Lecture 26: Domain Specific Architectures
Chapter 07, CAQA 6th Edition

CSCE 513 Computer Architecture

Department of Computer Science and Engineering

Yonghong Yan

yanyh@cse.sc.edu

https://passlab.github.io/CSCE513

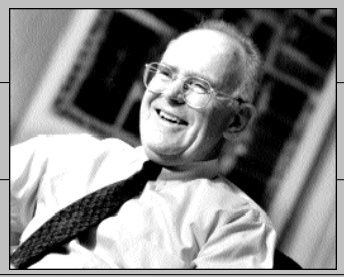
Copyright and Acknowledgements

- Machine Learning for Science" in 2018 and A Superfacility Model for Science" in 2017 By Kathy Yelic
  - https://people.eecs.berkeley.edu/~yelick/talks.html

CSE 564 Class Contents

- Introduction to Computer Architecture (CA)
- Quantitative Analysis, Trend and Performance of CA
  - Chapter 1
- Instruction Set Principles and Examples
  - Appendix A
- Pipelining and Implementation, RISC-V ISA and Implementation
  - Appendix C, RISC-V (riscv.org) and UCB RISC-V impl
- Memory System (Technology, Cache Organization and Optimization, Virtual Memory)
  - Appendix B and Chapter 2
  - Midterm covered till Memory Tech and Cache Organization
- Instruction Level Parallelism (Dynamic Scheduling, Branch Prediction, Hardware Speculation, Superscalar, VLIW and SMT)
  - Chapter 3
- Data Level Parallelism (Vector, SIMD, and GPU)
  - Chapter 4
- Thread Level Parallelism
  - Chapter 5
- Domain-Specific Architecture
  - Chapter 7

The Moore's Law Trend



2X Transistors/Chip
Every 1.5 Years

Superscalar/Vector/Parallel

GPUs

Parallel

Vector

Super Scalar

Scalar

1 PFlop/s (10^{15})

1 TFlop/s (10^{12})

1 GFlop/s (10^{9})

1 MFlop/s (10^{6})

1 KFlop/s (10^{3})

IBM

BG/L

ASCI White Pacific

ASCI Red

TMC CM-5

Cray T3D

TMC CM-2

Cray 2

Cray X-MP

Cray 1

CDC 7600

IBM 360/195

CDC 6600

IBM 7090

UNIVAC 1

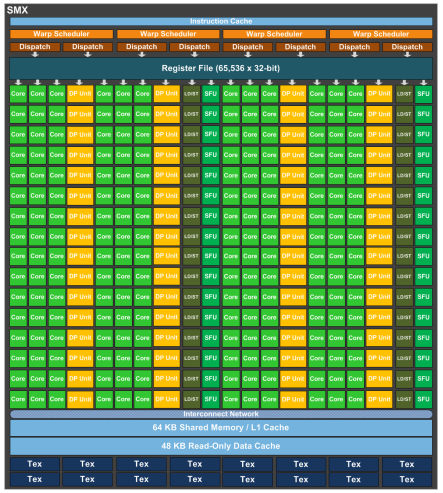
EDSAC 1

1941	1 (Floating Point operations / second, Flop/s)
1945	100
1949	1,000 (1 KiloFlop/s, KFlop/s)
1951	10,000
1961	100,000
1964	1,000,000 (1 MegaFlop/s, MFlop/s)
1968	10,000,000
1975	100,000,000
1987	1,000,000,000 (1 GigaFlop/s, GFlop/s)
1992	10,000,000,000
1993	100,000,000,000
1997	1,000,000,000,000 (1 TeraFlop/s, TFlop/s)
2000	10,000,000,000,000
2005	131,000,000,000,000 (131 Tflop/s)

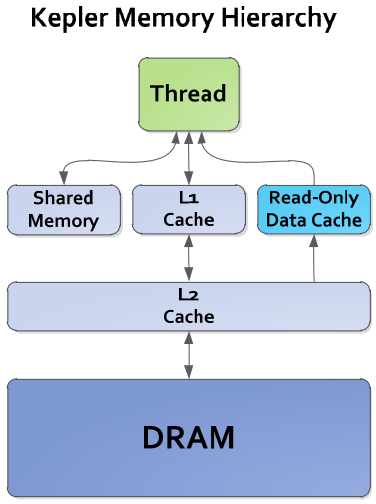
1950 1960 1970 1980 1990 2000 2010 2020

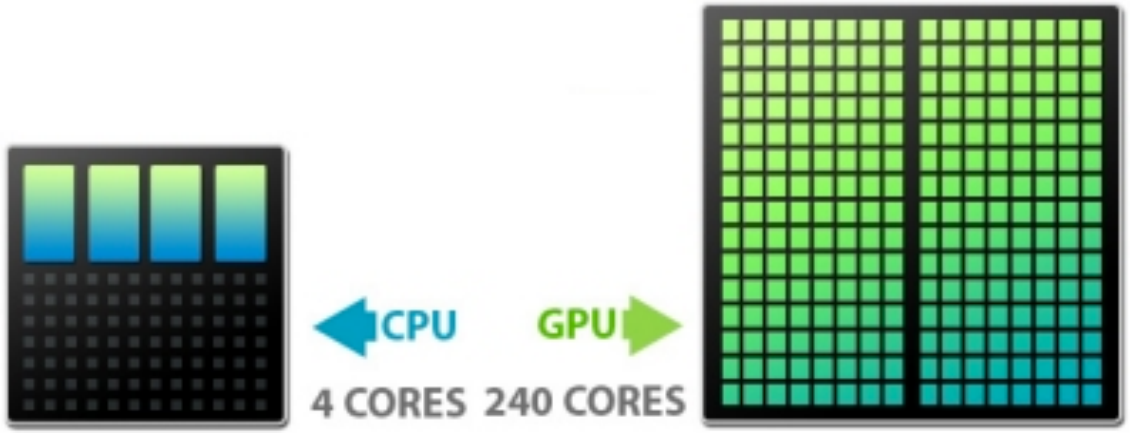
Recent Manycore GPU processors

- **Massively Parallelism, e.g. ~5k cores**



SMX: 192 single-precision CUDA cores, 64 double-precision units, 32 special function units (SFU), and 32 load/store units (LD/ST).

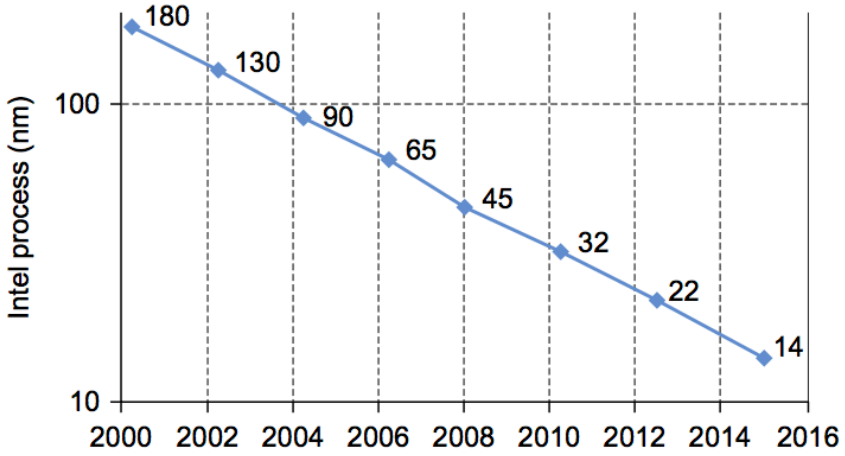




Introduction

Moore's Law enabled:

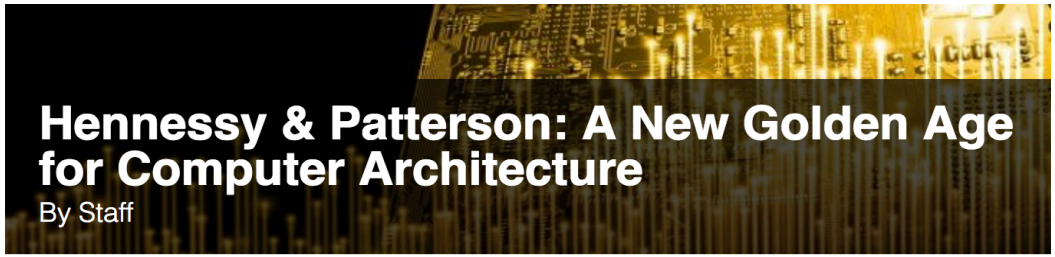
- Deep memory hierarchy, e.g. 3- or even 4-level caches
- Wide SIMD units, e.g. 512 bit SIMD register
- Deep pipelines, e.g. 10-20 stages
- Branch prediction, e.g. close to > 90% accurate rate
- Out-of-order execution to achieve data-flow and remove WAR/WAR hazards
- Speculative prefetching
- Multithreading
- Multiprocessing



- Objective:
  - Extract performance from software that is oblivious to architecture

Introduction

- **Need factor of 100 improvements in number of operations per instruction**
  - **Requires domain specific architectures**
  - **For ASICs, NRE cannot be amoratized over large volumes**
  - **FPGAs are less efficient than ASICs**



April 17, 2018

On Monday June 4, 2018, 2017 A.M. Turing Award Winners John L. Hennessy and David A. Patterson will deliver the Turing Lecture at the 45th International Symposium on Computer Architecture (ISCA) in Los Angeles.

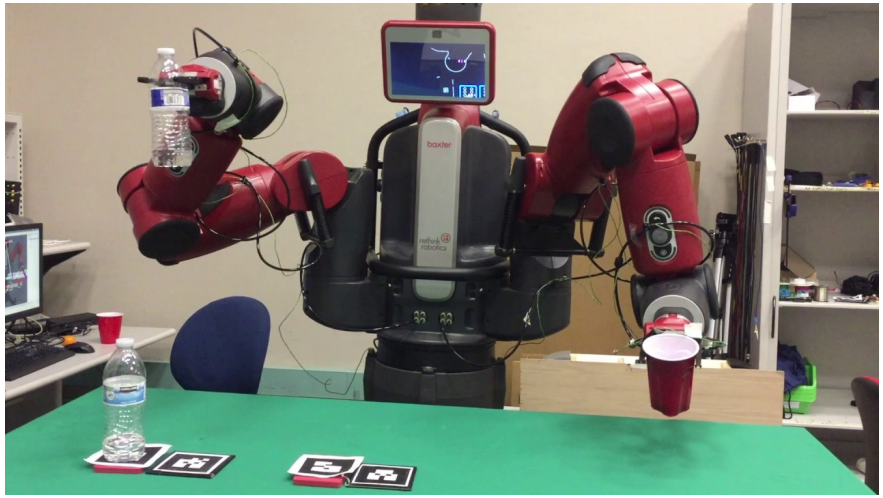
- **Video: https://www.acm.org/hennessy-patterson-turing-lecture**
- **Short summary: https://www.hpcwire.com/2018/04/17/hennessy-patterson-a-new-golden-age-for-computer-architecture**

