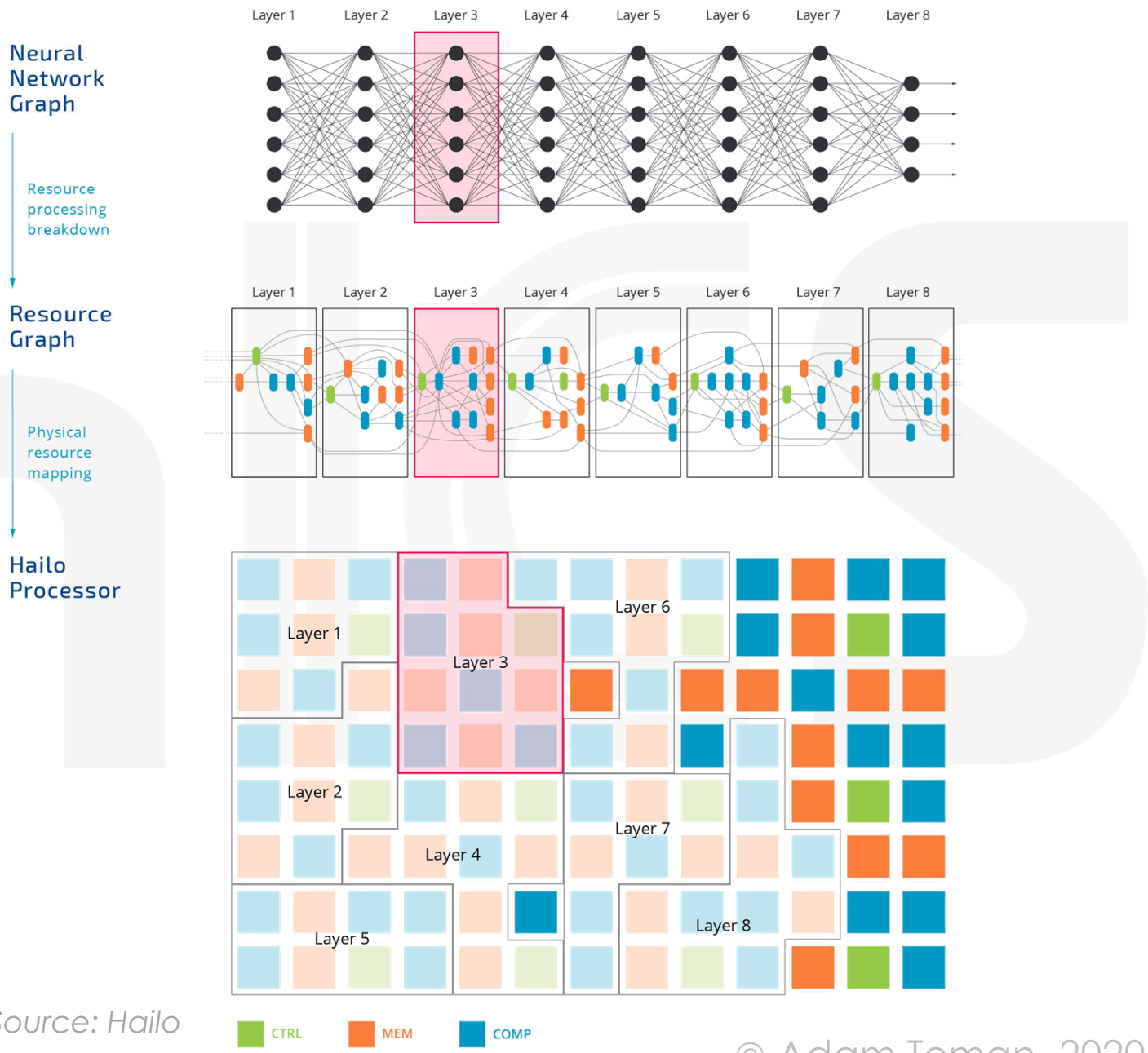
Hailo-8

Source: Vision Systems

- **Hailo-8 Edge Inference Processor**
  - “structure-defined data flow architecture”
  - Distributed on-chip memory fabric
  - Novel control schemes
  - Efficient interconnect
  - Full-stack software toolchain co-designed with the HW architecture
- **Performance**
  - 780 FPS @ 2W and 2.9 TOPS/W on 8-bit ResNet (no pruning)
    (as compared to 656 FPS @ 31W and 0.14 TOPS/W for NVIDIA Xavier AGX)

Hailo-8

- **No external memory,**
  No hardware-defined pipeline
  - Memory, control and compute blocks are distributed
  - Software allocates memory, control and compute blocks to neural networks as required
  - Enables uneven distribution of resources between layers



Source: Hailo

Additional Machine Learning Processors

- **Alibaba Hanguang 800 (2019)**
  - Cloud inference, 80K images/second (ResNet-50), 12nm, 17B transistors,
- **Amazon AWS Inferntia (2019)**
  - Cloud inference, 4 NeuronCores/chip, 128 TOPS
- **Baidu Kunlun (2020)**
  - XPU edge-to-cloud, 260 TOPS, 512 GB/s HBM2, Samsung 14nm
- **Facebook AI Chips (exp. 2020)**
  - Zion – training platform, Kings Canyon – inference, Mount Shasta – video transcoding
- **Huawei Ascend 910 (2019)**
  - 7nm Training chip, DaVinchi architecture, 256 TOPS (FP16), 512 TOPS (INT8), MindSpore AI computing framework
- **Tesla Full Self-Driving (FSD) (2019)**
  - 260 mm^2, 6B transistors, 72 TOPS @ 72W
- **Flexlogix InferX (2020)**
  - Edge inference. eFPGA with nnMAX clusters. 16nm, 54 mm^2, 4K MACs, 13.5W

Small comparison

Chip	Process	Die size mm^2	TDP (W)	On-chip RAM (MB)	Peak FP32 (TFLOPs)	Peak FP16 (TFLOPs)	Mem b/w (GB/s)	IO b/w (GB/s)
Cerebras WSE†	TSMC 16nm	510	180	225	40.6	n/a	0	Unknown
Google TPU v1	28nm	Unknown	75	28	n/a	23 (INT16)	30 (DDR3)	14
Google TPU v2	20nm*	Unknown	200*	Unknown	Unknown	45	600 (HBM)	8*
Google TPU v3	16/12nm*	Unknown	200*	Unknown	Unknown	90	1200 (HBM2)*	8*
Graphcore IPU	16nm	800*	150	300	Unknown	125	0	384
Habana Gaudi	TSMC 16nm	500*	300	Unknown	Unknown	Unknown	1000 (HBM2)	250
Huawei Ascend 910	7nm+ EUV	456	350	64	Unknown	256	1200 (HBM2)	115
Intel NNP-T	TSMC 16FF+	688	250	60	Unknown	110	1220 (HBM2)	447
Nvidia Volta	TSMC 12nm FFN	815	300	21.1	15.7	125	900 (HBM2)	300
Nvidia Turing	TSMC 12nm FFN	754	250	24.6	16.3	130.5	672 (HBM2)	100

*Speculated

† Figures given for a single chip

Source: James W. Hanlon

Deep Learning Software Stack

So, you thought hardware was confusing…

- **Let's *try* to make some sense of the deep learning software stack.**
  - I can't be sure that I got this categorization correct, but here goes:

- **From the bottom up, we have:**
  - Compilers (GCC, LLVM,…)
  - Hardware backends (CUDA, CuDNN, OpenCL, …)
  - Intermediate Libraries (CuDNN, CuBLAS, TVM+NNVM, …)
  - Software frontend (Python, C, C++, …)
  - Deep Learning Framework (TensorFlow, PyTorch, …)
  - Model Exchange (ONNX)
  - High Level Library (Keras)

The Backend

- **So before discussing the (more interesting) frontend frameworks, let's try to introduce the backends**
  - A compiler (e.g., GCC, LLVM) turns a high-level language into instructions that run on the hardware.
  - For parallelization, hardware-aware abstractions have been introduced. The most popular ones are CUDA (NVIDIA) and OpenCL.
  - And NVIDIA has introduced libraries to abstract away some of the CUDA. These include CuDNN and CuBLAS.
  - Many specialized hardware providers also use intermediate graph representations in (mostly proprietary) APIs.
  - TVM+NNVM is an attempt to make a generic intermediate graph compiler.

