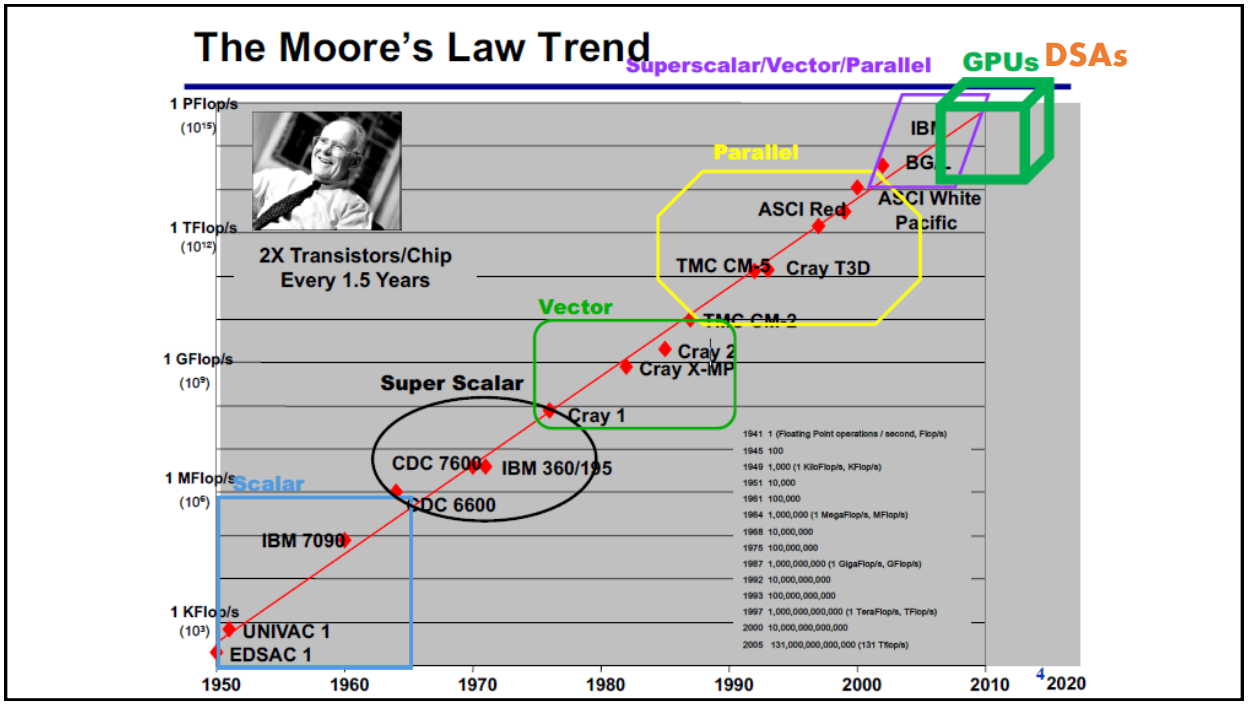
COEN-4730/EECE-5730 Computer Architecture

Lecture 12
Domain Specific Architectures (DSA)
Chapter 7

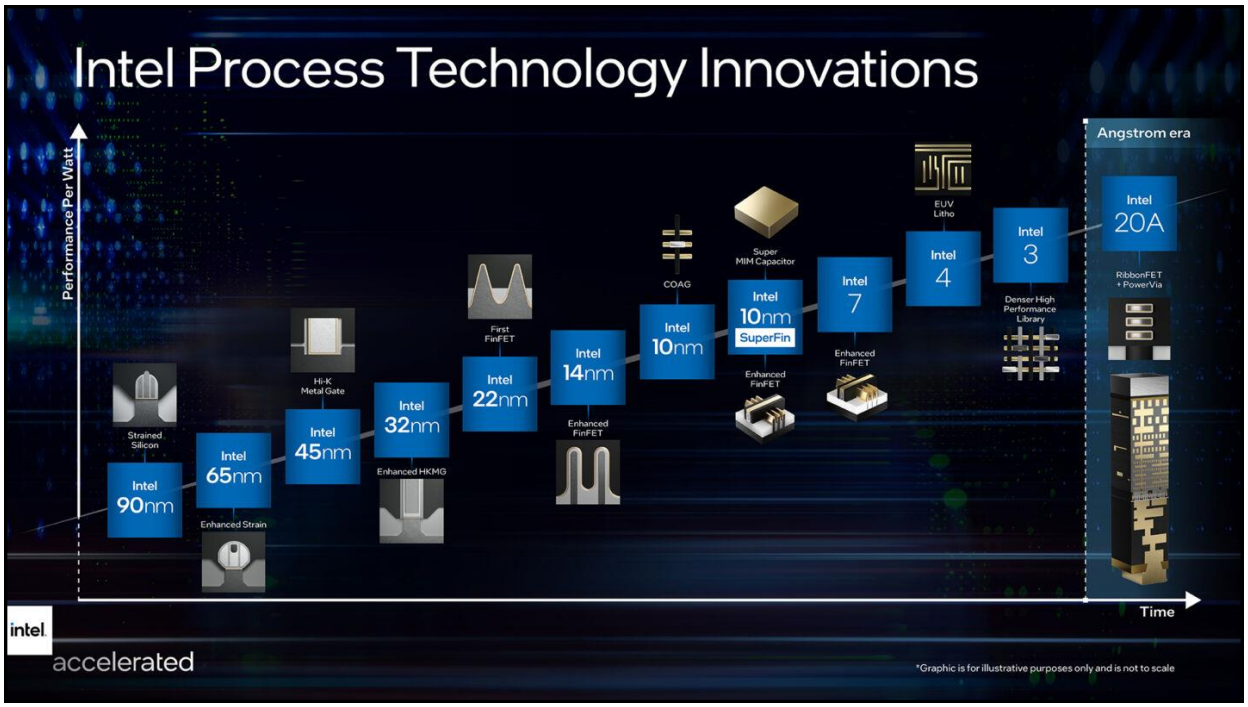
Cris Ababei
Dept. of Electrical and Computer Engineering
MARQUETTE UNIVERSITY
BE THE DIFFERENCE.

Credits: Slides adapted from presentations of Sudeep Pasricha and others: Kubiatowicz, Patterson, Mutlu, Elsevier

1



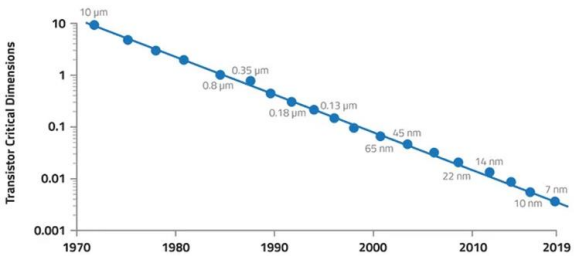
2

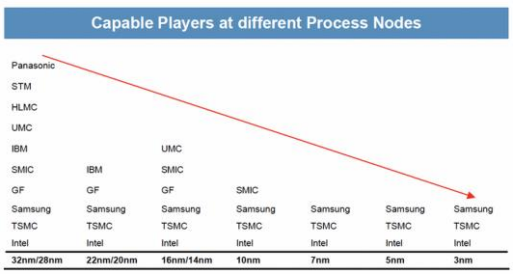


3

Introduction

- Moore's Law enabled:
  - Deep memory hierarchy
  - Wide SIMD units
  - Deep pipelines
  - Branch prediction
  - Out-of-order execution
  - Speculative prefetching
  - Multithreading
  - Multiprocessing
- Objective:
  - Extract performance from software that is oblivious to architecture