Machine Learning Domain

On Images and Videos

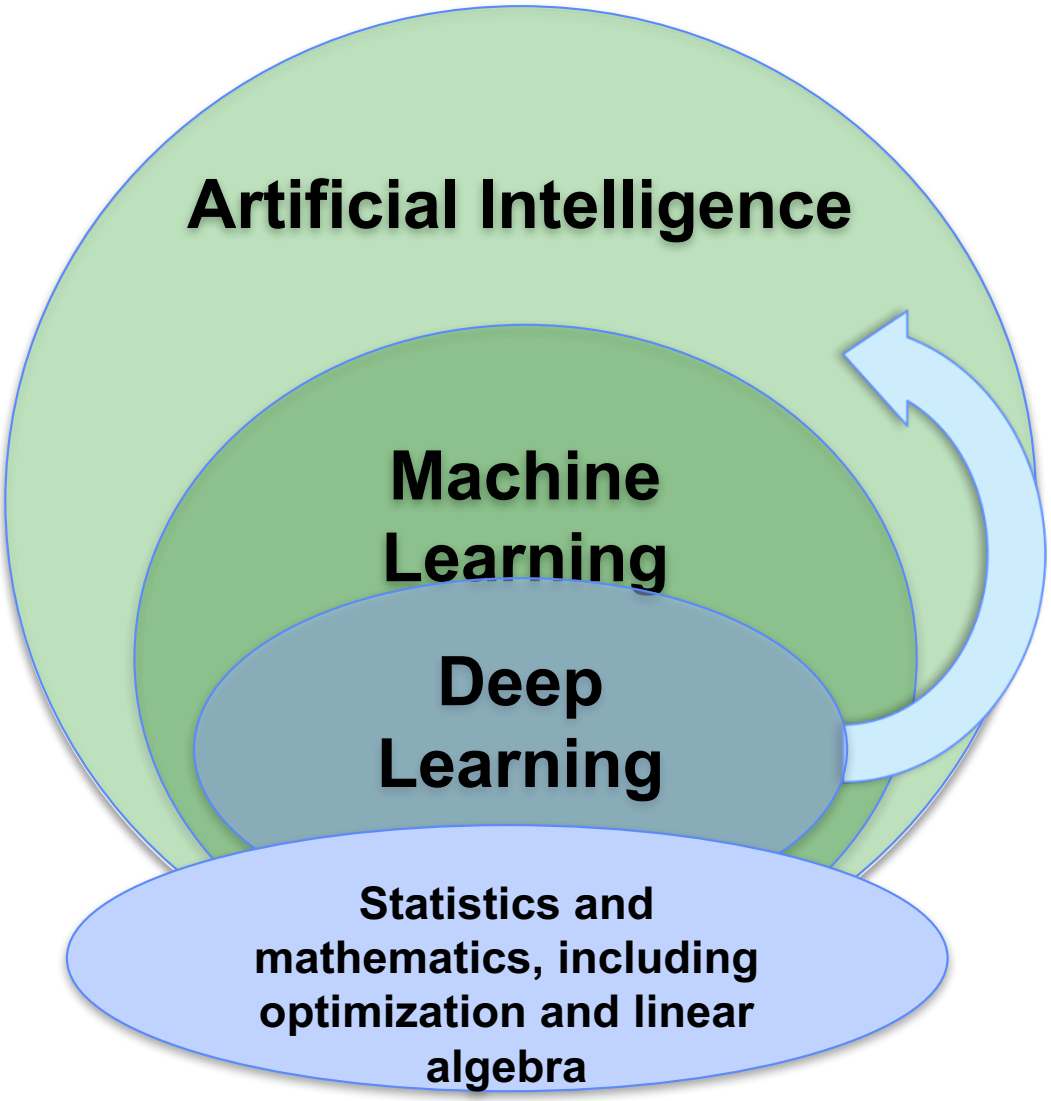
Language Understanding

Robotics



Reasoning

Artificial Intelligence, Machine Learning and Deep Learning



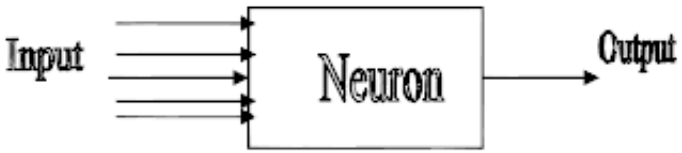
Big Data Processing

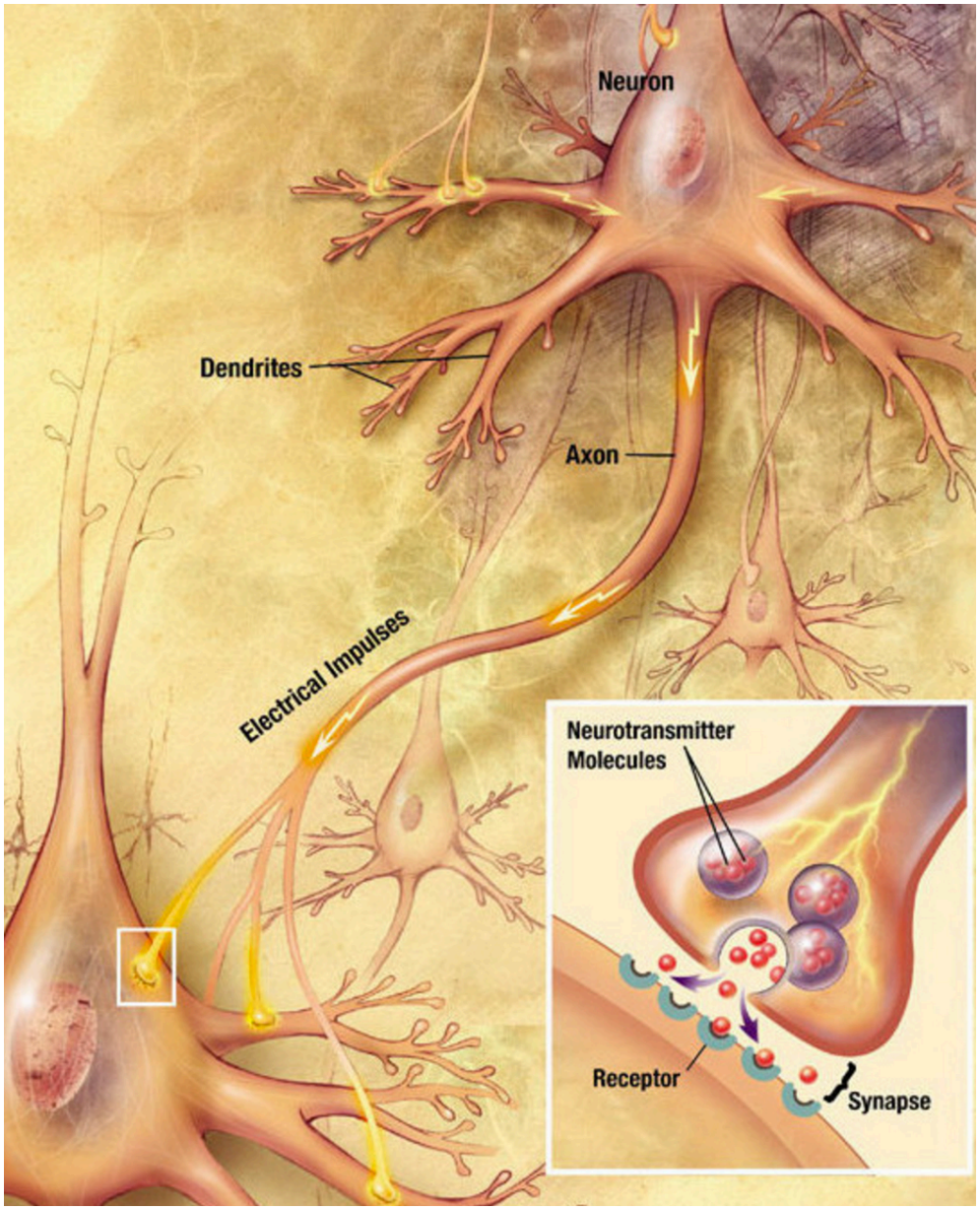
Sophisticated Algorithms

High Performance Machines

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

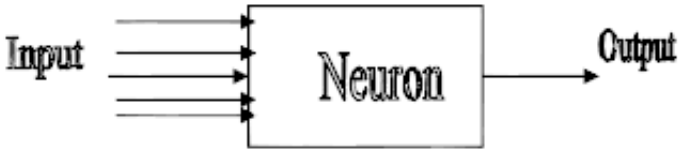


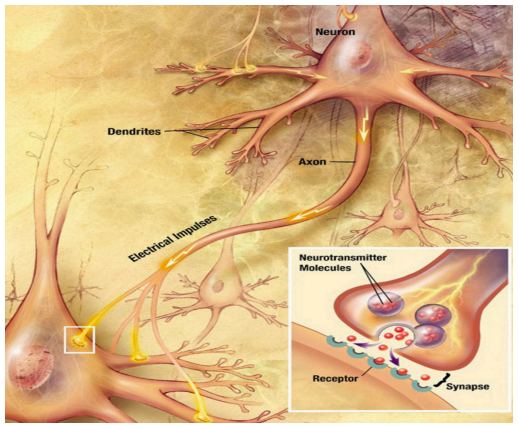


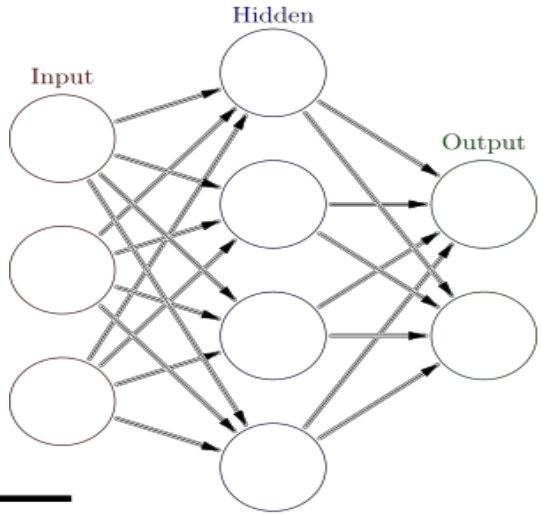
https://en.wikipedia.org/wiki/Nervous_system

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

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.

Example: 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

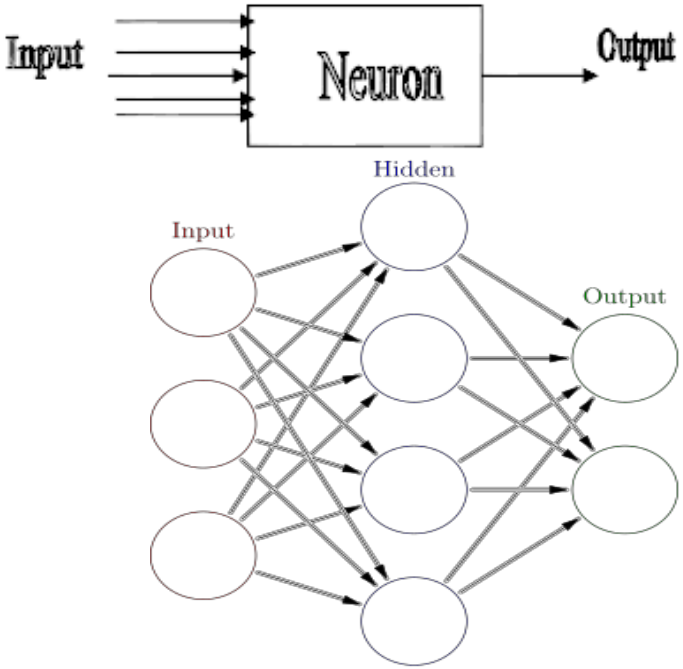
Type of data	Problem area	Size of benchmark's training set	DNN architecture	Hardware	Training time
text [1]	Word prediction (word2vec)	100 billion words (Wikipedia)	2-layer skip gram	1 NVIDIA Titan X GPU	6.2 hours
audio [2]	Speech recognition	2000 hours (Fisher Corpus)	11-layer RNN	1 NVIDIA K1200 GPU	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