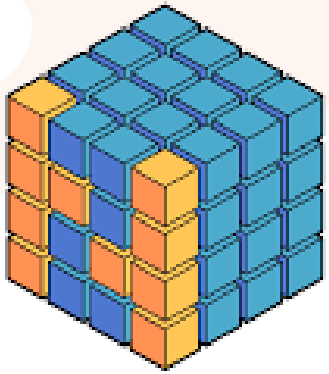
The Frontend

- Or let's just call it by its name…

Python*

- Just learn Python and you'll be a much more happy engineer.

- And while you're at it, don't forget to import NumPy and MatPlotLib…



matplotlib

*almost as popular and useful as Salamandra!

The Framework

- **On top of Python (or other language), there are many popular DL frameworks:**
  - TensorFlow – Created by Google
  - PyTorch – Developed by Facebook (successor of Torch)
  - Caffé2 – Successor of Caffé, now merged with PyTorch.
  - Theano – Developed by U. Montreal. Now dead…
  - CNTK – a.k.a. The Microsoft Cognitive Toolkit
  - MXNet – Created by Apache
- **And on top of them you can use a Keras**
- **And for model exchange use ONNX**

TensorFlow

PYTORCH

Caffe

theano

Microsoft Cognitive Toolkit

mxnet

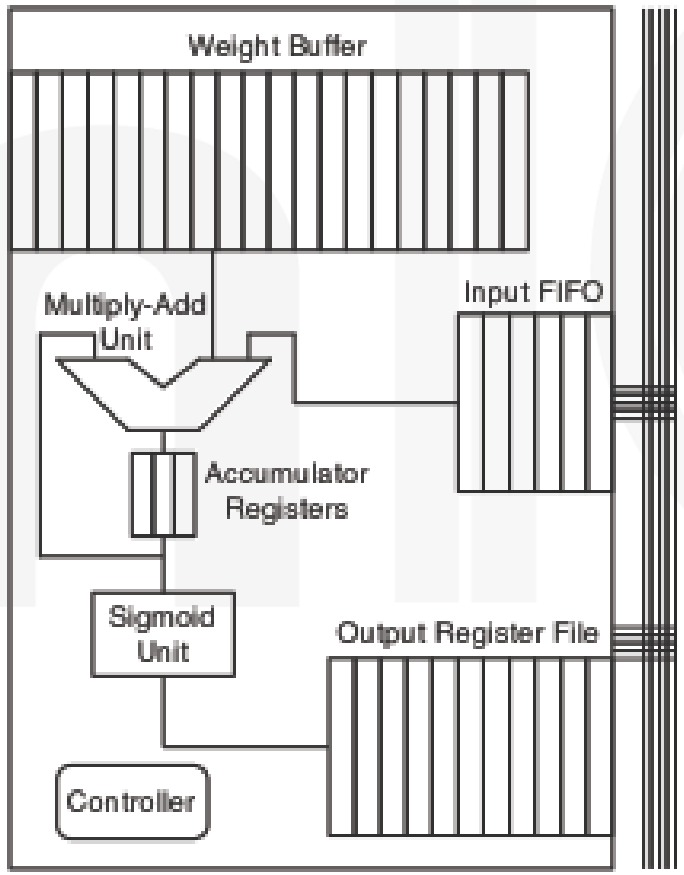
ONNX

Keras

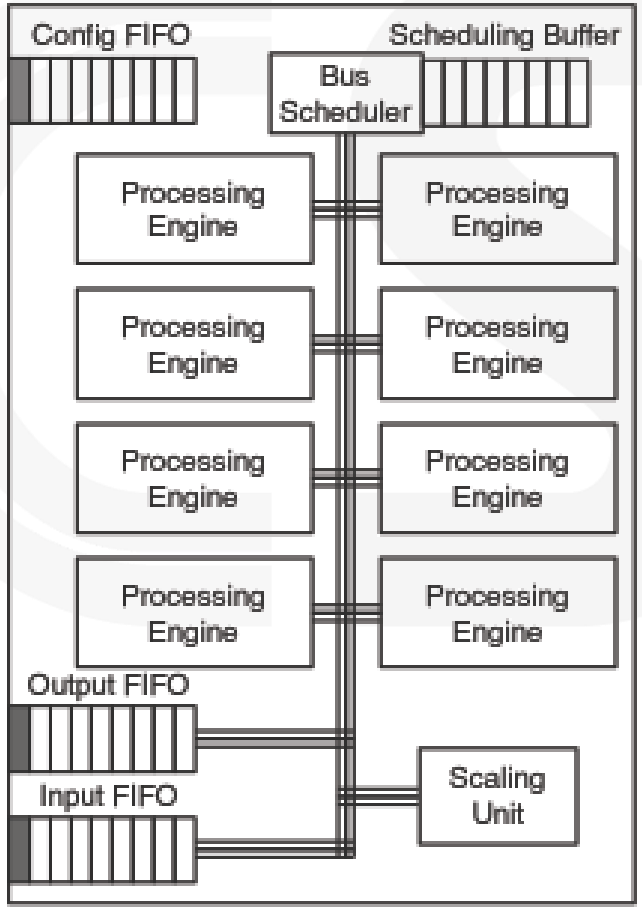
Specialized Research ASICs

General Approach: The NPU

- **The Neural Processing Unit (2012), is a general architecture for spatial neural network acceleration.**
- **A processing engine (PE):**
  - Weight buffer
  - Input buffer
  - Output buffer
  - MAC (+activation)
- **Several PEs connected with some scheduling and control**