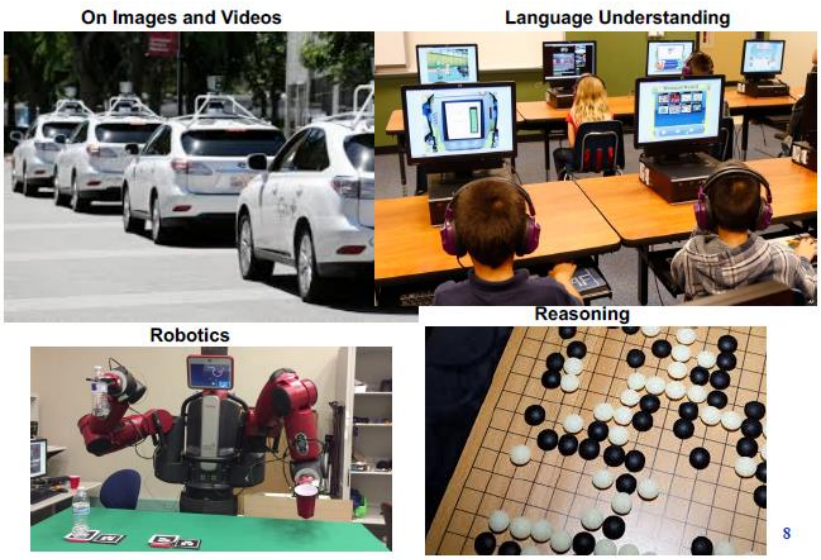
4

Introduction

- Need factor of 100 improvements in number of operations per instruction
  - Requires domain specific architectures (DSAs)
  - For ASICs, NRE cannot be amortized over large volumes
  - FPGAs are less efficient than ASICs
- Video:
  - https://www.acm.org/hennessy-patterson-turing-lecture
- Paper:
  - https://cacm.acm.org/magazines/2019/2/234352-a-new-golden-age-for-computer-architecture/fulltext
- Slides:
  - https://iscaconf.org/isca2018/docs/HennessyPattersonTuringLectureISCA4June2018.pdf

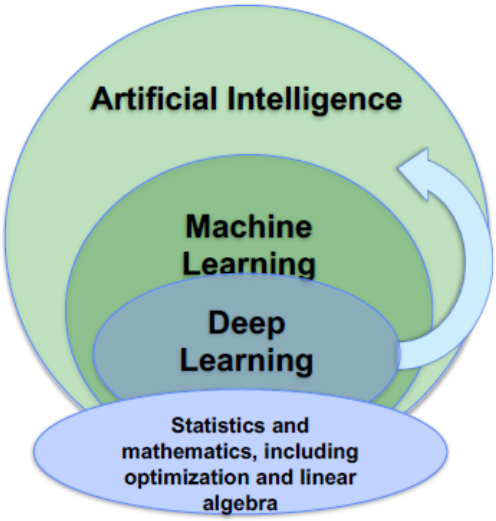
5

AI/ML Domain



6

Artificial Intelligence, Machine Learning and Deep Learning



Big Data Processing

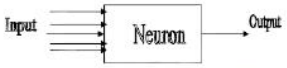
Sophisticated Algorithms

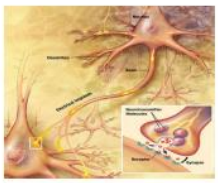
High Performance Machines

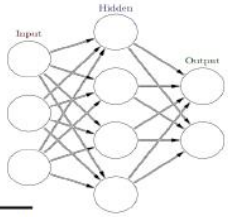
7

Example Domain: 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

Figure 7.5 Six DNN applications that represent 95% of DNN workloads for inference at Google in 2016, which we use in Section 7.9. The columns are the DNN name, the number of layers in the DNN, the number of weights, and operations per weight (operational intensity). Figure 7.41 on page 595 goes into more detail on these DNNs.

8

Example Domain: Deep Neural Networks

- **Most practioners will choose an existing design**
  - **Topology and Data type**
- **Training (learning):**
  - **Calculate weights using backpropagation algorithm**
  - **Supervised learning: stocastic gradient 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

Figure 7.6 Training set sizes and training time for several DNNs (Iandola, 2016).

- **Inferrence: use neural network for classification**

9

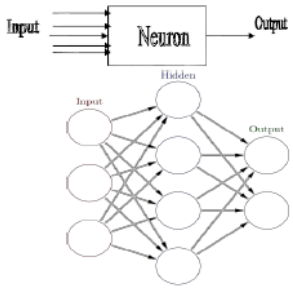
Example Domain: Deep Neural Networks

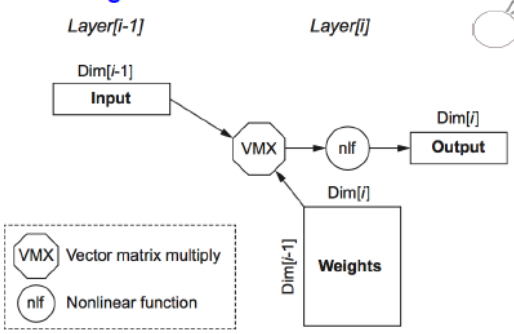
- Three of the most important DNNs:
  1. Multi-Layer Perceptron
  2. Convolutional Neural Network
  3. Recurrent Neural Network

10

1) Multi-Layer Perceptron (MLP)

- **Parameters:**
  - Dim[i]: number of neurons
  - Dim[i-1]: dimension of input vector
  - Number of weights: Dim[i-1] x Dim[i]
  - Operations: 2 x Dim[i-1] x Dim[i]
  - Operations/weight: 2

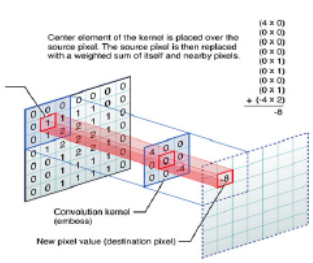


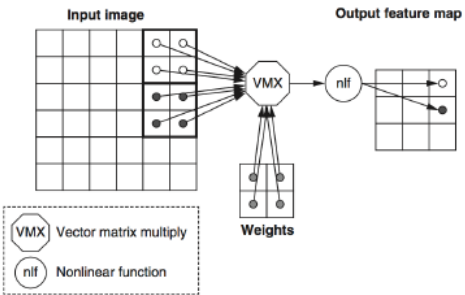


11

2) Convolutional Neural Network (CNN)