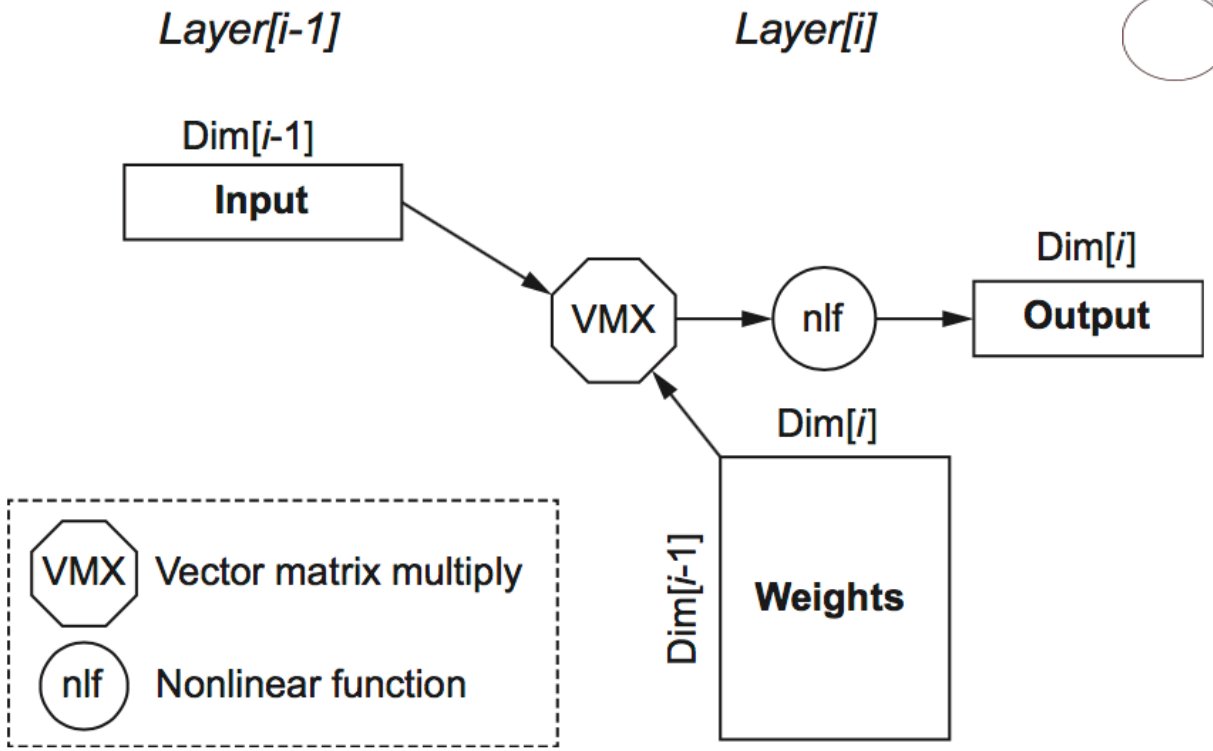
Figure 7.6 Training set sizes and training time for several DNNs (Iandola, 2016).

- Inferrence: use neural network for classification

Multi-Layer Perceptrons

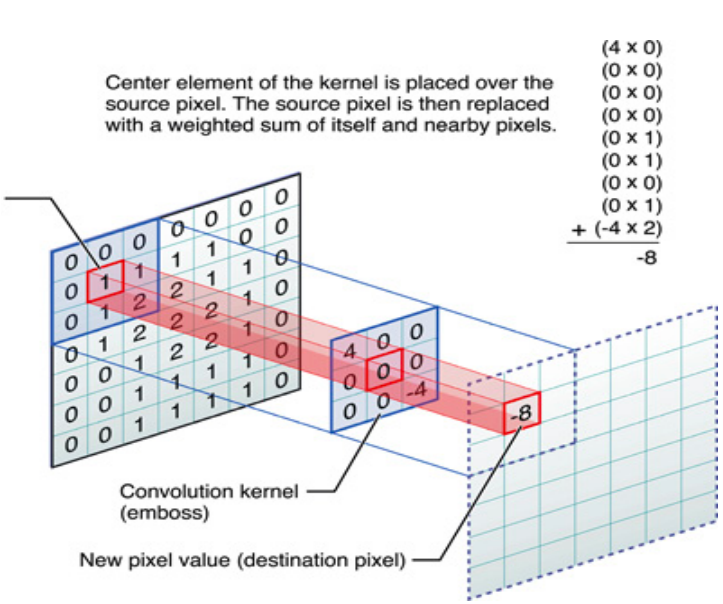
- **Parameters:**
  - **Dim[i]: number of neurons**
  - **Dim[i-1]: dimension of input vector**
  - **Number of weights: Dim[i-1] x Dim[i]**
  - **Operations: 2 x Dim[i-1] x Dim[i]**
  - **Operations/weight: 2**

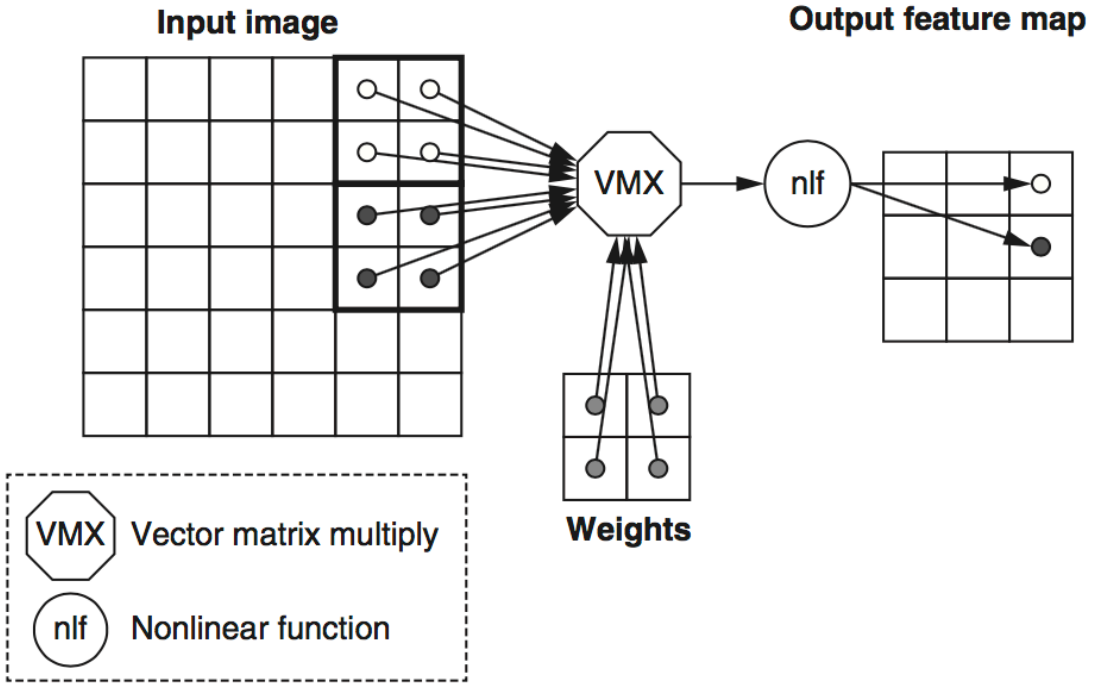




Convolutional Neural Network

- 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





Convolutional Neural Network

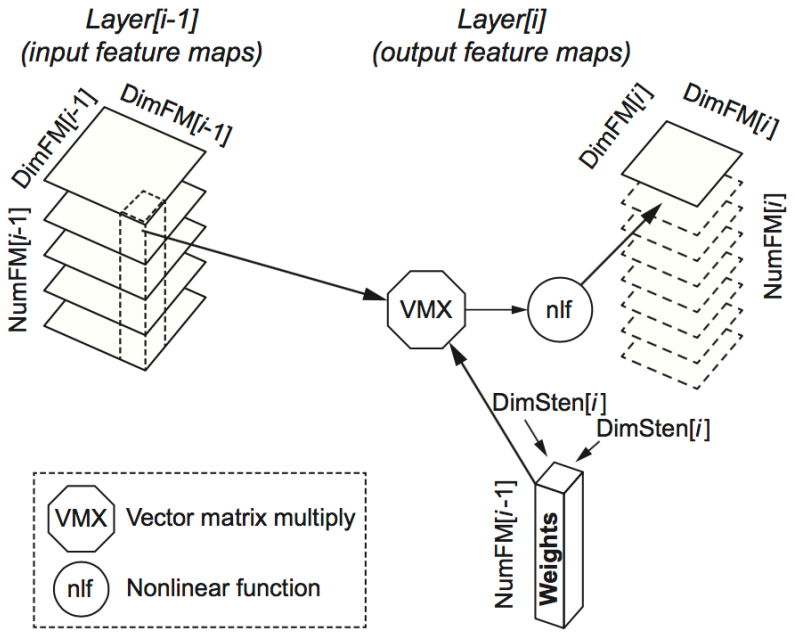


Figure 7.9 CNN general step showing input feature maps of Layer[$i-1$] on the left, the output feature maps of Layer[i] on the right, and a three-dimensional stencil over input feature maps to produce a single output feature map. Each output feature map has its own unique set of weights, and the vector-matrix multiply happens for every one. The dotted lines show future output feature maps in this figure. As this figure illustrates, the dimensions and number of the input and output feature maps are often different. As with MLPs, ReLU is a popular nonlinear function for CNNs.

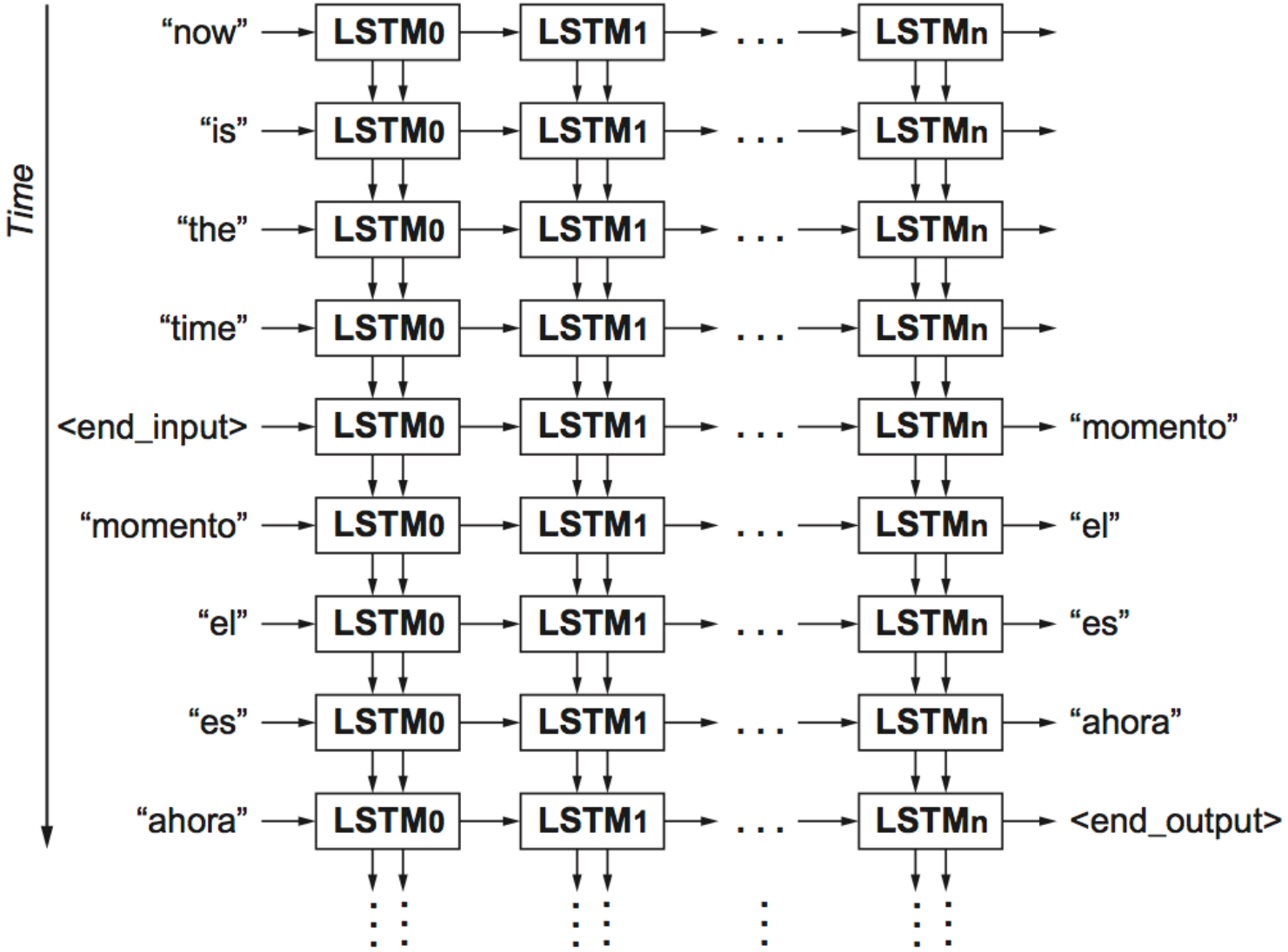
- **Parameters:**
  - DimFM[i-1]: Dimension of the (square) input Feature Map
  - DimFM[i]: Dimension of the (square) output Feature Map
  - DimSten[i]: Dimension of the (square) stencil
  - NumFM[i-1]: Number of input Feature Maps
  - NumFM[i]: Number of output Feature Maps
  - Number of neurons: NumFM[i] 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