Single Processing Element

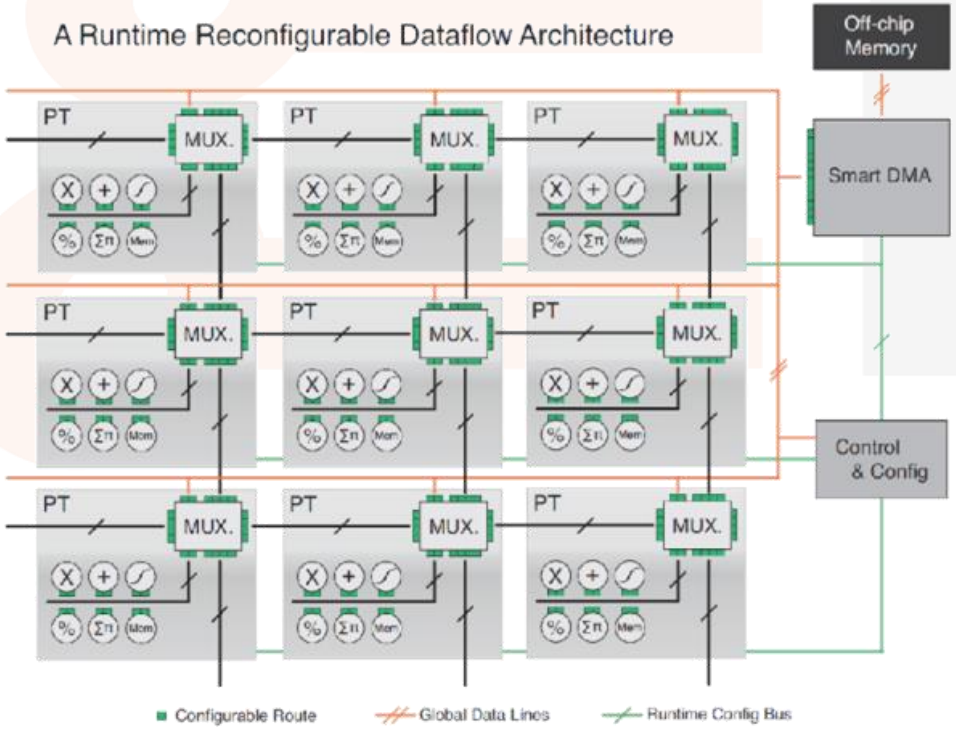


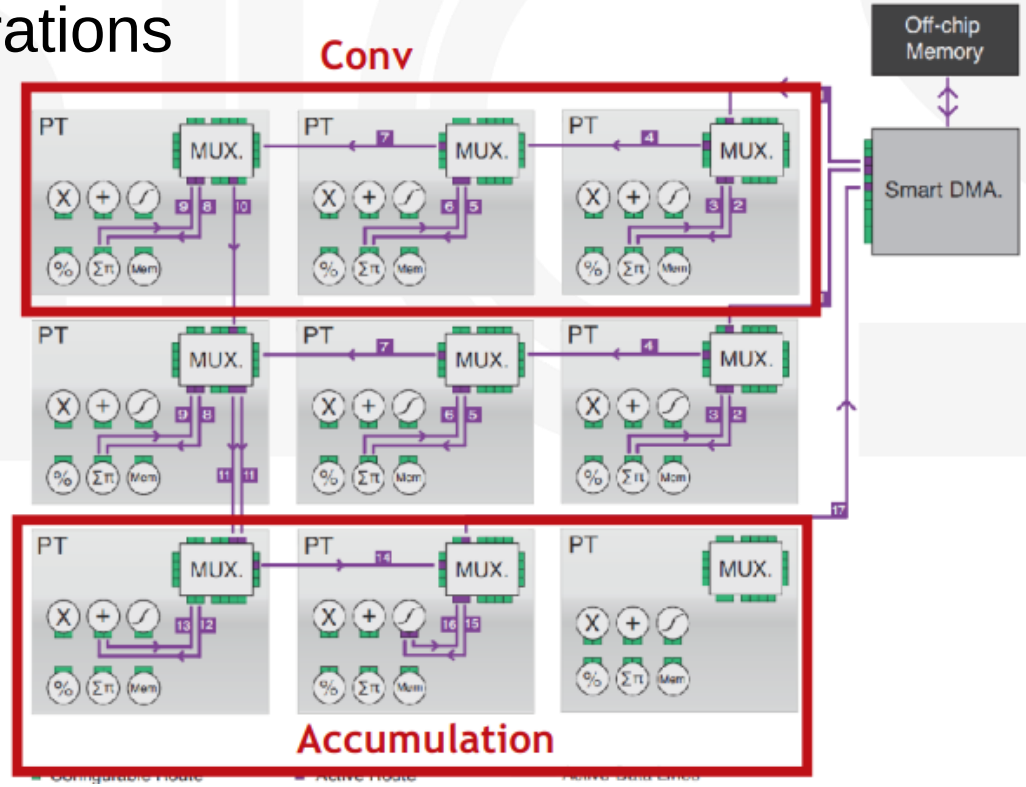
8-PE NPU

Source: Esmaeilzadeh, Micro 2012

NeuFlow (Farabet, 2011)

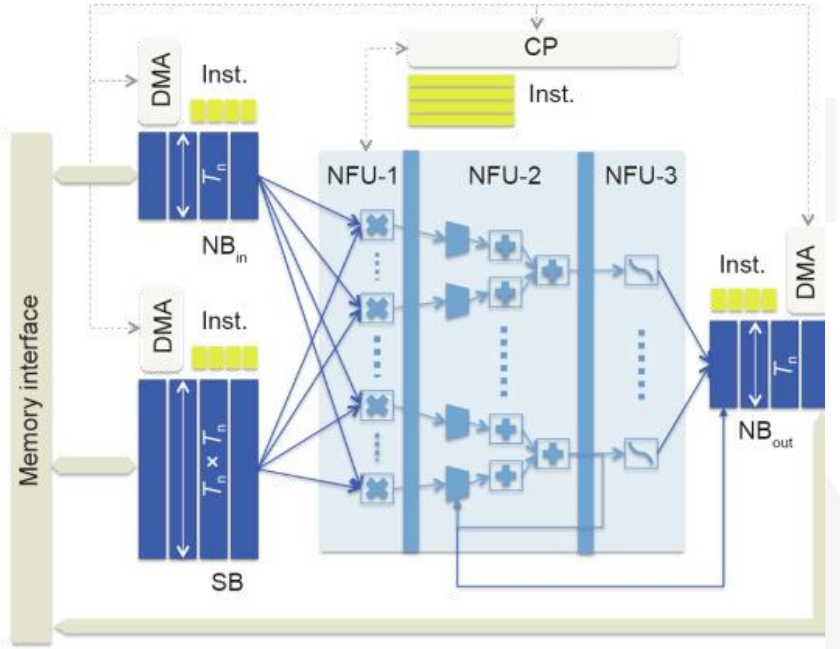
- **Specialization:**
  - Customized logic to efficiently map CONV to hardware with more parallelism
- **Flexibility:**
  - Runtime reconfigurable bus and operations





DianNao Family (2014-15)

- **DianNao (2014)**
  - Separate buffers for input neurons (NBin), output neurons (NBout) and synaptic weights (SB)
  - Neural functional unit (NFU) for computation
- **DaDianNao (2014)**
  - Integrates eDRAM to avoid main memory access
  - Tiles with NFU and four eDRAM banks (2MB)
- **ShiDianNao (2015)**
  - Low power CNN accelerator
  - Entire model fits in SRAM