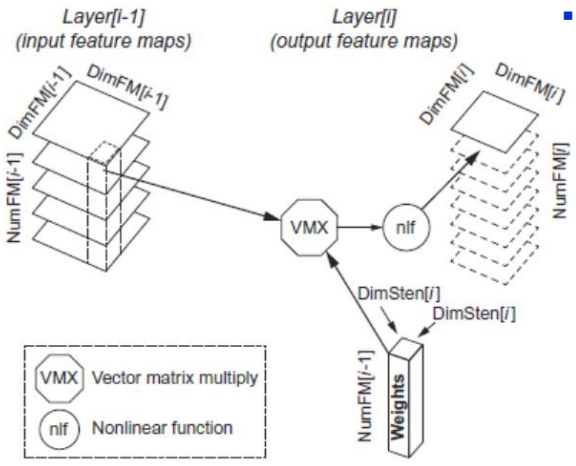
- **Computer vision**
- **Each layer raises the level of abstraction**
  - First layer recognizes horizontal and vertical lines
  - Second layer recognizes corners
  - Third layer recognizes shapes
  - Fourth layer recognizes features, such as ears of a dog
  - Higher layers recognizes different breeds of dogs





12

2) Convolutional Neural Network (CNN)



- Parameters:
  - DimFM[i-1]: Dimension of the (square) input Feature Map
  - DimFM[i]: Dimension of the (square) output Feature Map
  - DimSten[i]: Dimension of the (square) stencil
  - NumFM[i-1]: Number of input Feature Maps
  - NumFM[i]: Number of output Feature Maps
  - Number of neurons: NumFM[i] x DimFM[i]^2
  - Number of weights per output Feature Map: NumFM[i-1] x DimSten[i]^2
  - Total number of weights per layer: NumFM[i] x Number of weights per output Feature Map
  - Number of operations per output Feature Map: 2 x DimFM[i]^2 x Number of weights per output Feature Map
  - Total number of operations per layer: NumFM[i] x Number of operations per output Feature Map = 2 x DimFM[i]^2 x NumFM[i] x Number of weights per output Feature Map = 2 x DimFM[i]^2 x Total number of weights per layer
  - Operations/Weight: 2 x DimFM[i]^2

13

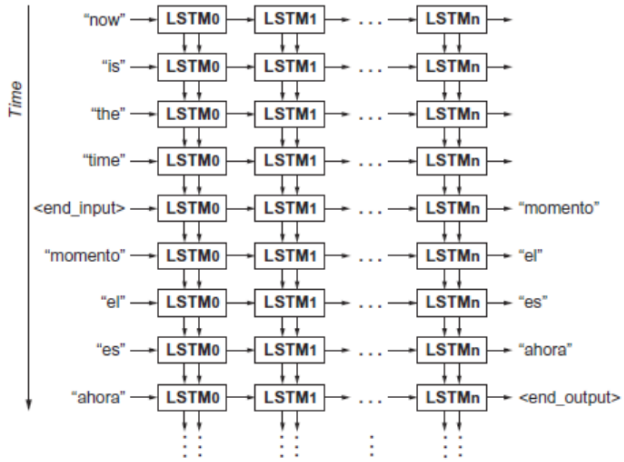
2) Convolutional Neural Network (CNN)

- Batches:
  - Reuse weights once fetched from memory across multiple inputs
  - Increases operational intensity
- Quantization
  - Use 8- or 16-bit fixed point
- Summary:
  - Need the following kernels:
    - Matrix-vector multiply
    - Matrix-matrix multiply
    - Stencil
    - ReLU
    - Sigmoid
    - Hyperbolic tangent

14

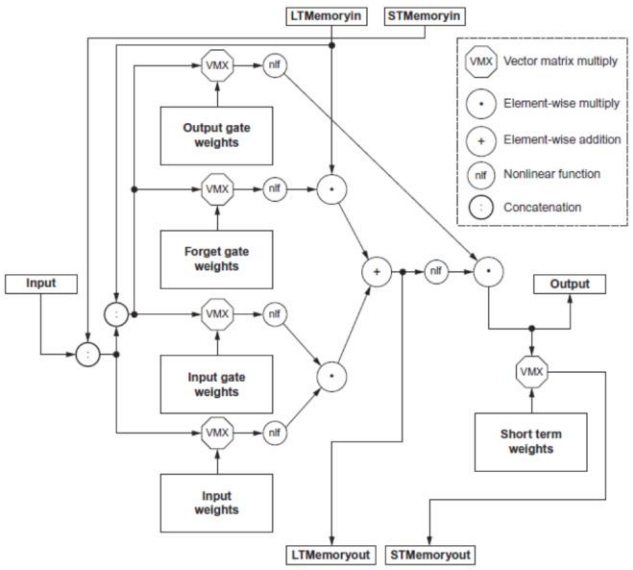
3) Recurrent Neural Network (RNN)

- Speech recognition and language translation
- Long short-term memory (LSTM) network



15

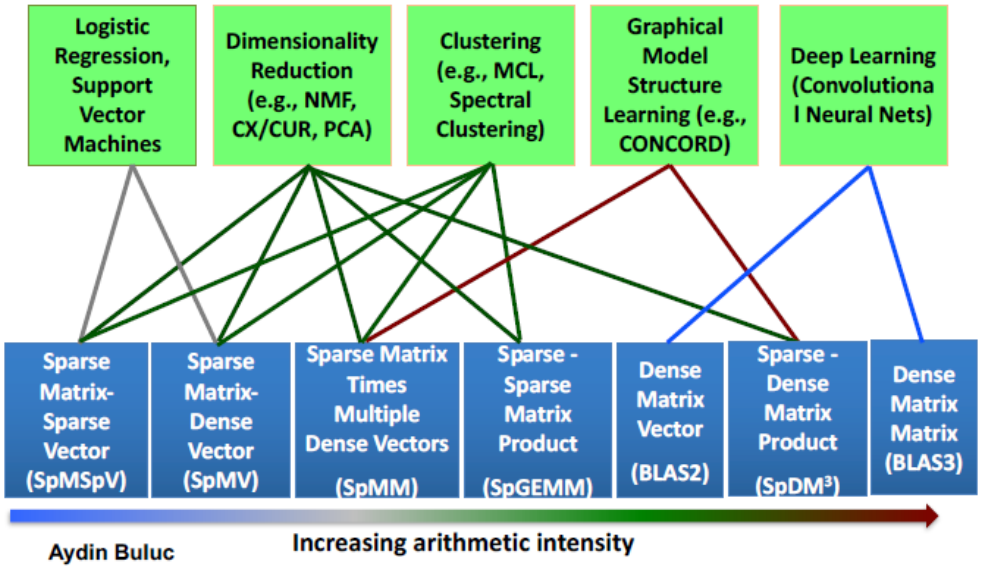
3) Recurrent Neural Network (RNN)



- Parameters:
  - Number of weights per cell: 3 x (3 x Dim x Dim)+(2 x Dim x Dim) + (1 x Dim x Dim) = 12 x Dim^2
  - Number of operations for the 5 vector-matrix multiplies per cell: 2 x Number of weights per cell = 24 x Dim^2
  - Number of operations for the 3 element-wise multiplies and 1 addition (vectors are all the size of the output): 4 x Dim
  - Total number of operations per cell (5 vector-matrix multiplies and the 4 element-wise operations): 24 x Dim^2 + 4 x Dim
  - Operations/Weight: ~2

16

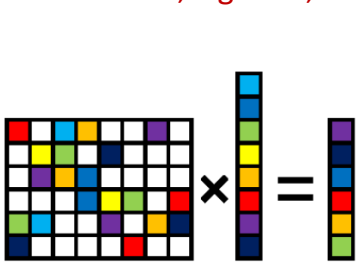
Machine Learning Mapping to Linear Algebra

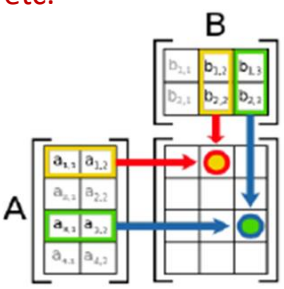


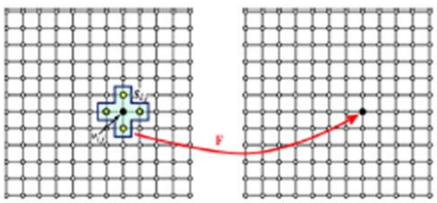
17

Summary

- Need high-efficient (performance and power) implementation of dense matrix operations
  - Matrix-vector, matrix-matrix multiplication, and stencil
- Other nonlinear functions
  - ReLU, Sigmoid, tanh, etc.