Convolutional Neural Network

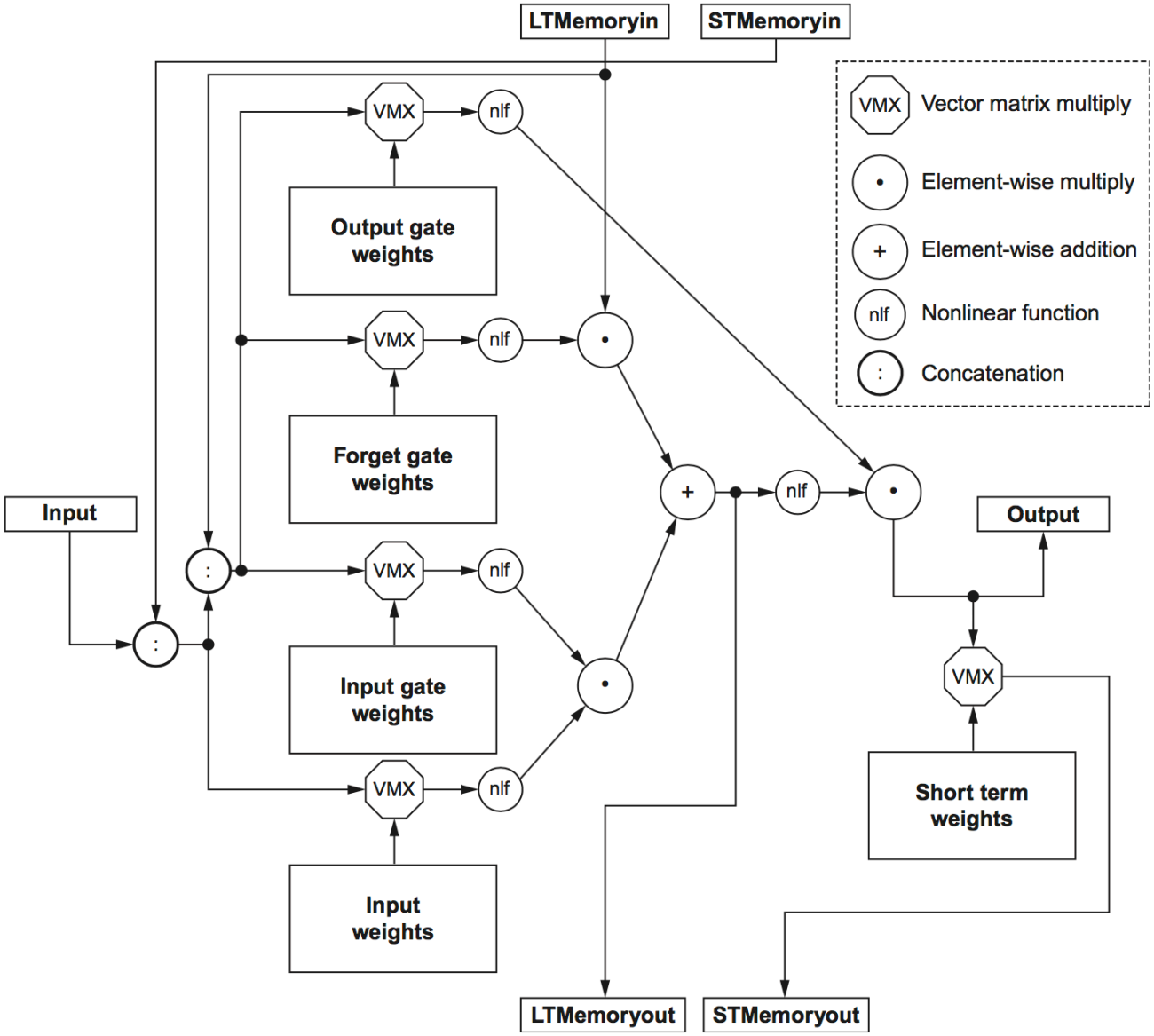
- **Batches:**
  - Reuse weights once fetched from memory across multiple inputs
  - Increases operational intensity
- **Quantization**
  - Use 8- or 16-bit fixed point
- **Summary:**
  - Need the following kernels:
    - Matrix-vector multiply
    - Matrix-matrix multiply
    - Stencil
    - ReLU
 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

Recurrent Neural Network

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

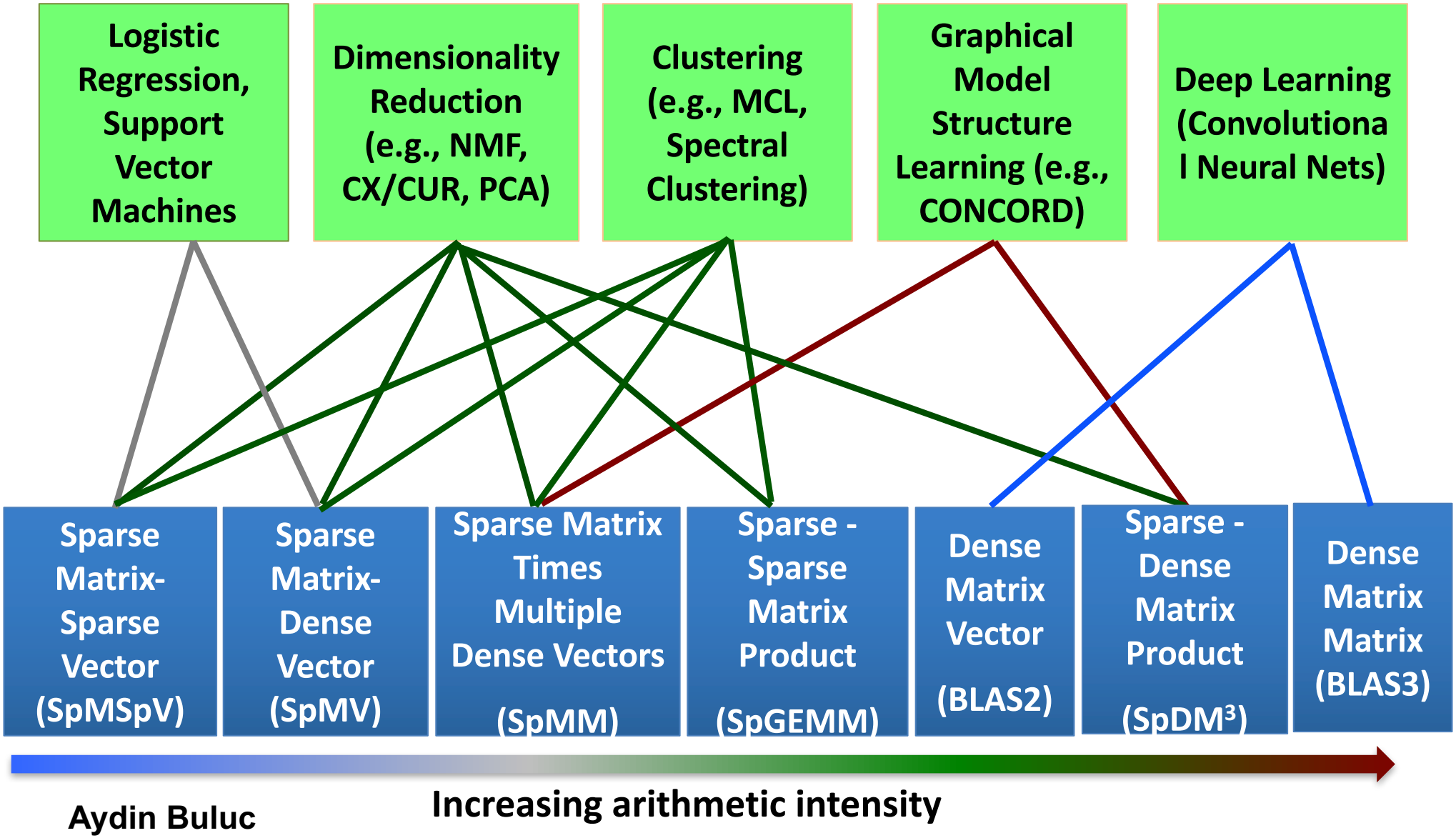


Recurrent Neural Network



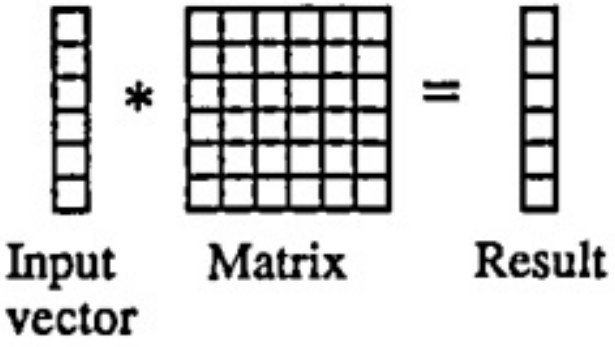
- **Parameters:**
  - Number of weights per cell: 3 x (3 x Dim x Dim)+(2 x Dim x Dim) + (1 x Dim x Dim) = 12 x Dim^2
  - Number of operations for the 5 vector-matrix multiplies per cell: 2 x Number of weights per cell = 24 x Dim^2
  - Number of operations for the 3 element-wise multiplies and 1 addition (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

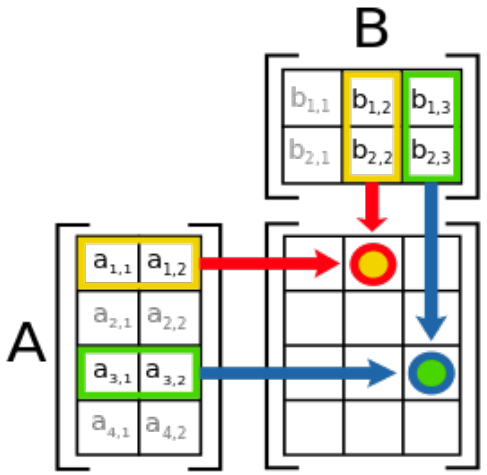
Machine Learning Mapping to Linear Algebra