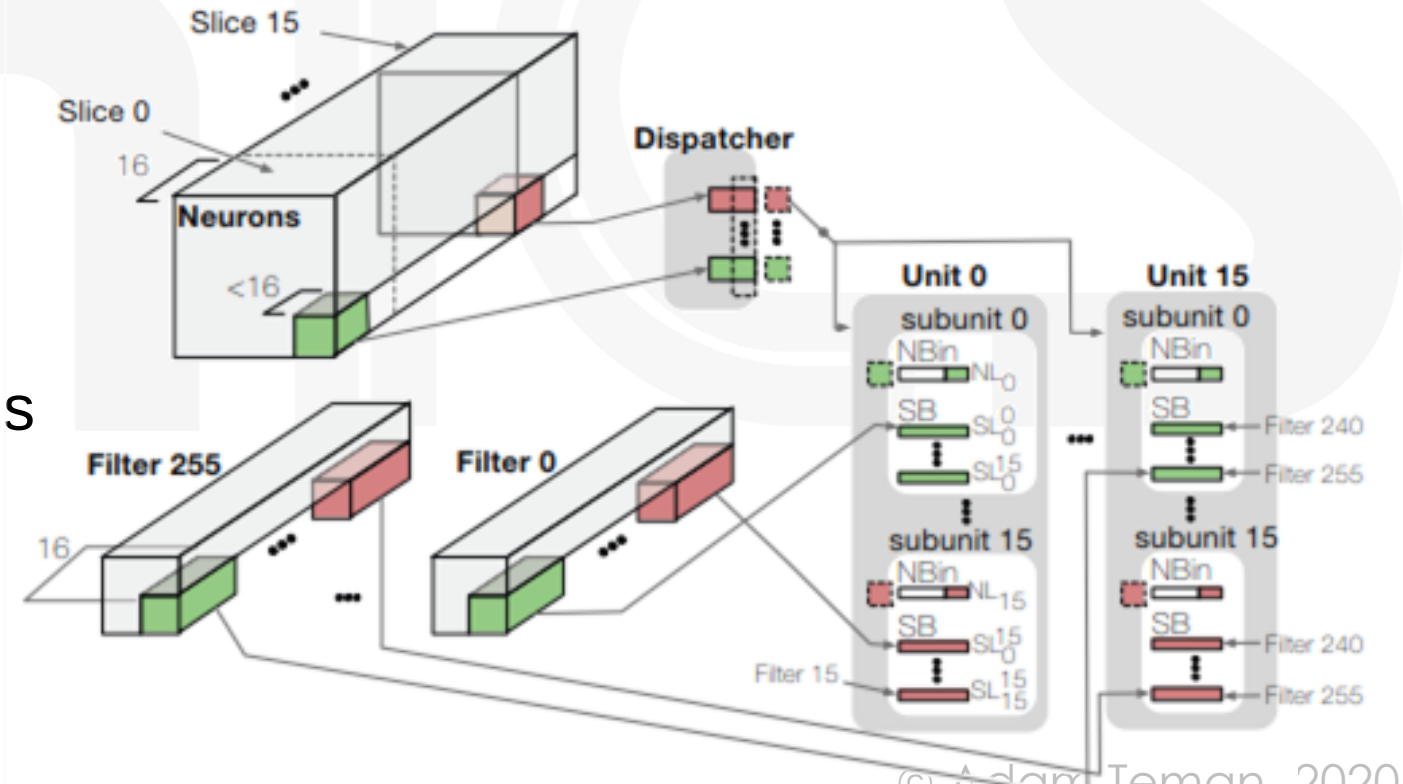


Source: Chen

	Process (nm)	Peak Performance (GOPs)	Peak Power (W)	Area (mm2)
DianNao	65	452	0.485	3.02
DaDianNao	28	5585	15.97	67.73
ShiDianNao	65	194	0.32	4.86

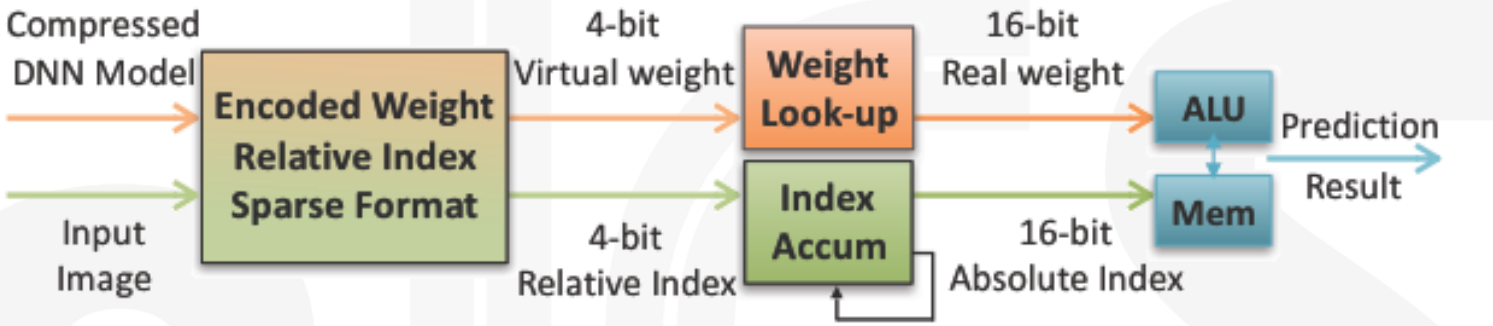
Cnvlutin (2016)

- **Ineffectual-Neuron-Free Deep Neural Network Computing**
  - A value-based approach to dynamically eliminate ineffectual multiplications
  - Zero-skipping on dynamic activations
- **Slight modification to DaDianNao:**
  - Split the processing of filters so they are not in lock step.
  - Encode the zero skipping of the output of the previous layer
  - Independently skip multiplications in each computation lane.

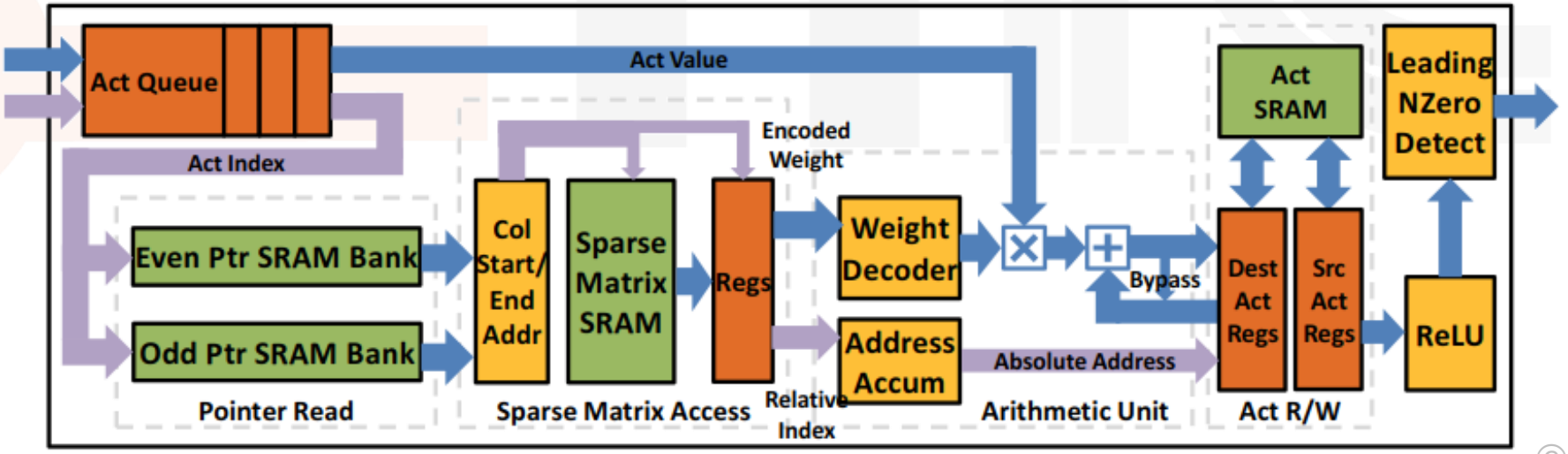


Efficient Inference Engine (2016)

- **EIE takes advantage of:**
  - Static weight sparsity
  - Dynamic activation sparsity
  - Relative indexing
  - Weight sharing
  - Extremely narrow weights (4-bits)