18

Five Guidelines for Domain Specific Architectures (DSAs)

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

19

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

20

Examples of DSAs

1. Tensor Processing Unit
2. Microsoft Catapult
3. Intel Crest
4. Pixel Visual Core

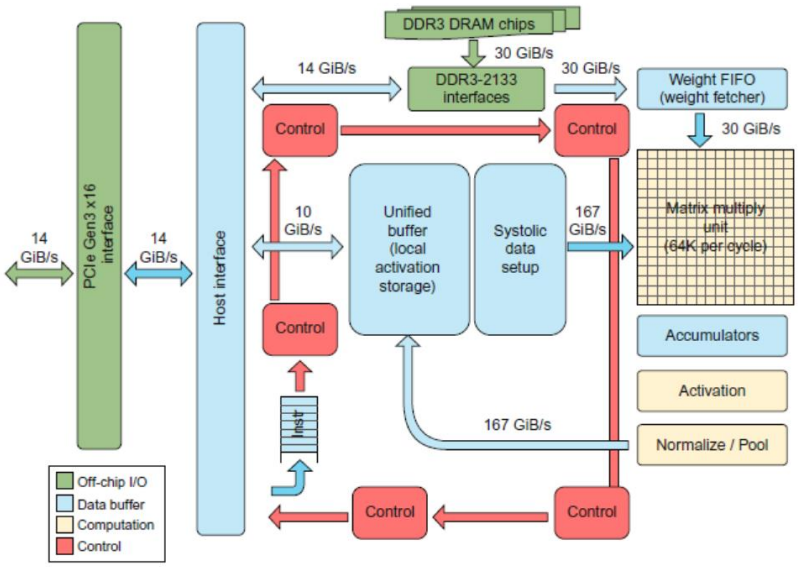
21

1) Tensor Processing Unit (TPU)

- Google's DNN ASIC (Application-specific Integrated Circuit)
  - Designed for inference phase
  - TensorFlow programming interface
  - First TPU in 2015, Second 2017, Third in May 2018
    - » Design-verification-build-deployment in 15 months for the first one
- Heart:
  - 256 x 256 8-bit matrix multiply-add unit
  - Large software-managed scratchpad
- Coprocessor on the PCIe bus

22

Tensor Processing Unit



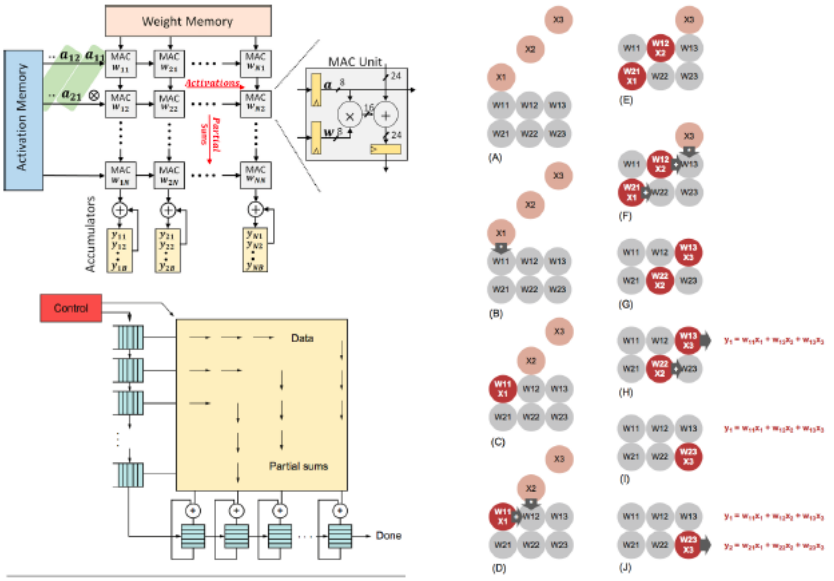
23

Tensor Processing Unit

- Read_Host_Memory
  - Reads memory 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 matrix-matrix multiply, a vector-matrix multiply, an element-wise matrix multiply, an element-wise vector multiply, 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
- Write_Host_Memory
  - Writes data from unified buffer into host memory

24

TPU Microarchitecture – Systolic Array



25

TPU Implementation

- TPU chip fabricated using the 28-nm process, 700 MHz clock.
  - Less than half size of an Intel Haswell CPU, which is 662 mm².

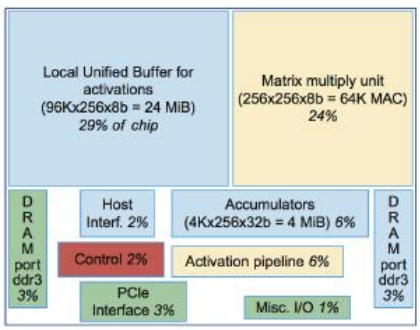


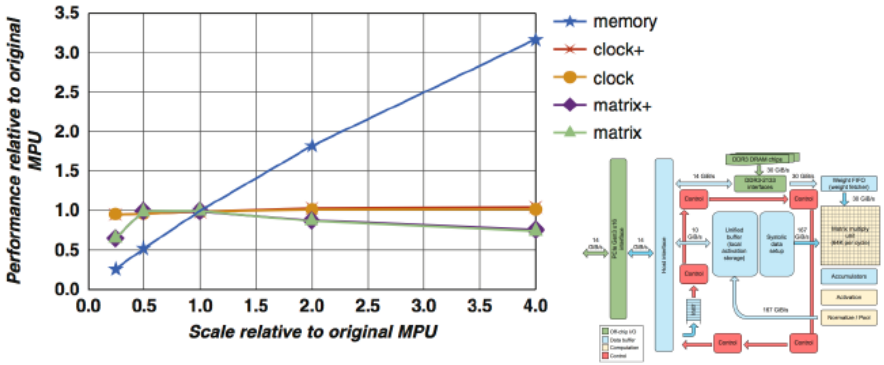
Figure 7.16 TPU printed circuit board. It can be inserted into the slot for an SATA disk in a server, but the card uses the PCIe bus.

Figure 7.15 Floor plan of TPU die. The shading follows Figure 7.14. The light data buffers are 37%, the light computation units are 30%, the medium I/O is 10%, and the dark control is just 2% of the die. Control is much larger (and much more difficult to design) in a CPU or GPU. The unused white space is a consequence of the emphasis on time to tape-out for the TPU.