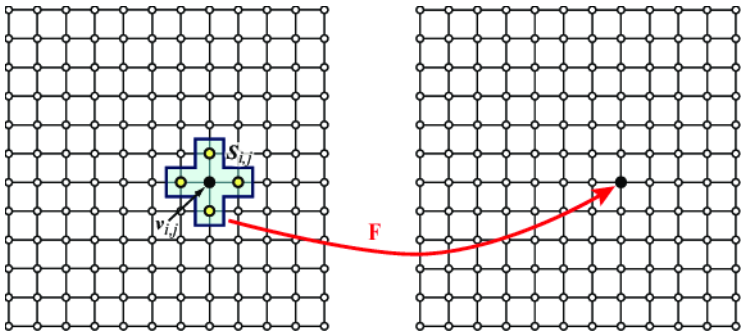


Summary

- **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**

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

Guidelines for DSAs

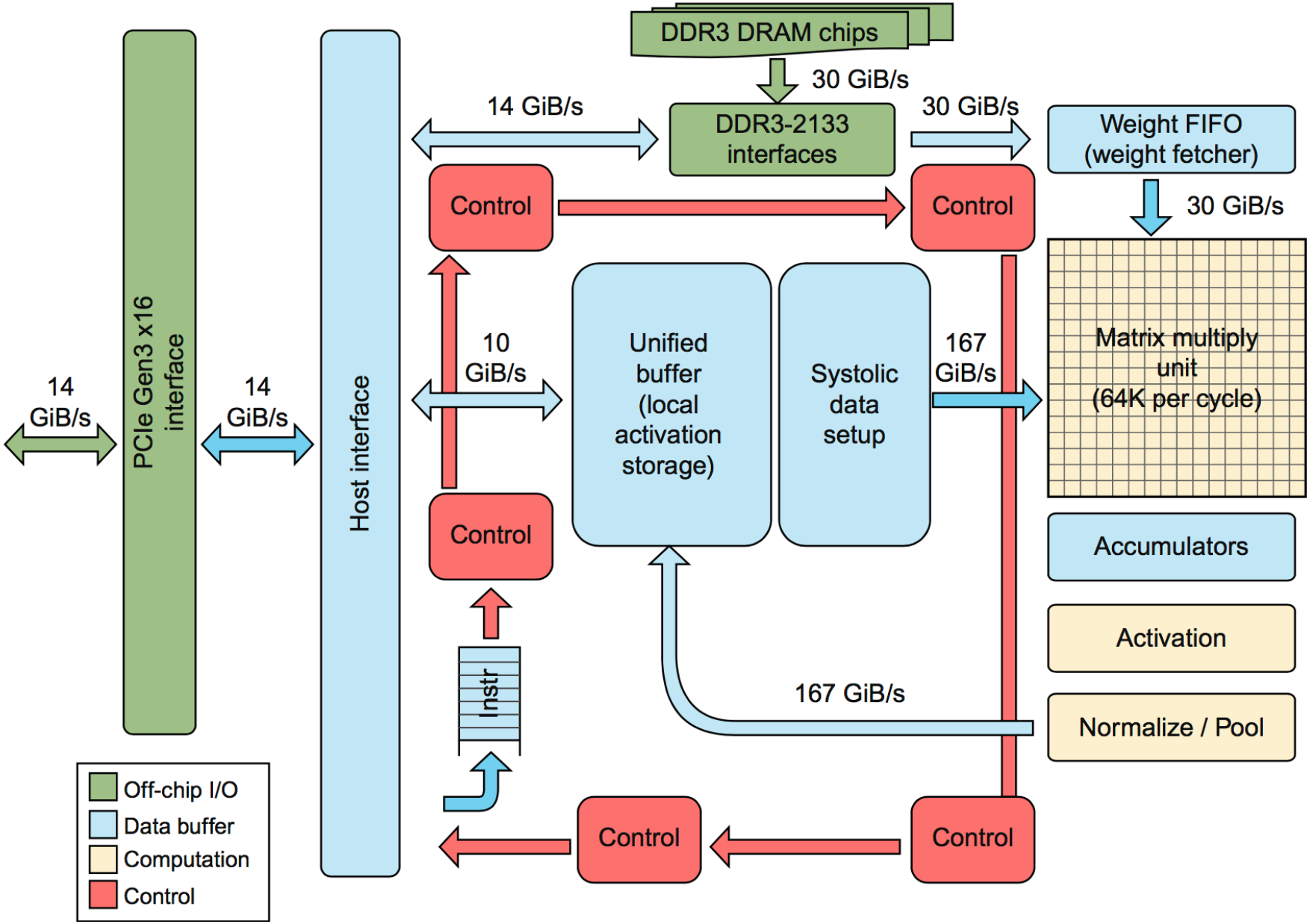
Guideline	TPU	Catapult	Crest	Pixel Visual Core
Design target	Data center ASIC	Data center FPGA	Data center ASIC	PMD ASIC/SOC IP
1. Dedicated memories	24 MiB Unified Buffer, 4 MiB Accumulators	Varies	N.A.	Per core: 128 KiB line buffer, 64 KiB P.E. memory
2. Larger arithmetic unit	65,536 Multiply-accumulators	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.

Tensor Processing Unit

- 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

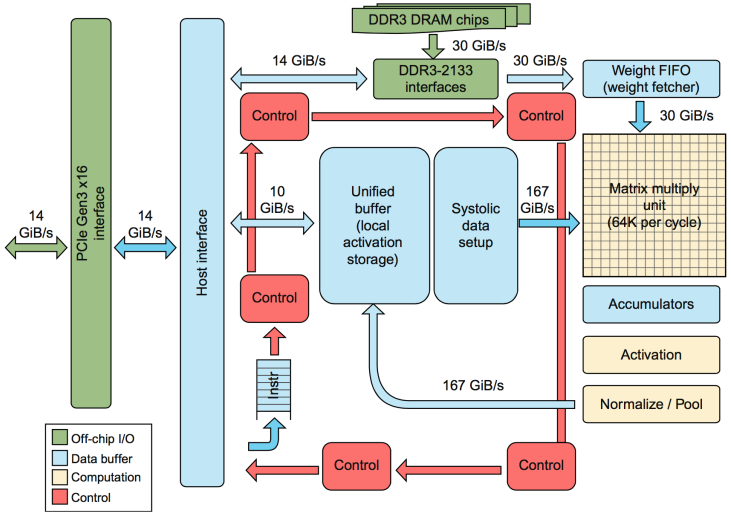
Tensor Processing Unit



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-byte-wide (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



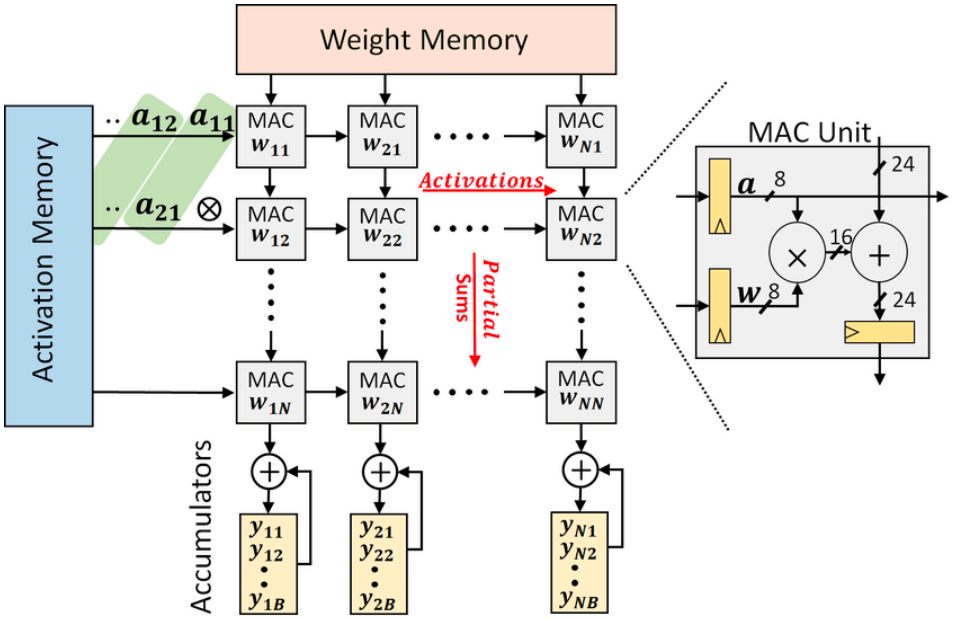
- TPU is CISC tradition, CPI are typically 10-20
- No program counter, no branch instructions
- About a dozen instructions, five key ones:

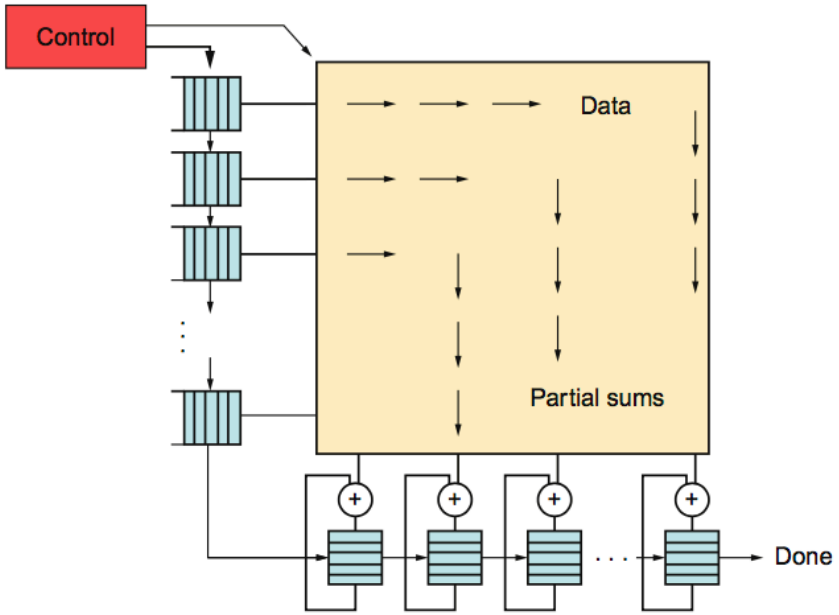
- Read_Host_Memory
  - Reads data from the CPU memory into the unified buffer
- Read_Weights
  - Reads weights from the Weight Memory into the Weight FIFO as input to the Matrix Unit

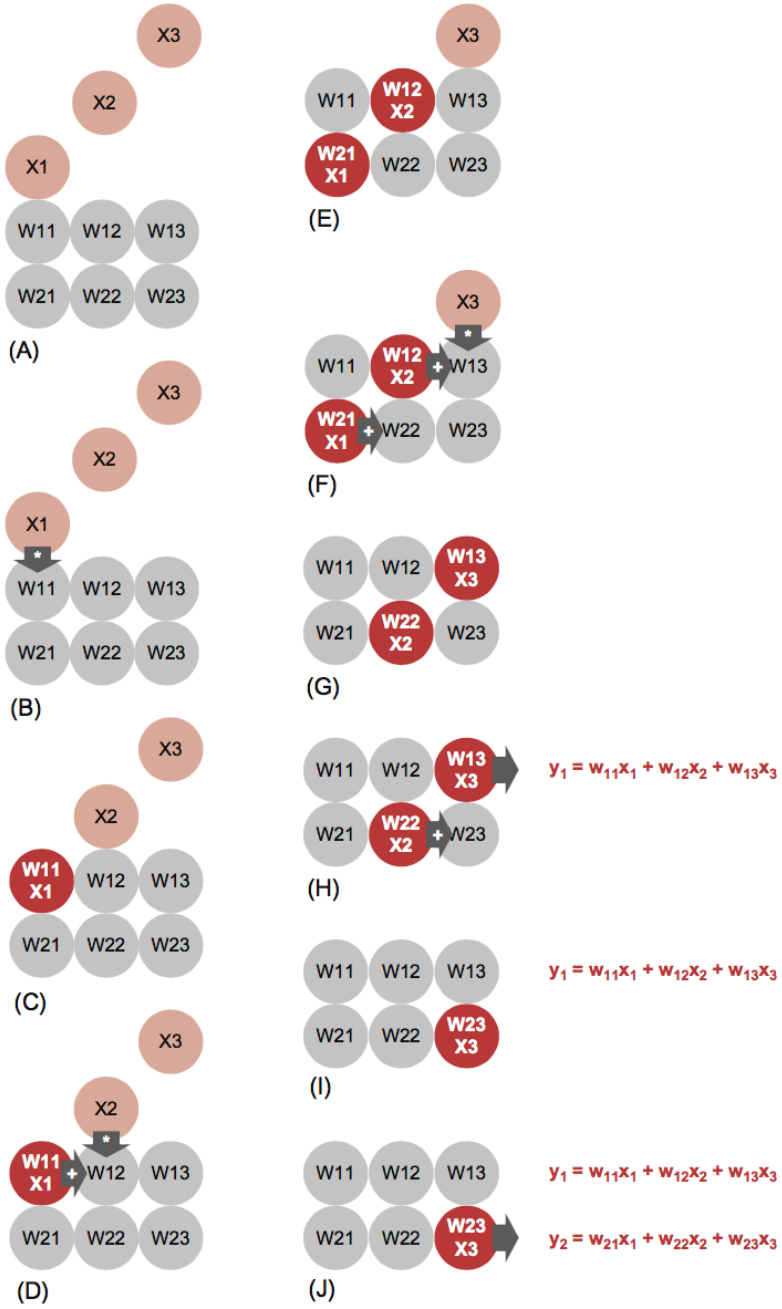
- MatrixMatrixMultiply/Convolve
  - Perform a MM multiply, a MV multiply, an element-wise MM, an element-wise MV, or a convolution from the Unified Buffer into the accumulators
  - Takes a variable-sized B*256 input, multiplies it by a 256x256 constant input, and produces a B*256 output, taking B pipelined 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 Microarchitecture – Systolic Array