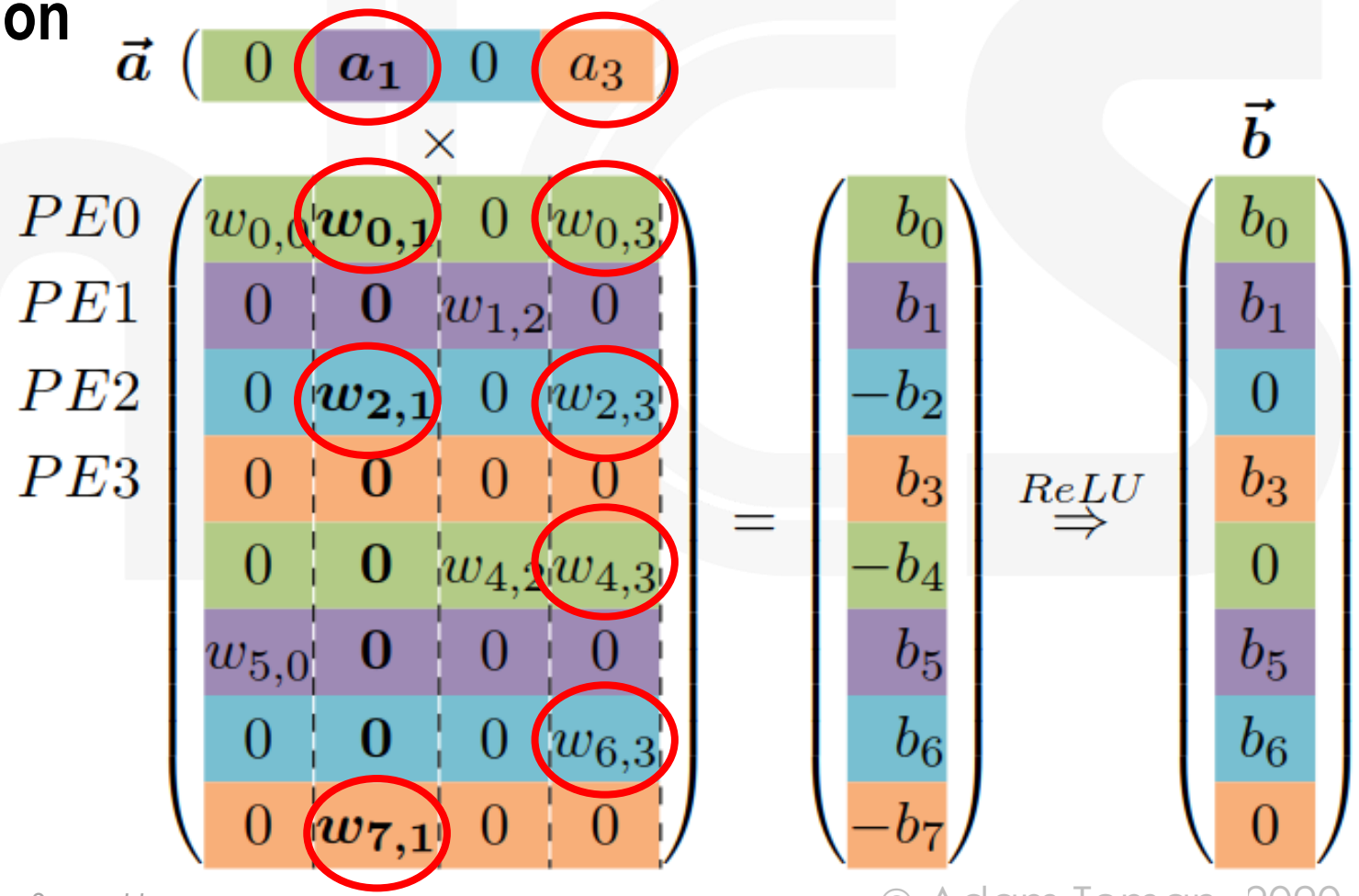


Source: Song Han



EIE Sparse Matrix Multiplication

- **Distribute the activations (a), weights (w) and outputs (b) between N=4 PEs.**
- **Broadcast next non-zero activation**
  - Skip a_0, broadcast a_1
- **Multiply by non-zero weights**
  - $PE0$: $b_0 = b_0 + a_1 w_{0,1}$
  - $PE2$: $b_2 = b_2 + a_1 w_{2,1}$
  - $PE7$: $b_7 = b_7 + a_1 w_{7,1}$
- **And to the next activation (a_3)**
  - $PE0$: $b_0 = b_0 + a_3 w_{0,3}$
  - $PE2$: $b_2 = b_2 + a_3 w_{2,3}$
  - $PE4$: $b_4 = b_4 + a_3 w_{4,3}$
  - $PE6$: $b_6 = b_6 + a_3 w_{6,3}$



Source: Song Han

EIE Compressed Sparse Column (CSC) Format

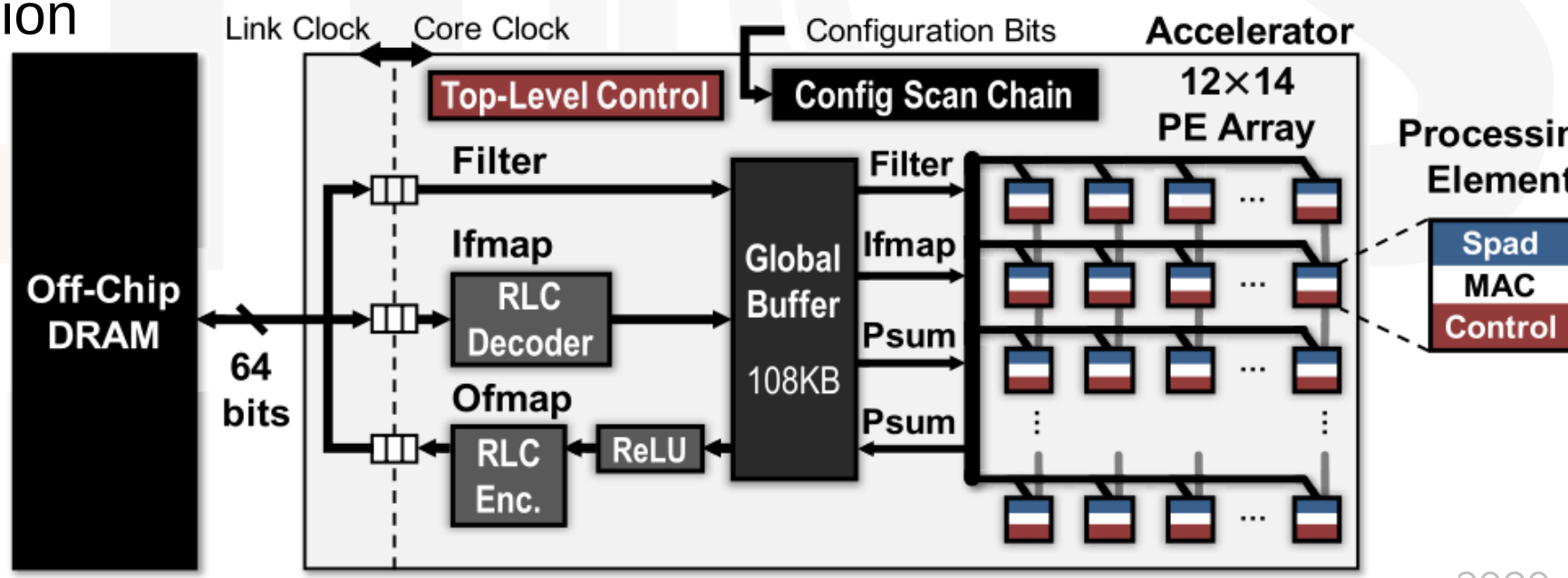
- **For each column of the weight matrix, store:**
  - A vector $\boldsymbol{v}$ with the non-zero weights
  - An equal-length vector $\boldsymbol{z}$ that encodes the number of zeros before the corresponding entry in $\boldsymbol{v}$
  - Each entry of $\boldsymbol{v}$ and $\boldsymbol{z}$ is represented by a four-bit value.
  - If >15 zeros appear before a non-zero entry, we add a zero in vector $\boldsymbol{v}$.
- **For example:**

$$[0, 0, 1, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \mathbf{0}, 0, 0, 3]$$

  - $\boldsymbol{v} = [1, 2, 0, 3]$
  - $\boldsymbol{z} = [2, 0, 15, 2]$
- **Now, simply multiply each non-zero activation by all of the non-zero elements in its corresponding column.**