26

Improving the TPU

- First, increasing memory bandwidth (memory) has the biggest impact:
  - improves 3 on average when memory bandwidth increases 4 , because it reduces the time waiting for weight memory.
- Second, clock rate has little benefit on average with or without more accumulators.
- Third, the average performance slightly degrades when the matrix unit expands from 256x256 to 512x512 for all applications, whether or not having more accumulators.
  - The issue is analogous to internal fragmentation of large pages, only worse because it's in two dimensions.



27

TPU and the 5 Guidelines

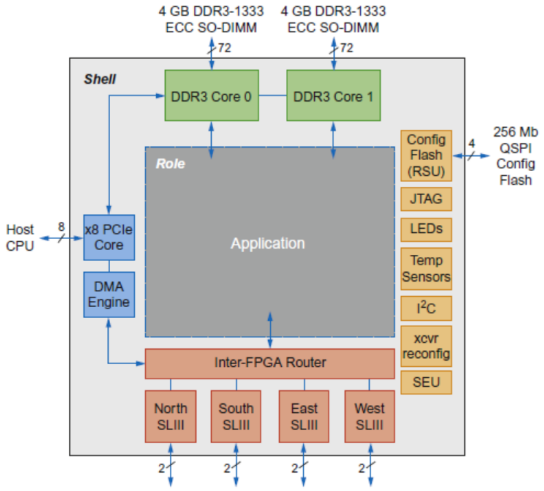
- Use dedicated memories
  - 24 MiB dedicated buffer, 4 MiB accumulator buffers
- Invest resources in arithmetic units and dedicated memories
  - 60% of the memory and 250X the arithmetic units of a server-class CPU
- Use the easiest form of parallelism that matches the domain
  - Exploits 2D SIMD parallelism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

28

28

2) Microsoft Catapult

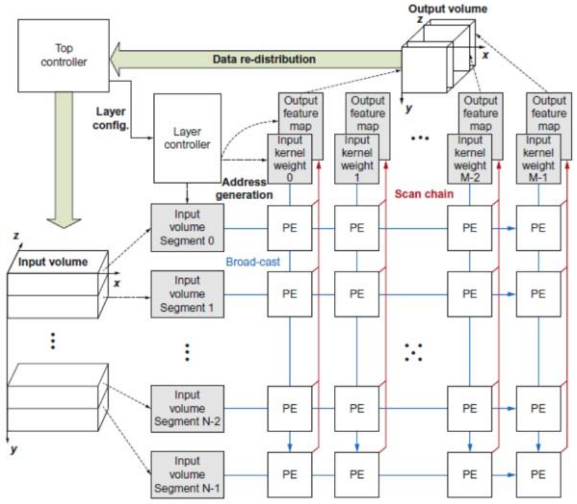
- Needed to be general purpose and power efficient
  - Uses FPGA PCIe board with dedicated 20 Gbps network in 6 x 8 torus
  - Each of the 48 servers in half the rack has a Catapult board
  - Limited to 25 watts
  - 32 MiB Flash memory
  - Two banks of DDR3-1600 (11 GB/s) and 8 GiB DRAM
  - FPGA (unconfigured) has 3962 18-bit ALUs and 5 MiB of on-chip memory
  - Programmed in Verilog RTL
  - Shell is 23% of the FPGA



29

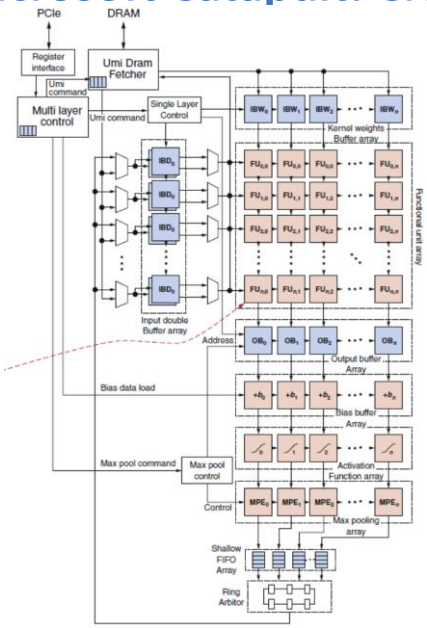
Microsoft Catapult: CNN

- CNN accelerator, mapped across multiple FPGAs



30

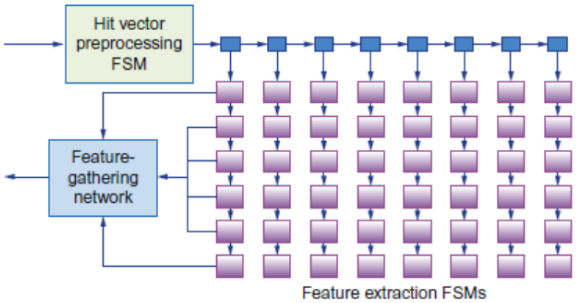
Microsoft Catapult: CNN



31

Microsoft Catapult: Search Ranking

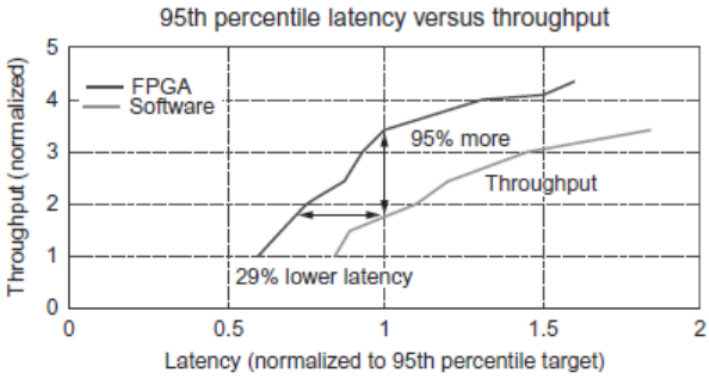
- Feature extraction (1 FPGA)
  - Extracts 4500 features for every document-query pair, e.g. frequency in which the query appears in the page
  - Systolic array of FSMs
- Free-form expressions (2 FPGAs)
  - Calculates feature combinations
- Machine-learned Scoring (1 FPGA for compression, 3 FPGAs calculate score)
  - Uses results of previous two stages to calculate floating-point score
- One FPGA allocated as a hot-spare



32

Microsoft Catapult: Search Ranking

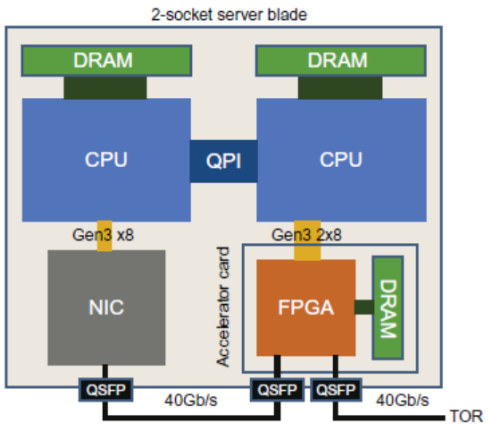
- Free-form expression evaluation
  - 60 core processor
  - Pipelined cores
  - Each core supports four threads that can hide each other's latency
  - Threads are statically prioritized according to thread latency