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 mm^2.**

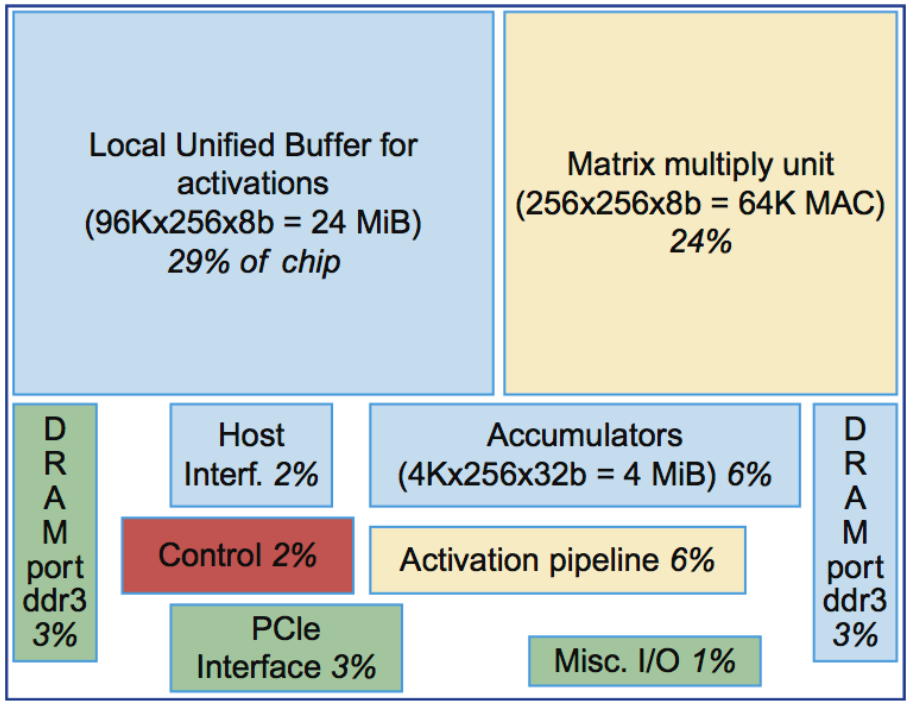


Figure 7.15 Floor plan of TPU die. The shading follows Figure 7.14. The light data buffers are 37%, the light computation units are 30%, the medium I/O is 10%, and the dark control is just 2% of the die. Control is much larger (and much more difficult to design) in a CPU or GPU. The unused white space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

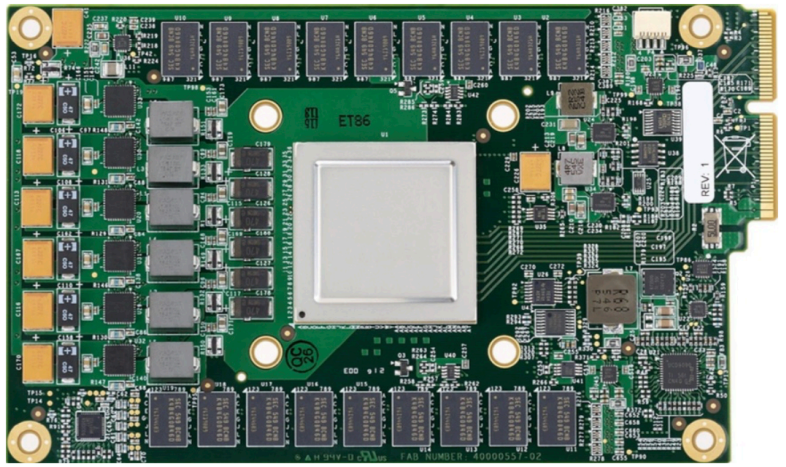
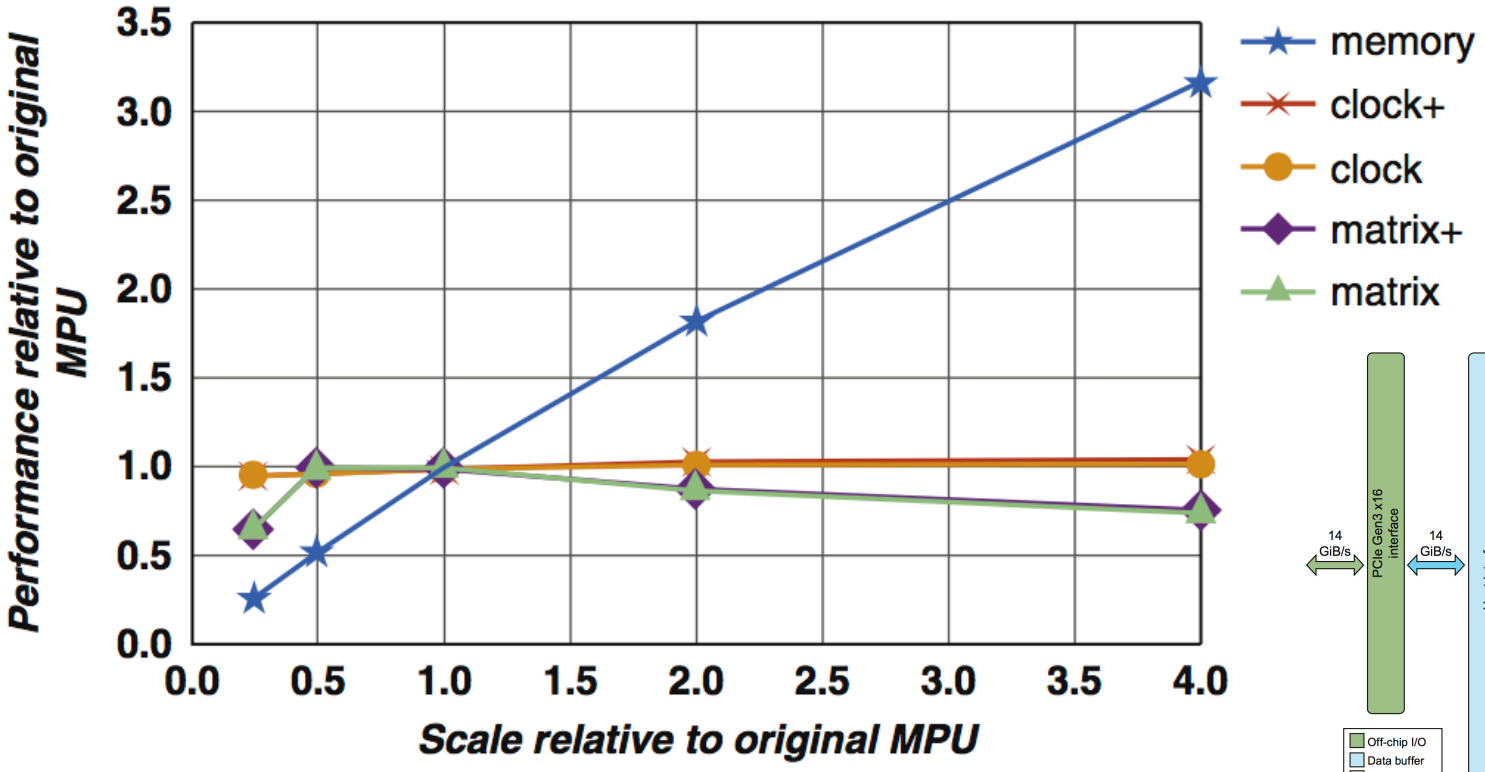
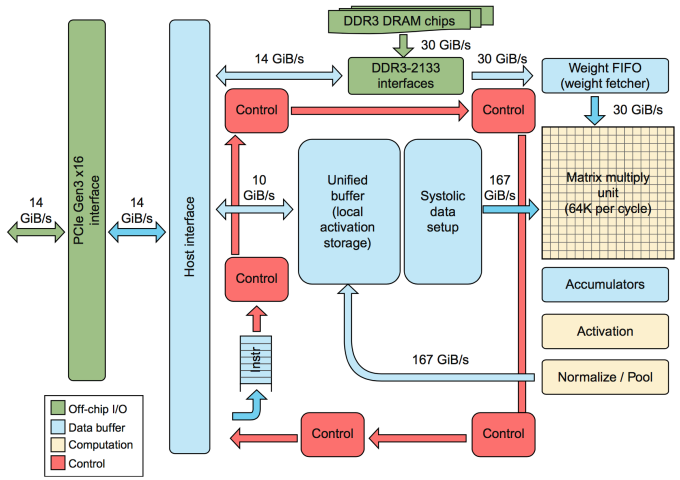


Figure 7.16 TPU printed circuit board. It can be inserted into the slot for an SATA disk in a server, but the card uses the PCIe bus.

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.





The TPU and the Guidelines

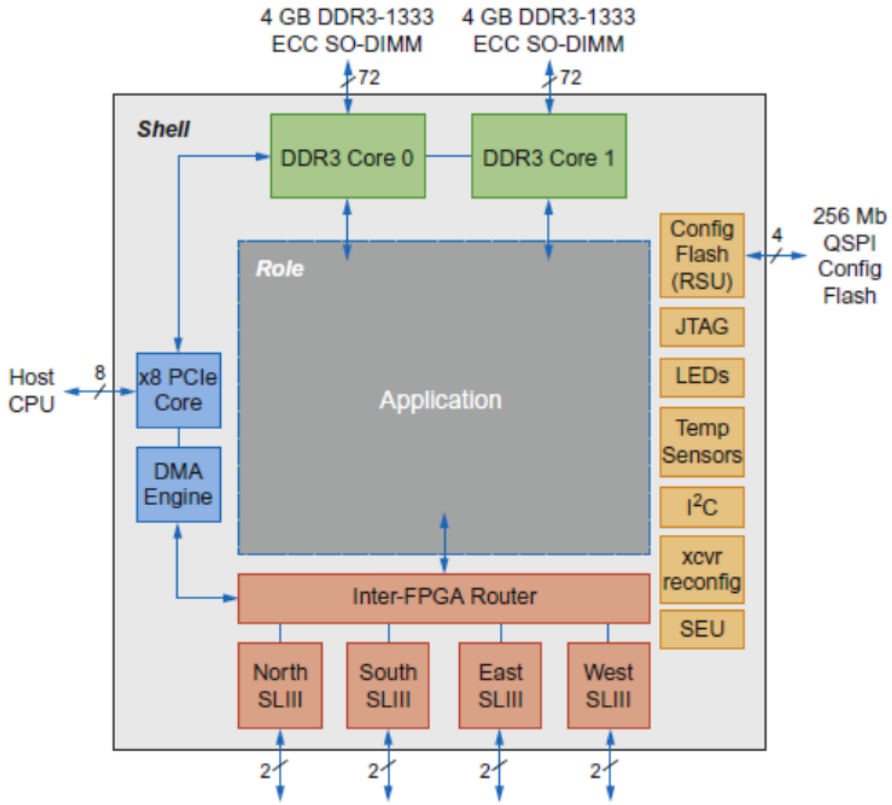
- Use dedicated memories
  - 24 MiB dedicated buffer, 4 MiB accumulator buffers
- Invest resources in arithmetic units and dedicated memories
  - 60% of the memory and 250X the arithmetic units of a server-class CPU
- Use the easiest form of parallelism that matches the domain
  - Exploits 2D SIMD parallelism 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