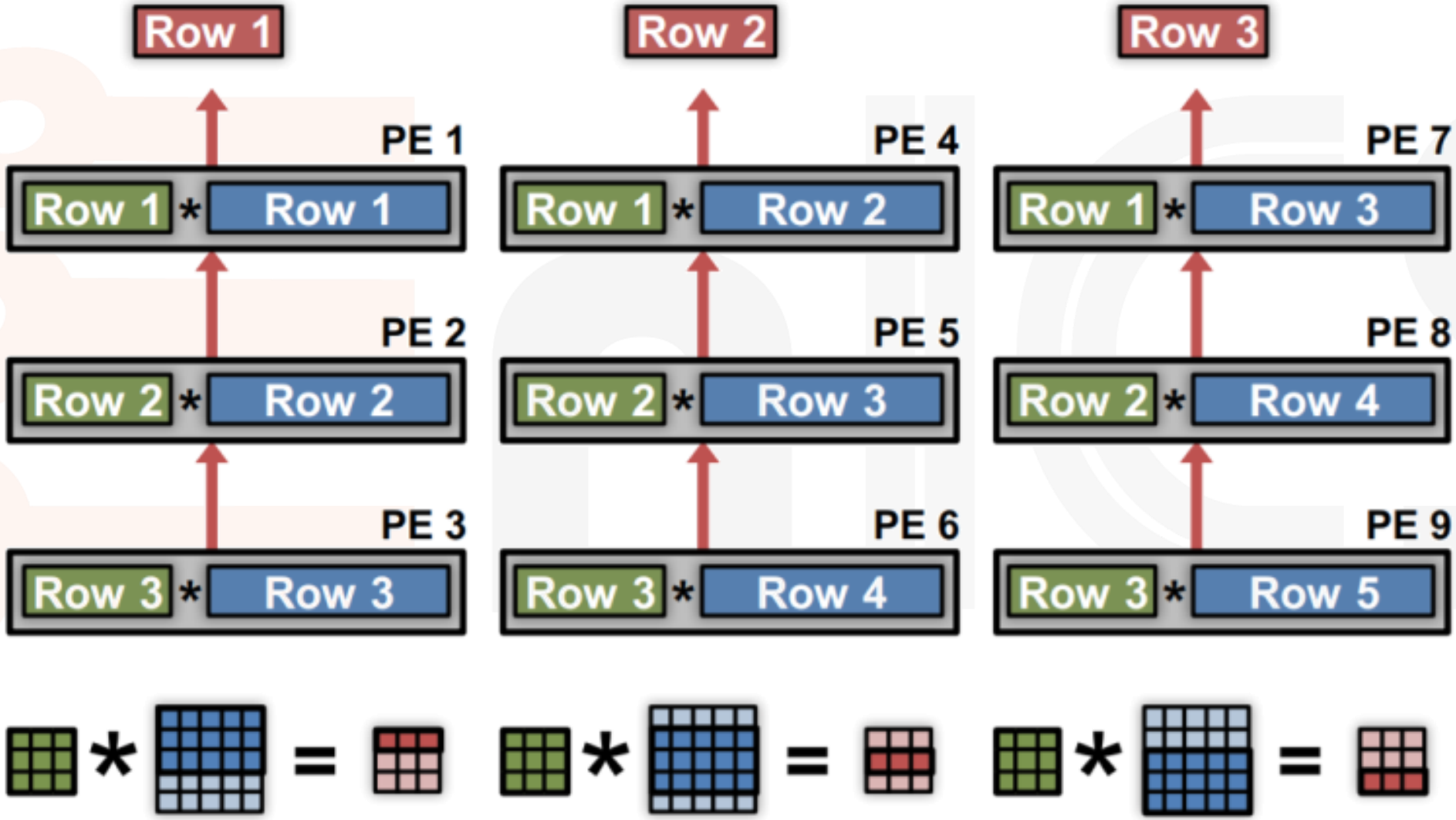
Eyeriss (2016)

- **High-throughput CNN inference optimized for energy efficiency**
  - Spatial architecture with 168 PEs
  - 4-level memory hierarchy (DRAM, Global buffer, inter-PE comm, scratchpads)
  - Energy-efficient Row Stationary (RS) CNN dataflow
  - Network-on-chip with multicast and point-to-point data delivery
  - Run-length compression and PE data gating for energy-efficiency



Source: Yu-Hsin Chen

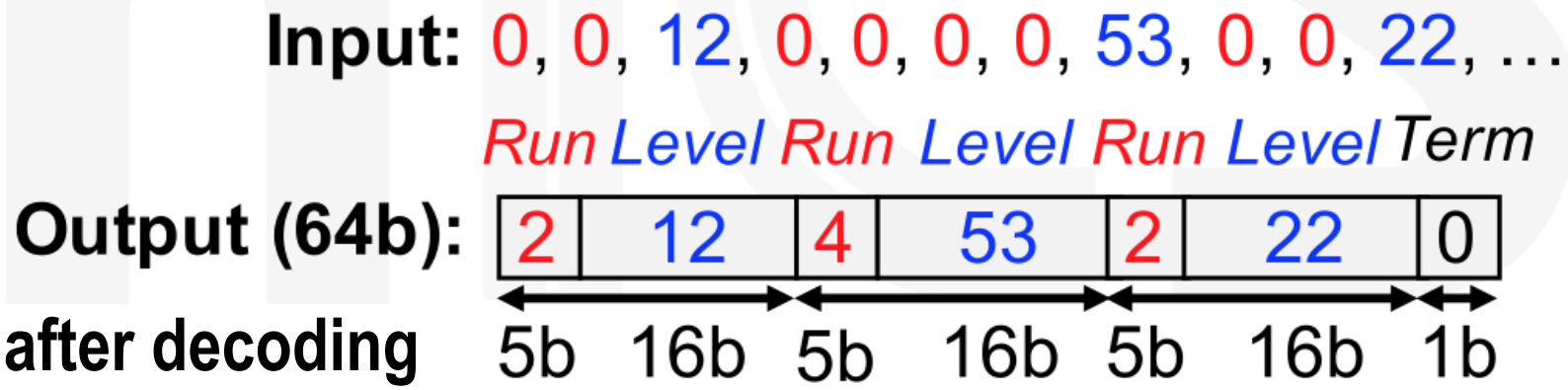
Eyeriss Row Stationary Dataflow



Source: Yu-Hsin Chen

Eyeriss RLC Encoding

- **Data is stored in DRAM in run length compression (RLC) format**
  - Number of zeros (run) encoded in 5-bits
  - Value of non-zero (Level) encoded in 16-bits
  - Three pairs are packed into a 64-bit word
  - Only adds 5%-10% overhead to the theoretical entropy limit

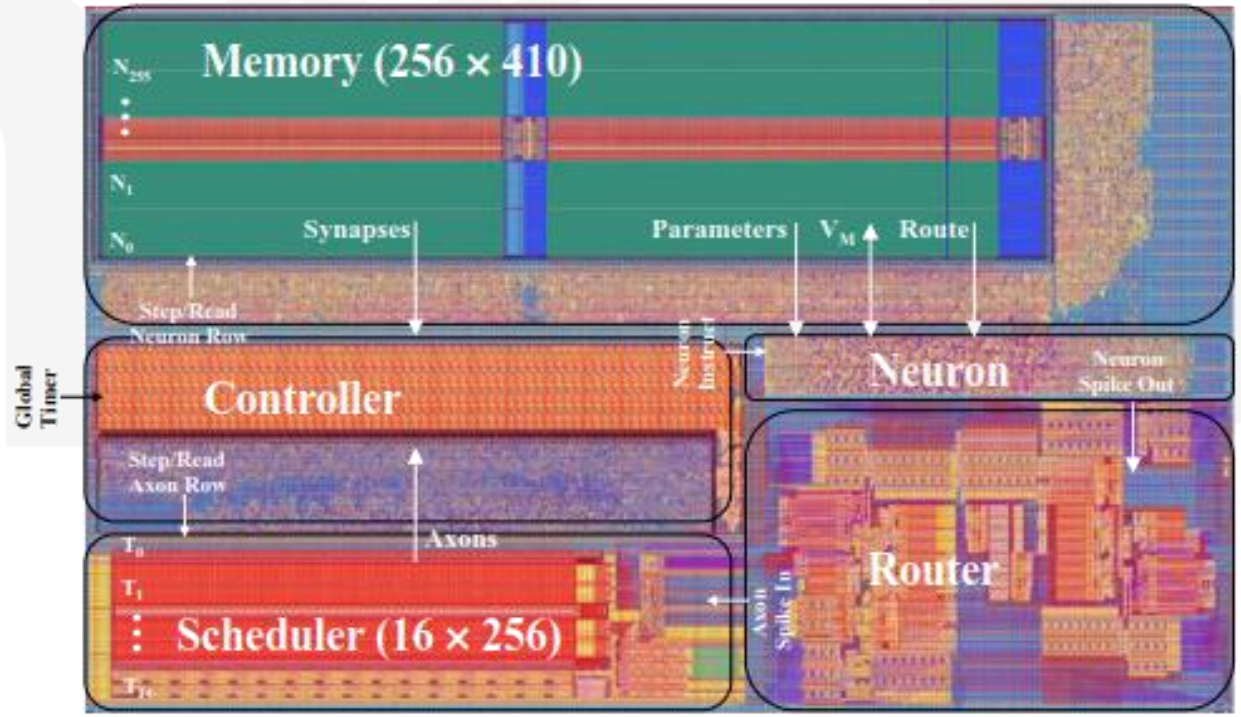


Source: Yu-Hsin Chen

- **Data stored in global buffer after decoding**
  - Further zero buffer points to on-chip zeros and data-gates the computation.