33

Microsoft Catapult: Search Ranking

- Version 2 of Catapult
  - Placed the FPGA between the CPU and NIC
  - Increased network from 10 Gb/s to 40 Gb/s
  - Also performs network acceleration
  - Shell now consumes 44% of the FPGA
  - Now FPGA performs only feature extraction



34

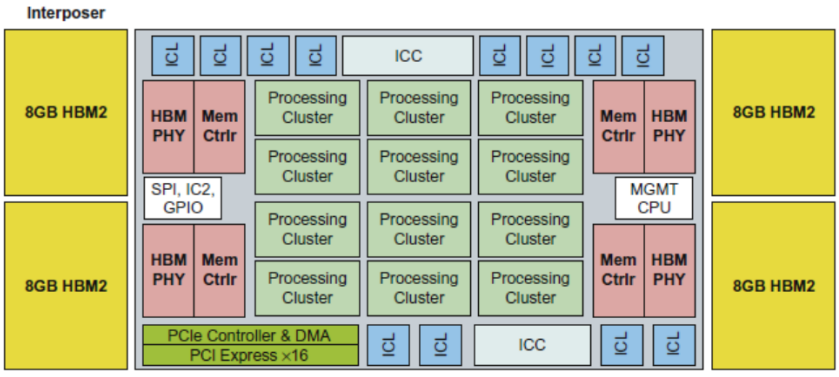
Catapult and the 5 Guidelines

- Use dedicated memories
  - 5 MiB dedicated memory
- Invest resources in arithmetic units and dedicated memories
  - 3926 ALUs
- Use the easiest form of parallelism that matches the domain
  - 2D SIMD for CNN, MISD parallelism for search scoring
- Reduce the data size and type needed for the domain
  - Uses mixture of 8-bit integers and 64-bit floating-point
- Use a domain-specific programming language
  - Uses Verilog RTL; Microsoft did not follow this guideline

35

3) Intel Crest

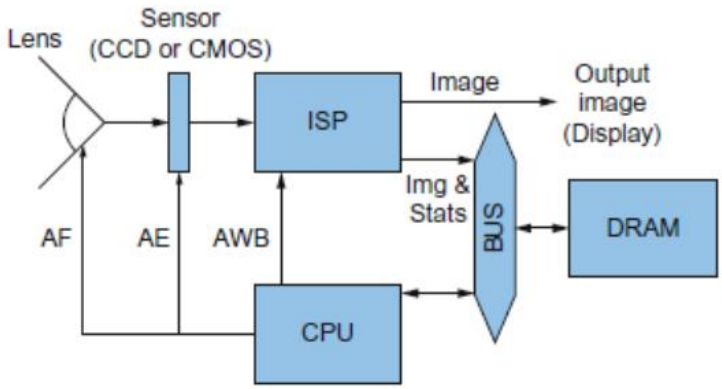
- **DNN training**
- **16-bit fixed point**
- **Operates on blocks of 32x32 matrices**
- **SRAM + HBM2**



36

4) Pixel Visual Core

- **Pixel Visual Core**
  - **Image Processing Unit**
  - **Performs stencil operations**
  - **Decended from Image Signal processor**



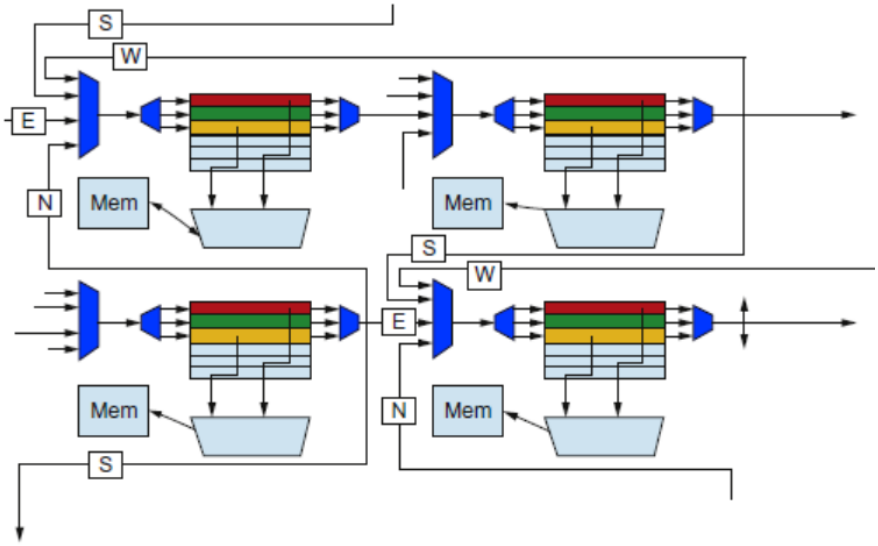
37

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

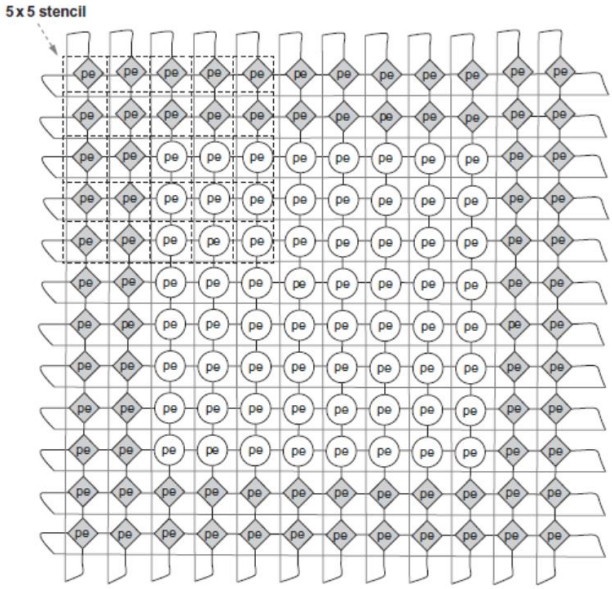
38

Pixel Visual Core



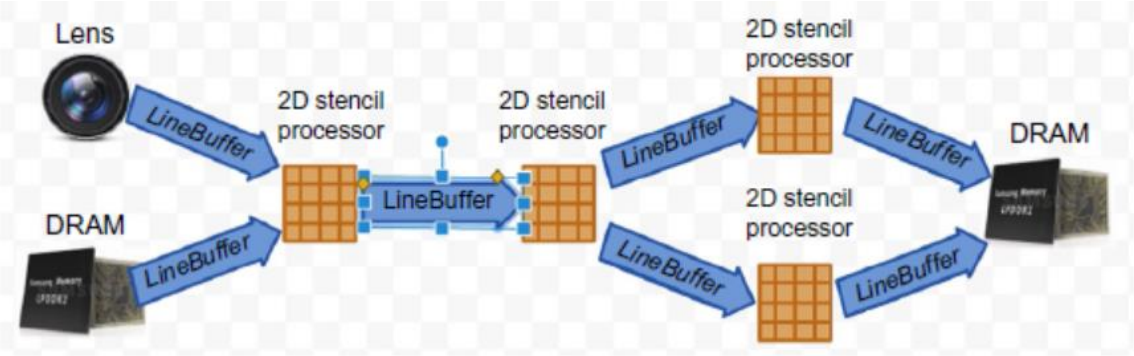
39

Pixel Visual Core



40

Pixel Visual Core



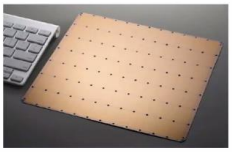
41

Visual Core and the 5 Guidelines

- Use dedicated memories
  - 128 + 64 MiB dedicated memory per core
- Invest resources in arithmetic units and dedicated memories
  - 16x16 2D array of processing elements per core and 2D shifting network per core
- Use the easiest form of parallelism that matches the domain
  - 2D SIMD and VLIW
- Reduce the data size and type needed for the domain
  - Uses mixture of 8-bit and 16-bit integers
- Use a domain-specific programming language
  - Halide for image processing and TensorFlow for CNNs