Class Lecture Stops Here

- **Other Important topics of this chapter**
  - **Microsoft Catapult, FPGA-based DSA solution**
  - **Intel Crest, more details needed**
  - **Google Pixel Visual Core: DSA for stencil, very interesting**
  - **Heterogeneity and Open Instruction Set (RISC-V)**
    - » **Checkout RISC-V Summit, 12/03 – 12/06 2018, https://tmt.knect365.com/risc-v-summit/**
  - **CPU vs GPUs vs DNN accelerators**
    - » **Performance comparison and Roofline Model**

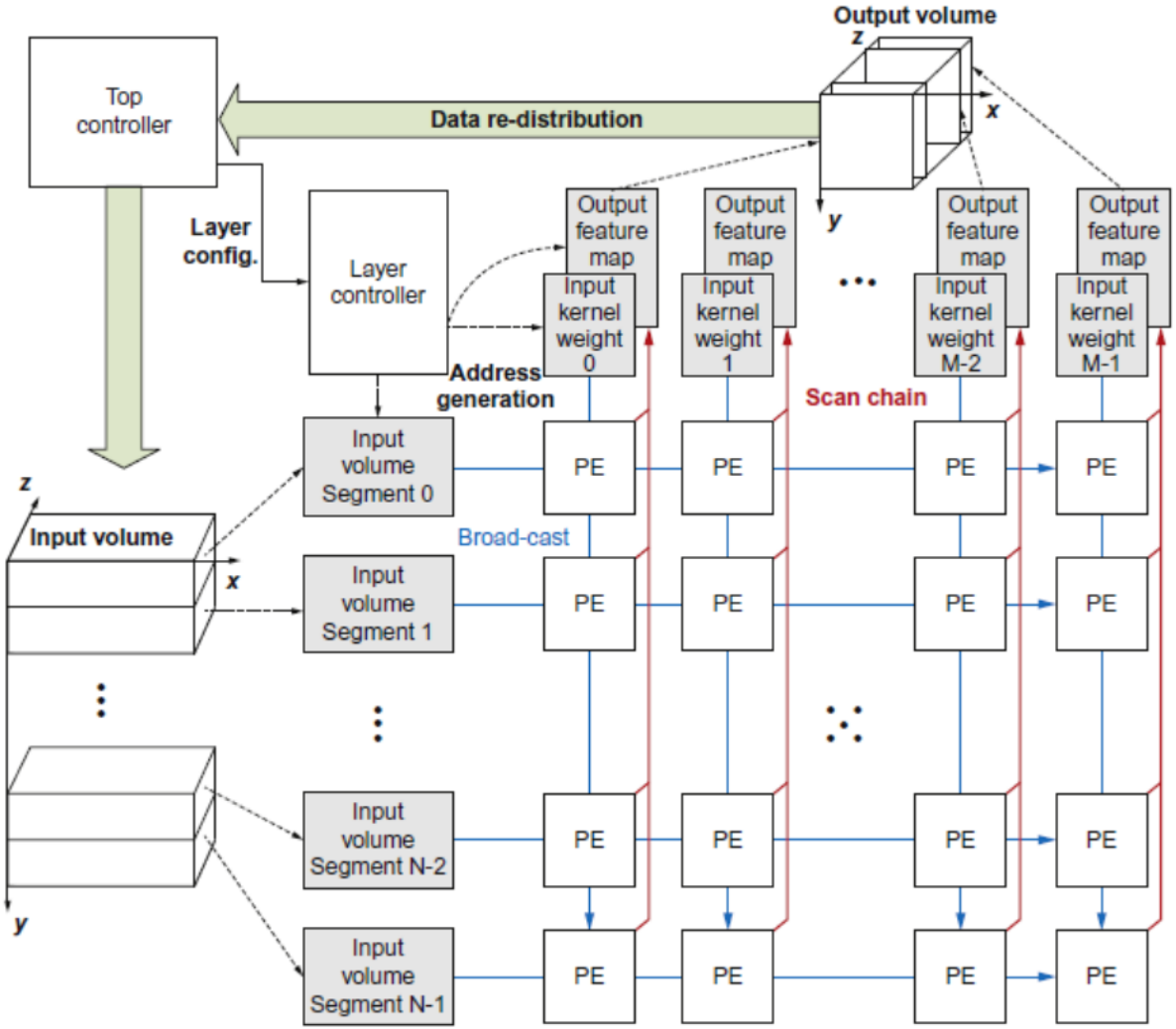
Microsoft Catapult

- **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)**
  - **FPGA (unconfigured) has 3962 18-b memory**
  - **Programmed in Verilog RTL**
  - **Shell is 23% of the FPGA**