IBM TrueNorth

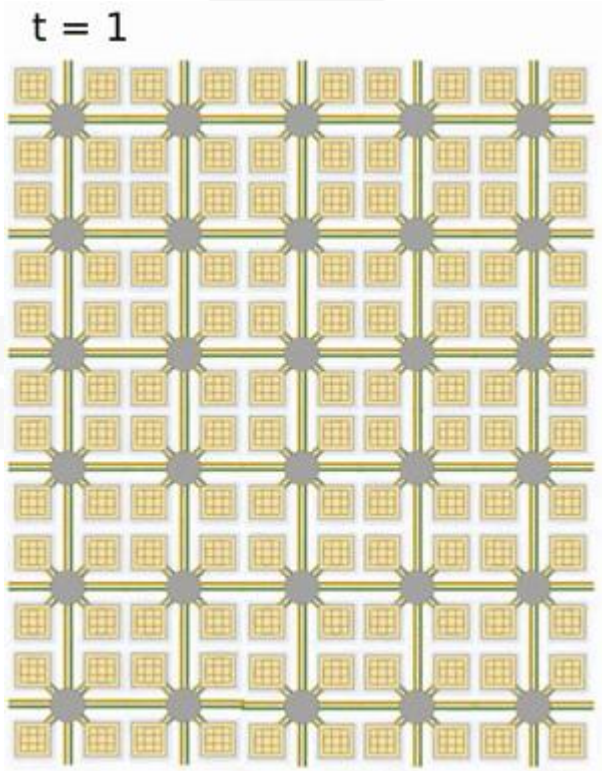
- **A 1M Neuron brain inspired machine**
  - Only 70 mW, power density 20mW/cm^2
  - 46 B synaptic operations per second, per watt
- **The idea:**
  - Non von-Neumann
  - Parallel
  - Distributed
  - Event-Driven
  - Scalable
  - Low Power



Source: IBM

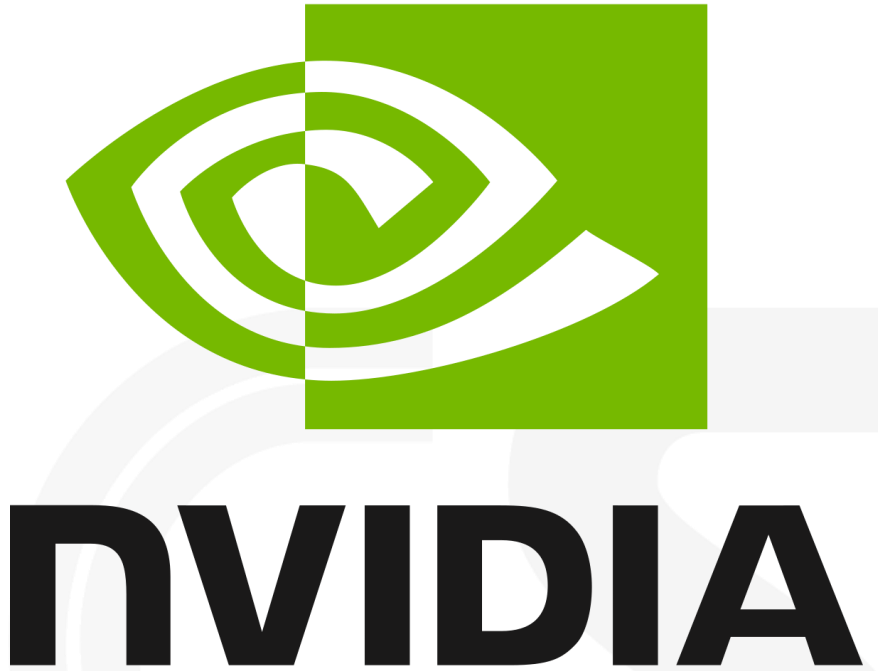
Intel Loihi Test Chip (2018)

- **"A first-of-its-kind self-learning neuromorphic chip"**
  - mimics how the brain functions by learning to operate based on various modes of feedback from the environment
- **Loihi Highlights**
  - Asynchronous neuromorphic (spiking) many core mesh
  - Each neuromorphic core includes a learning engine that can be programmed to adapt network parameters during operation, supporting supervised, unsupervised, reinforcement and other learning paradigms.
  - Intel's 14 nm process technology.
  - 128 neuromorphic cores for a total of 130,000 neurons and 130 million synapses.



Source: Intel

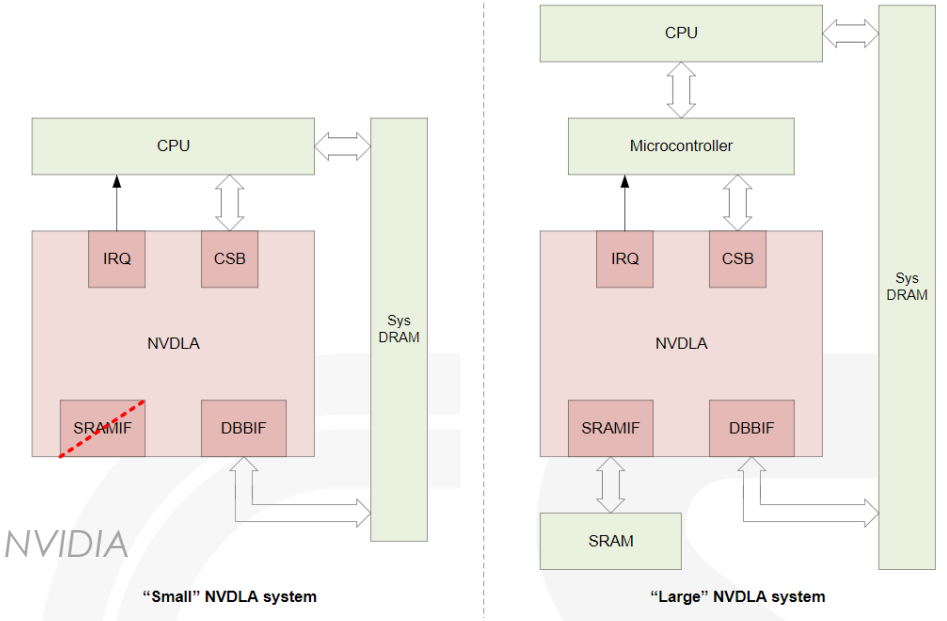
NVDLA (2017)