42

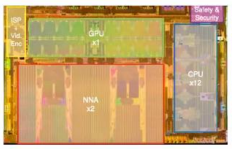
More Emerging ML Accelerators



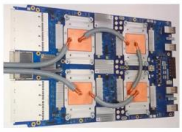
Cerebras: Wafer-Scale Deep Learning
- Largest Chip Ever Built!
- 46,225 mm2 silicon
- 1.2 trillion transistors
- 400,000 optimized AI cores
- 18 GB of on-chip memory
- TSMC 16nm process

NVIDIA DGX-2
- 2 PetaFlops System
- 16X Tesla V100
- Max Power: 10kW
- $399,000

NVIDIA Jetson Nano
- $99 computer for edge devices
- 472 GFLOPS
- Quad-core 64-bit ARM CPU
- 128-core GPU
- 5W/10W



Tesla: Self-Driving Computer
- 2 independent instances
- 2Ghz+ Design
- 32 MB SRAM / instance
- 96*96 MACs

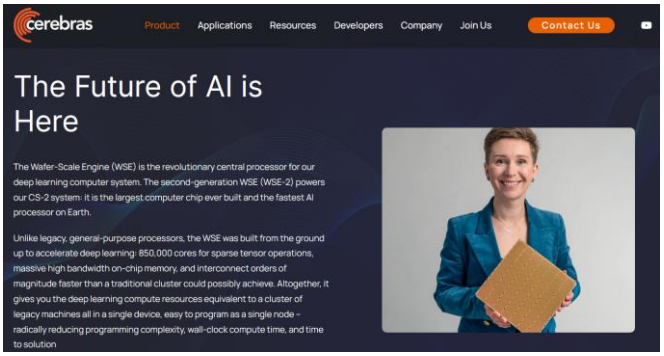


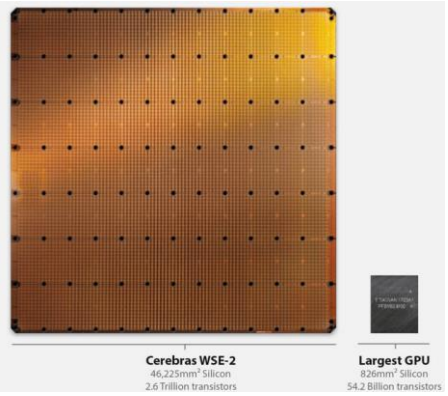
Systolic-array-based architecture
- V1: Inference only
- V2: Training with bfloat
- V3: 2x powerful than v2

Edge TPU
- Coral Dev Board
- 4 TOPS
- 2 TOPS/Watt
- Supports TensorFlow Lite

43

Cerebras





https://www.cerebras.net/product-chip/

44

NVIDIA DGX

Clusters and Systems

DGX GH200

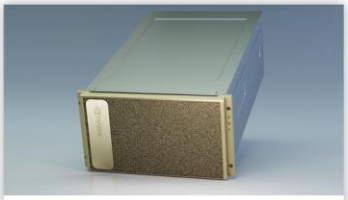
The world's first AI supercomputer designed to handle terabyte-class models.

Giant Memory for Giant Models ›

DGX H100

AI supercomputer optimized for large generative AI and other transformer-based workloads.

Proven Choice for Enterprise AI ›



DGX A100

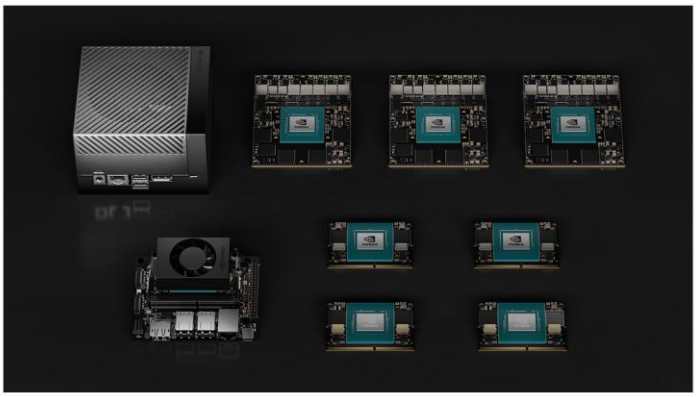
AI supercomputer delivering world-class performance for mainstream AI workloads.

Universal System for AI Infrastructure ›

https://www.nvidia.com/en-us/data-center/dgx-platform/

45

NVIDIA Jetson Modules

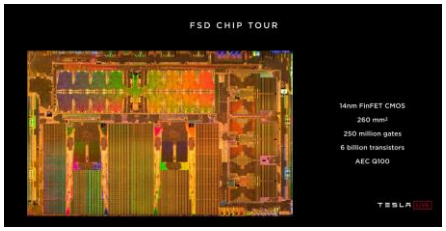


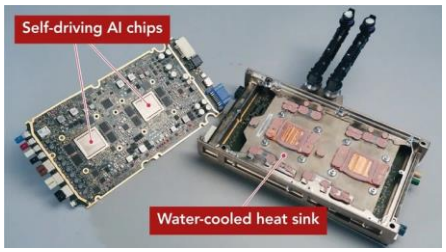
The Jetson family of modules all use the same NVIDIA CUDA-X™ software, and support cloud-native technologies like containerization and orchestration to build, deploy, and manage AI at the edge.

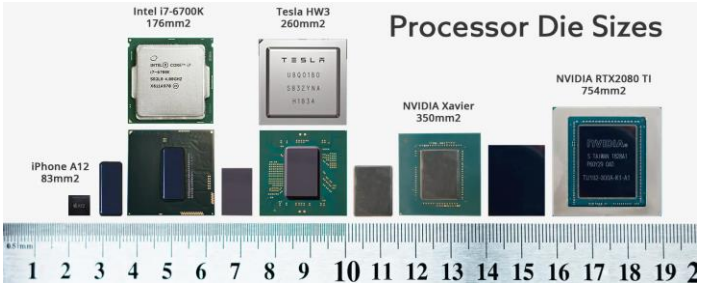
https://developer.nvidia.com/embedded/jetson-modules

46

TESLA: Self Driving Computer







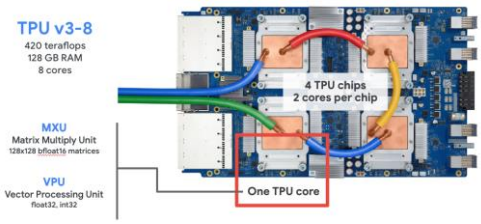
https://www.tesla.com/AI

47

Tensor Processing Unit (TPU) – V1, …, V5, …

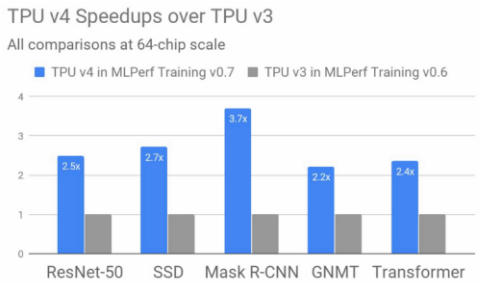
Tensor Processing Unit products[13][14][15]

	TPUv1	TPUv2	TPUv3	TPUv4[14][16]	TPUv5[17]	Edge v1
Date introduced	2016	2017	2018	2021	2023	2018
Process node	28 nm	16 nm	16 nm	7 nm	Unstated	
Die size (mm²)	331	< 625	< 700	< 400	Unstated	
On-chip memory (MiB)	28	32	32	32	48	
Clock speed (MHz)	700	700	940	1050	Unstated	
Memory	8 GiB DDR3	16 GiB HBM	32 GiB HBM	32 GiB HBM	16 GB HBM	
Memory bandwidth	34 GB/s	600 GB/s	900 GB/s	1200 GB/s	819 GB/s	
TDP (W)	75	280	220	170	Not Listed	2
TOPS (Tera Operations Per Second)	92	45	123	275	393	4
TOPS/W	0.31	0.16	0.56	1.62	Not Listed	2



Chip feature	Cloud TPU v3	Cloud TPU v4
Peak compute per chip	123 teraflops (bf16)	275 teraflops (bf16 or int8)
HBM2 capacity and bandwidth	32 GiB, 900 GB/s	32 GiB, 1200 GB/s
Measured min/mean/max power	123/220/262 W	90/170/192 W
TPU pod size	1024 chips	4096 chips
Interconnect topology	2D torus	3D torus
Peak compute per pod	126 petaflops (bf16)	1.1 exaflops (bf16 or int8)
All-reduce bandwidth per pod	340 TB/s	1.1 PB/s
Bisection bandwidth per pod	6.4 TB/s	24 TB/s

Cloud TPU v4 pods performance table (source: Google)



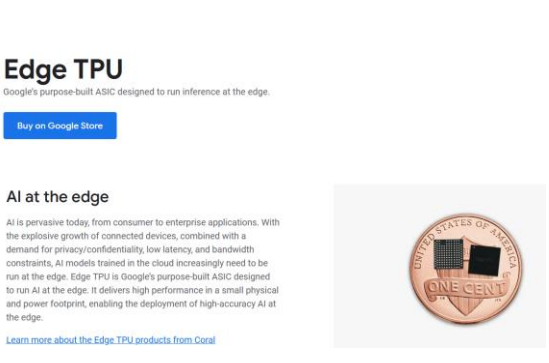
https://en.wikipedia.org/wiki/Tensor_Processing_Unit

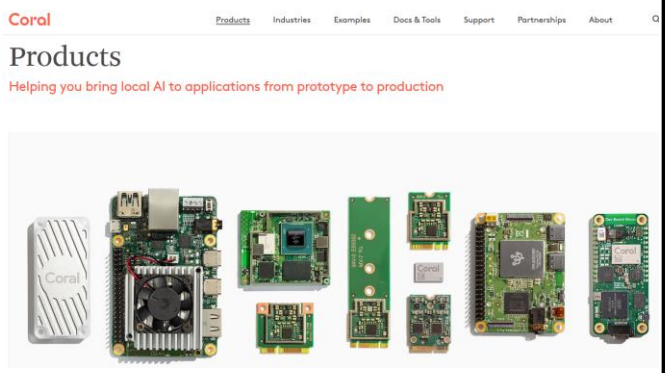
https://cloud.google.com/tpu

https://cloud.google.com/blog/topics/systems/tpu-v4-enables-performance-energy-and-co2e-efficiency-gains

48

Edge TPU

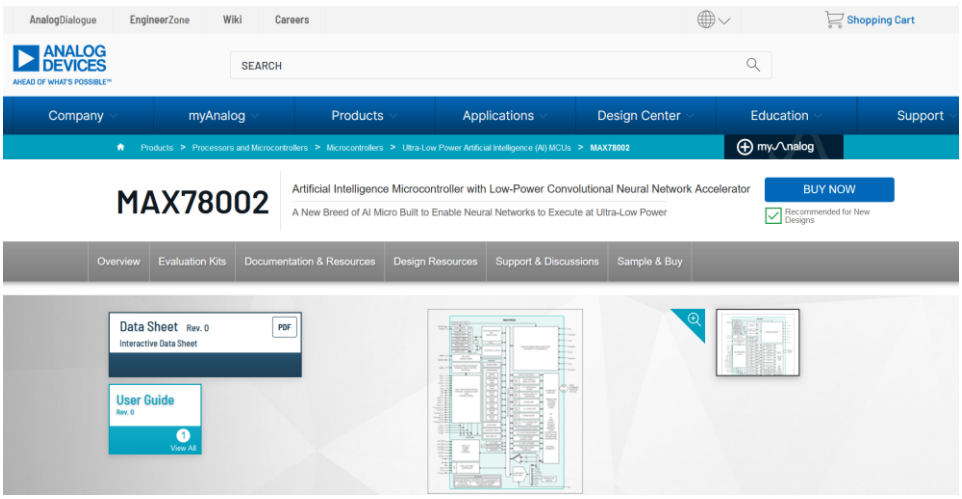




- https://cloud.google.com/edge-tpu
- https://coral.ai/products/

49

MAX78002



- https://www.analog.com/en/products/max78002.html

50

Conclusion: A New Golden Age

- End of Dennard Scaling and Moore's Law
  - architecture innovation to improve performance/cost/energy
- Security ⇒architecture innovation too
- Domain Specific Languages ⇒Domain Specific Architectures
- Free, open architectures and open-source implementations
  - everyone can innovate and contribute
- Cloud FPGAs ⇒all can design and deploy custom "HW"
- Agile HW development ⇒all can afford to make (small) chips
- Like 1980s, great time for architects in academia & in industry!


turing lecture

Innovations like domain-specific hardware, enhanced security, open instruction sets, and agile chip development will lead the way.

BY JOHN L. HENNESSY AND DAVID A. PATTERSON

A New Golden Age for Computer Architecture

51

Resources, Credits

- A New Golden Age for Computer Architecture Slides
  - https://iscaconf.org/isca2018/docs/HennessyPattersonTuringLectureISCA4June2018.pdf
- Yonghong Yan's slides
  - https://passlab.github.io/CSCE513/notes/lecture26_DSA_DomainSpecificArchitectures.pdf
- Kathy Yelic's Talks
  - https://people.eecs.berkeley.edu/~yelick/talks.html
- Sudeep Pasricha
  - Keynote talk at NoCArc 2021
- https://www.isc-hpc.com/
- https://io.google/2023

52

52