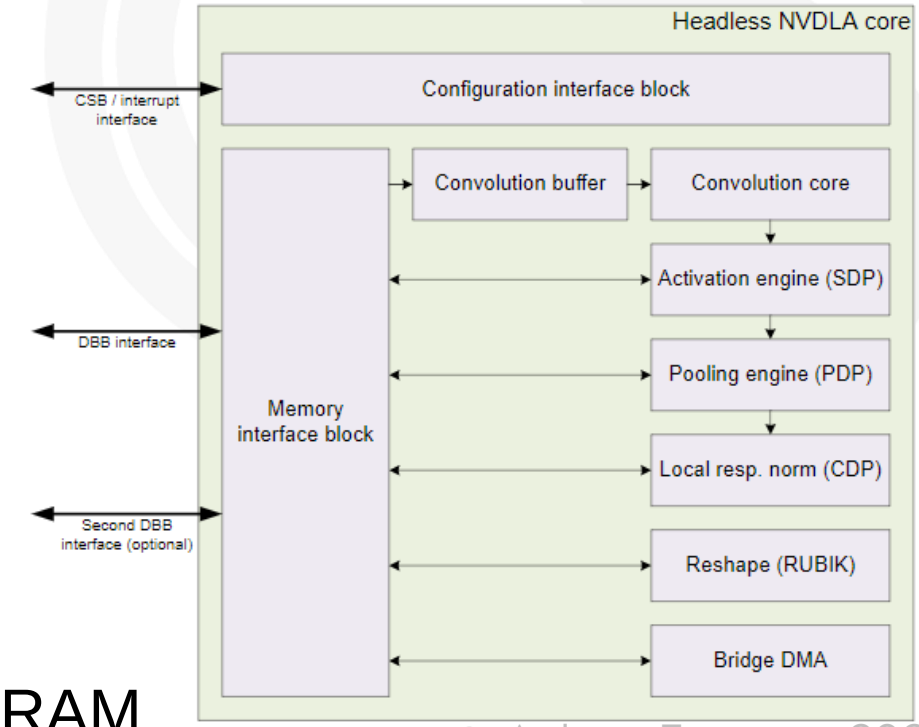
- **The NVIDIA deep learning accelerator**
  - Open source architecture and RTL release
    - Encourage Deep Learning applications
    - Invite contributions from the community
    - Targeted towards edge devices, IoT
    - Industry standard formats and parameterized
  - Complete Solution
    - Verilog and C model
    - Compiler
    - Linux drivers
    - Test benches and test suits
    - Kernel and user-mode software
    - Software development tools
- **Developed as part of Xavier – NVIDIA's SoC for autonomous driving**

NVDLA

- **Small model – for small (IoT) applications**
- **Large model – with ucontroller and SRAM**
- **Components**
  - Convolution – sparse weight compression, built in Winograd, internal conv buffer
  - Single Data Point Processor (SDP) – look up table for activations, normalization
  - Planar Data Processor (PDP) – used for max/min/avg pooling
  - Cross-channel Data Processor – calculates local response normalization (LRN)
  - Data Reshape Engine – split, slice, merge, …
  - Bridge DMA – move data between DRAM and SRAM

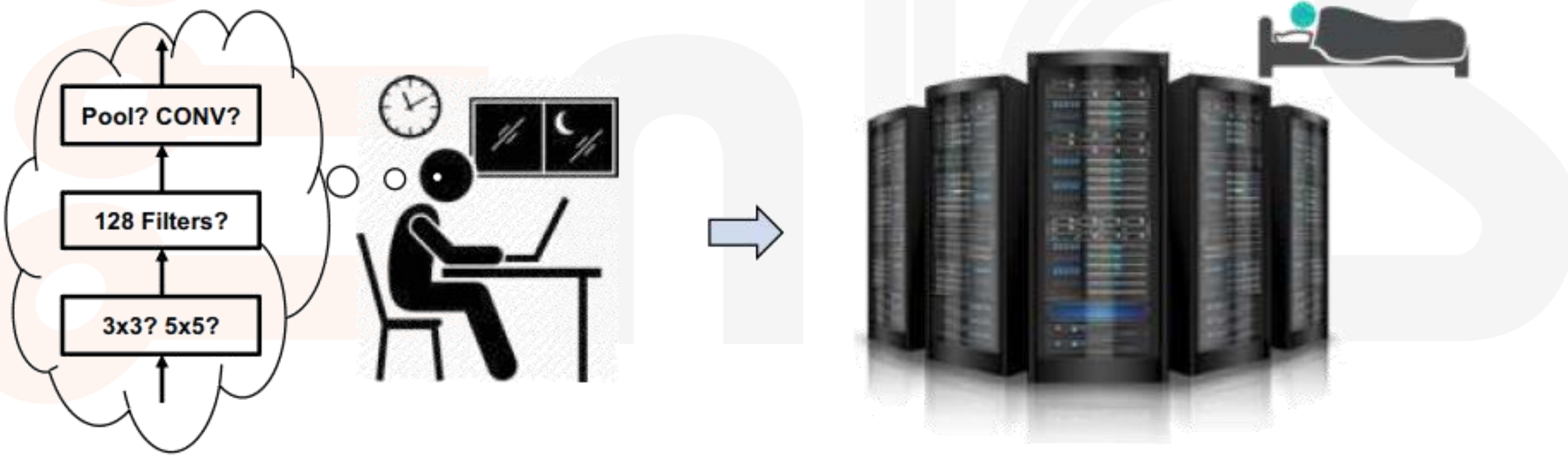


Source: NVIDIA



Network/Neural Architecture Search (NAS)

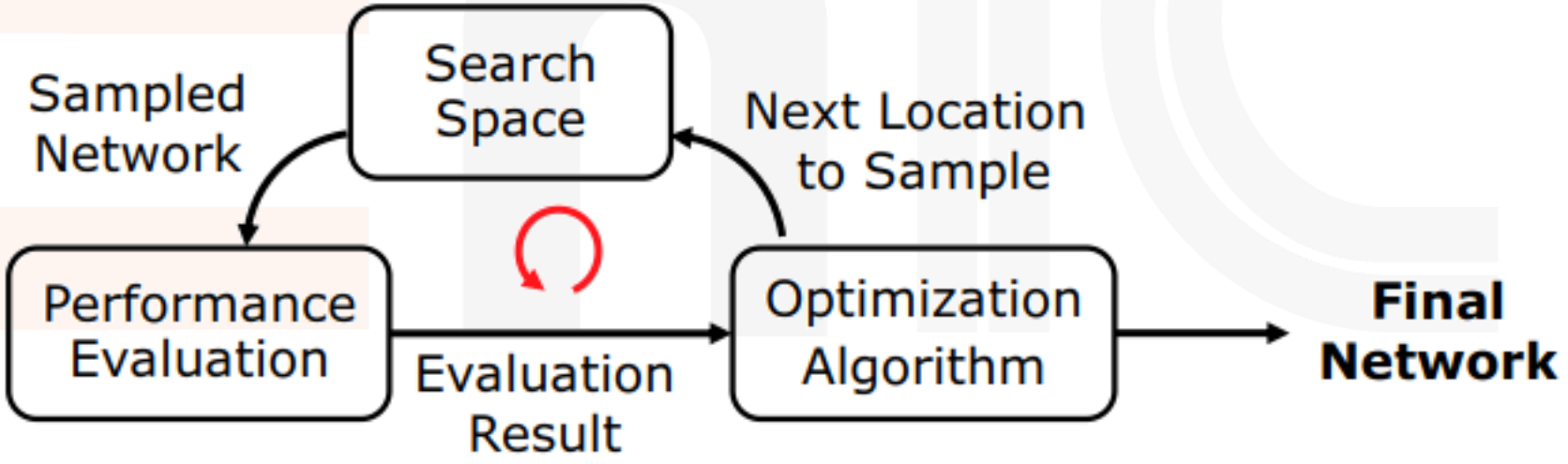
- **Rather than handcrafting the architecture, automatically search for it**



Source: Sze

Network/Neural Architecture Search (NAS)

- **Three main components:**
  - Search Space (what is the set of all samples)
  - Optimization Algorithm (where to sample)
  - Performance Evaluation (how to evaluate samples)



Key Metrics: Achievable DNN **accuracy** and required **search time**

Source: Sze

Benchmarking with ML-Perf

- **Mission**
  - To build fair and useful benchmarks for measuring training and inference performance of ML hardware, software, and services.
- **MLPerf Training**
  - The MLPerf training benchmark suite measures how fast a system can train ML models.
- **MLPerf Inference**
  - The MLPerf inference benchmark measures how fast a system can perform ML inference using a trained model.
- **Details at https://mlperf.org/**

Main References

- Song Han, various talks
- Vivienne Sze, various talks
- James W Hanlon (https://jameswhanlon.com/)
- Grigory Sapunov (Intento)
- Towards Data Science
- WikiChip
- …oh so many others (thanks Google ☺)