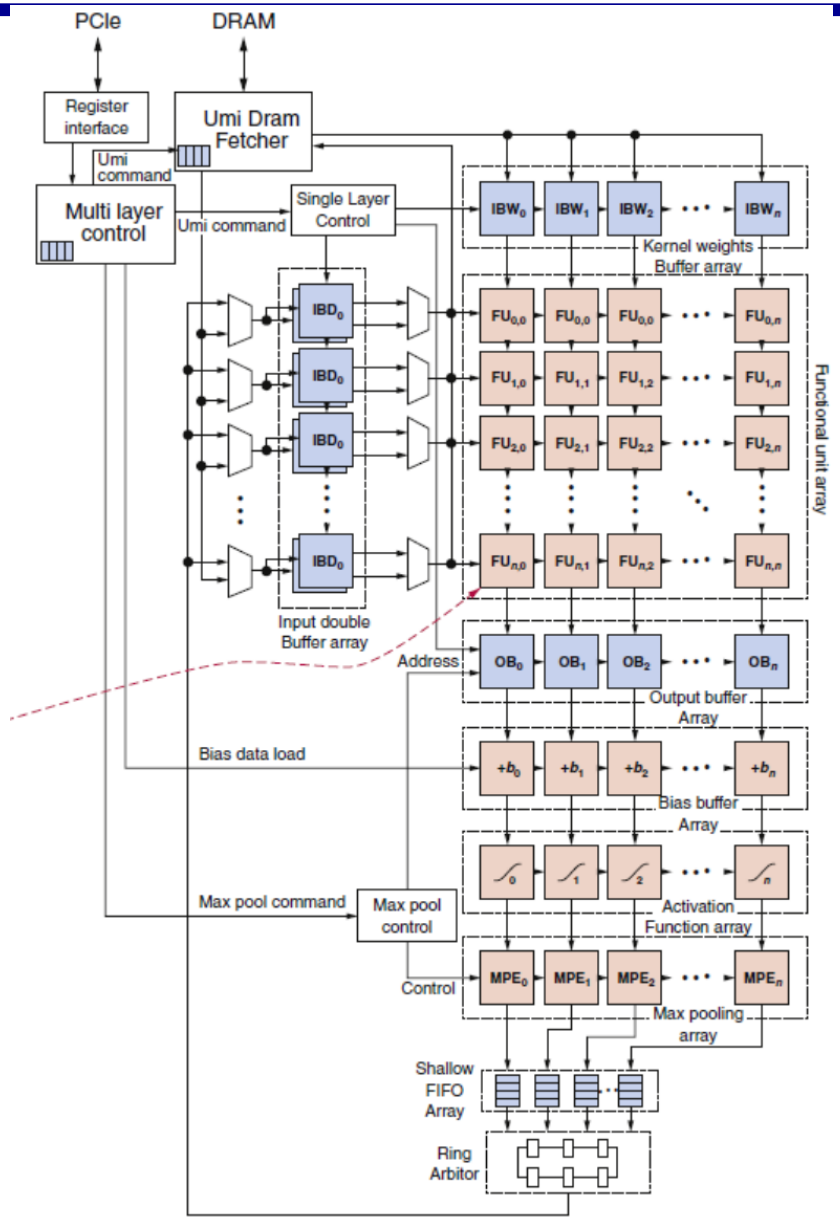


Microsoft Catapult: CNN

- CNN accelerator, mapped across multiple FPGAs

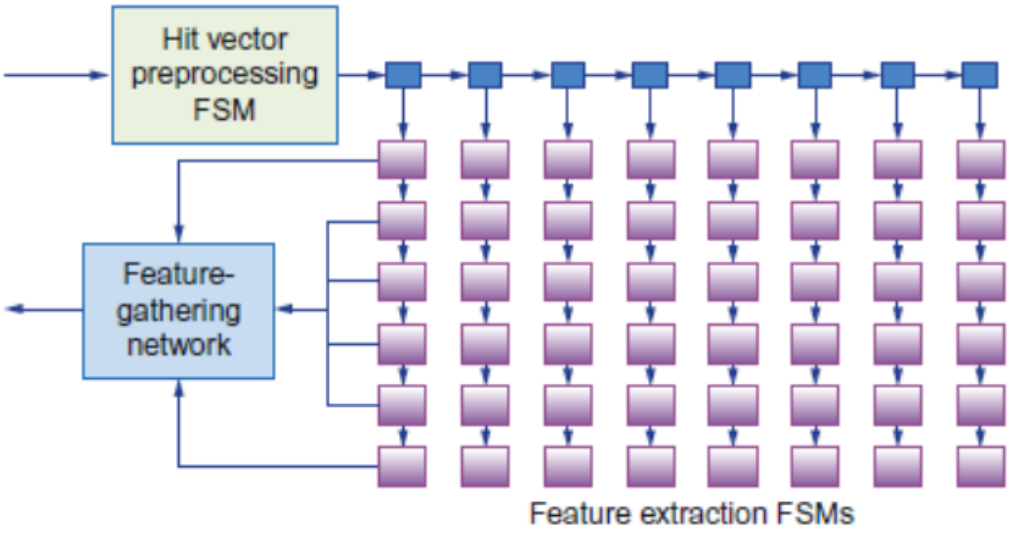


Microsoft Catapult: CNN



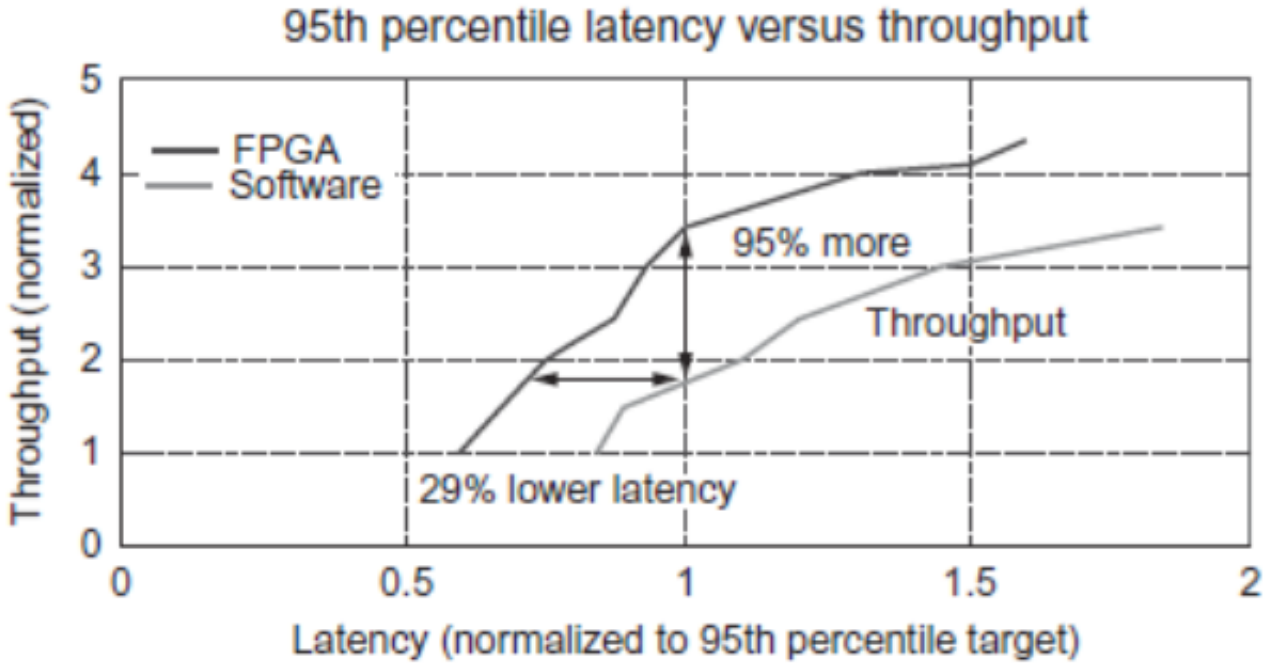
Microsoft Catapult: Search Ranking

- **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**



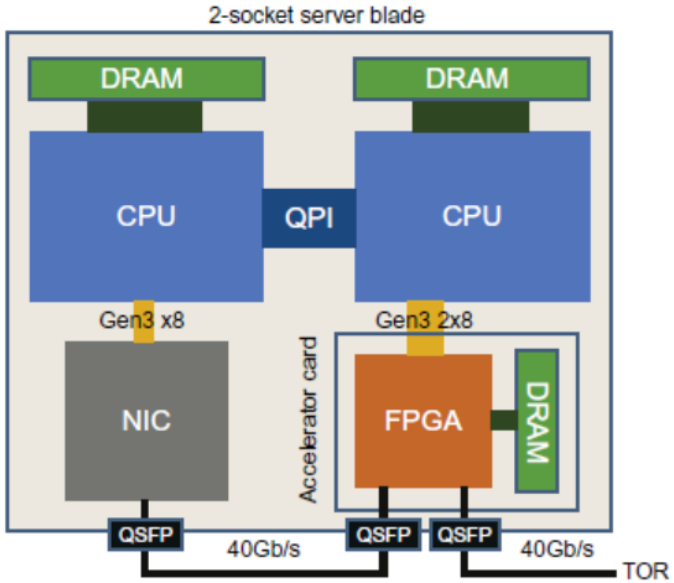
Microsoft Catapult: Search Ranking

- **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



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

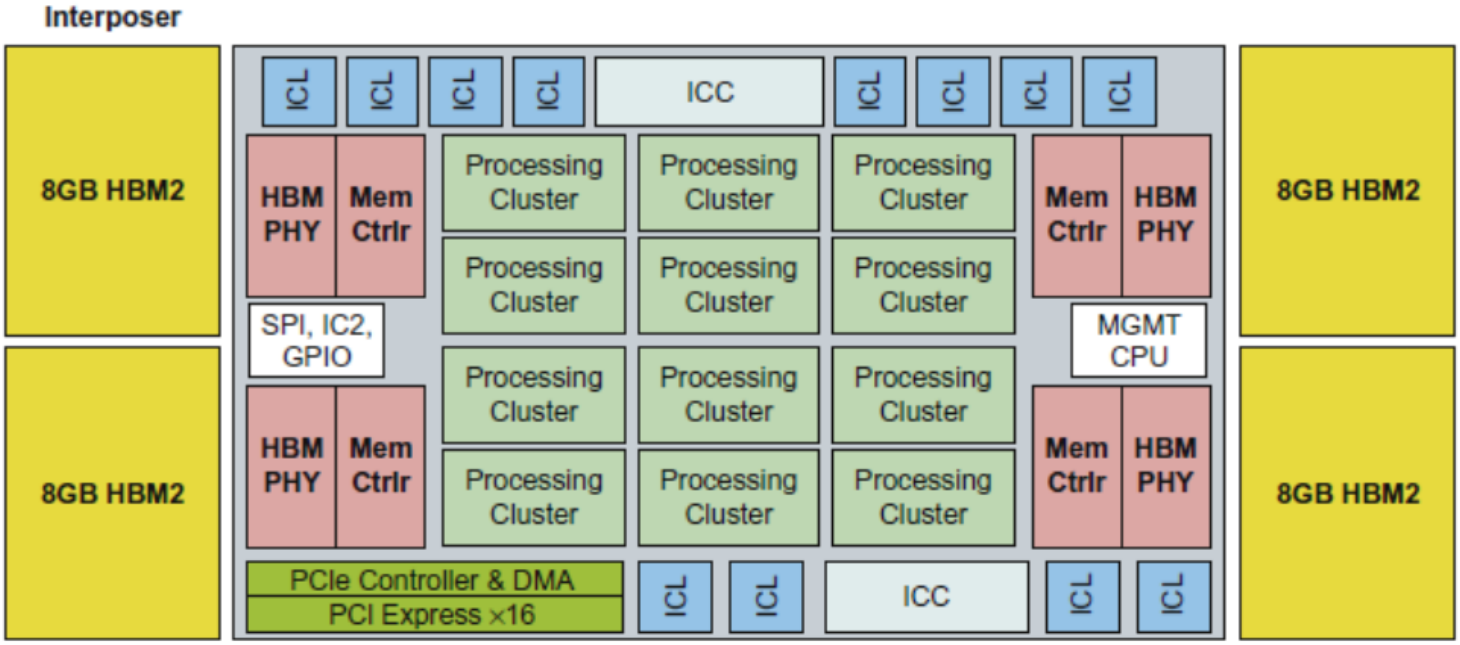


Catapult and the Guidelines

- 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

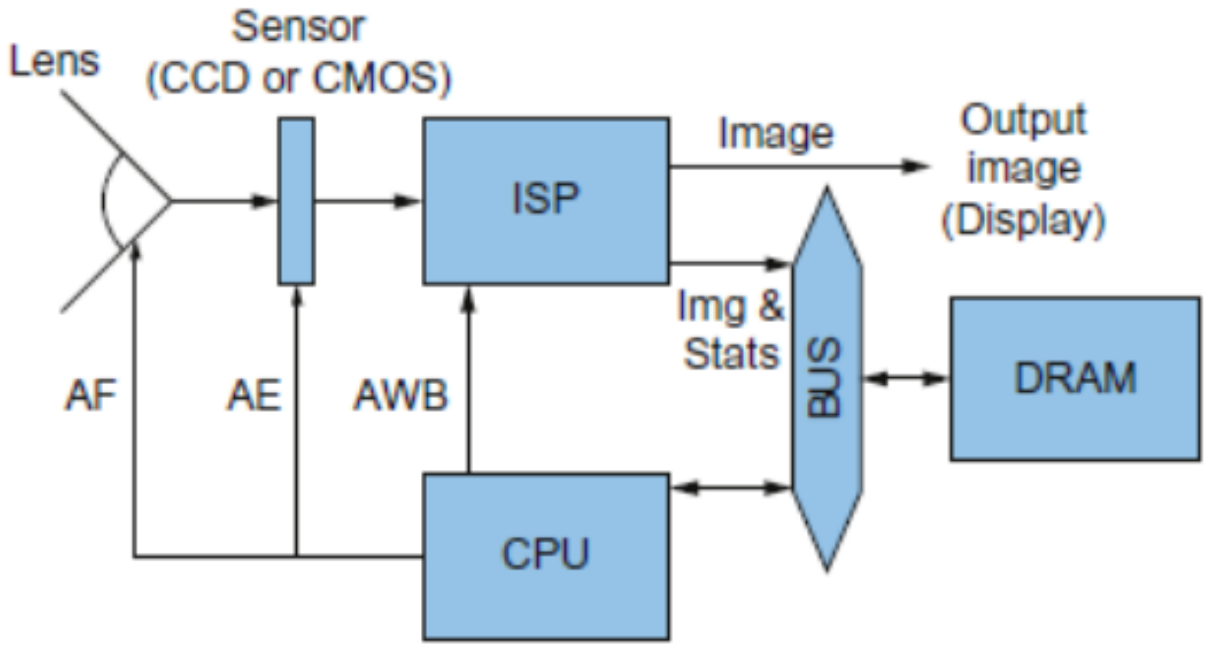
Intel Crest

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



Pixel Visual Core

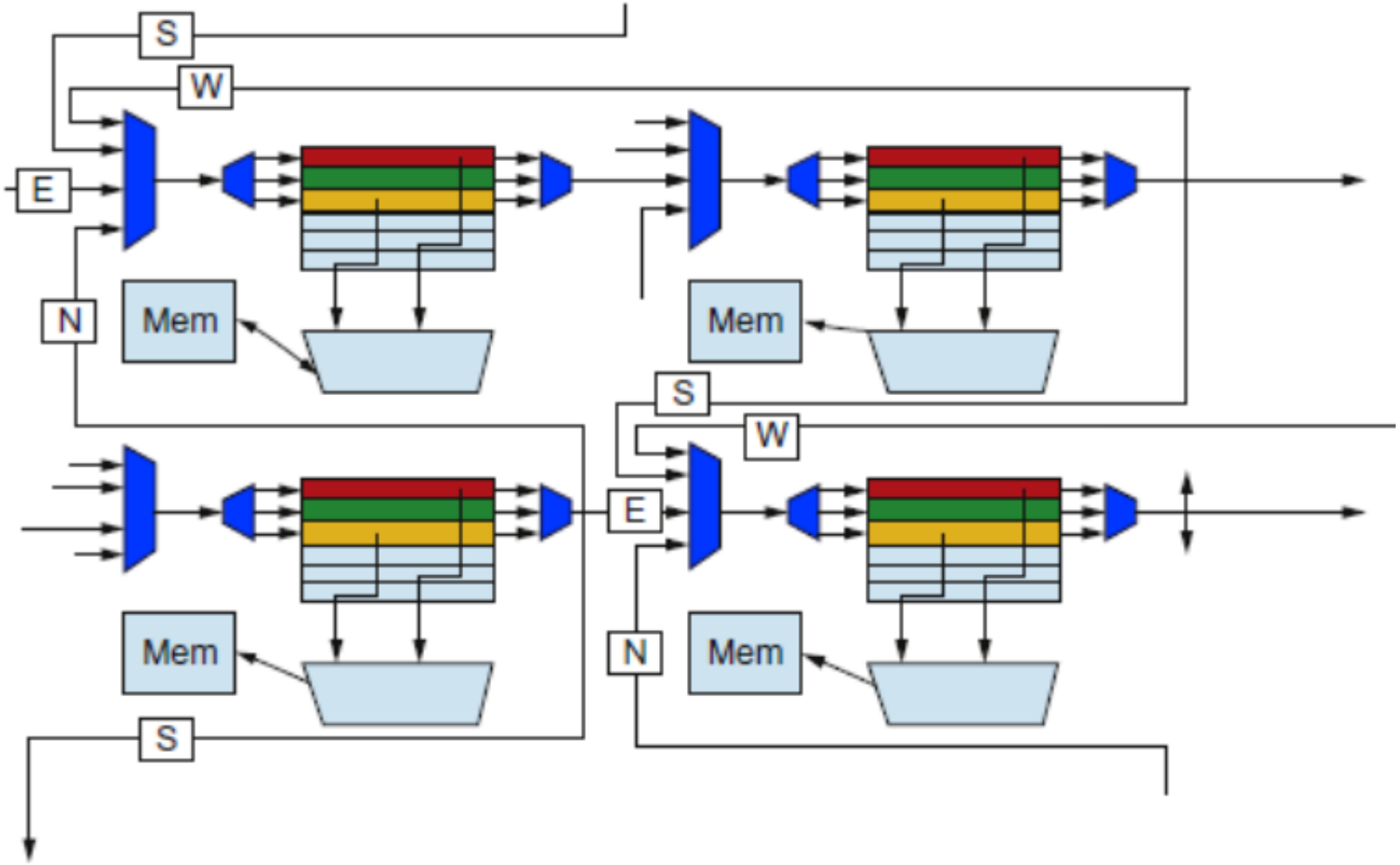
- **Pixel Visual Core**
  - **Image Processing Unit**
  - **Performs stencil operations**
  - **Decended from Image Signal processor**



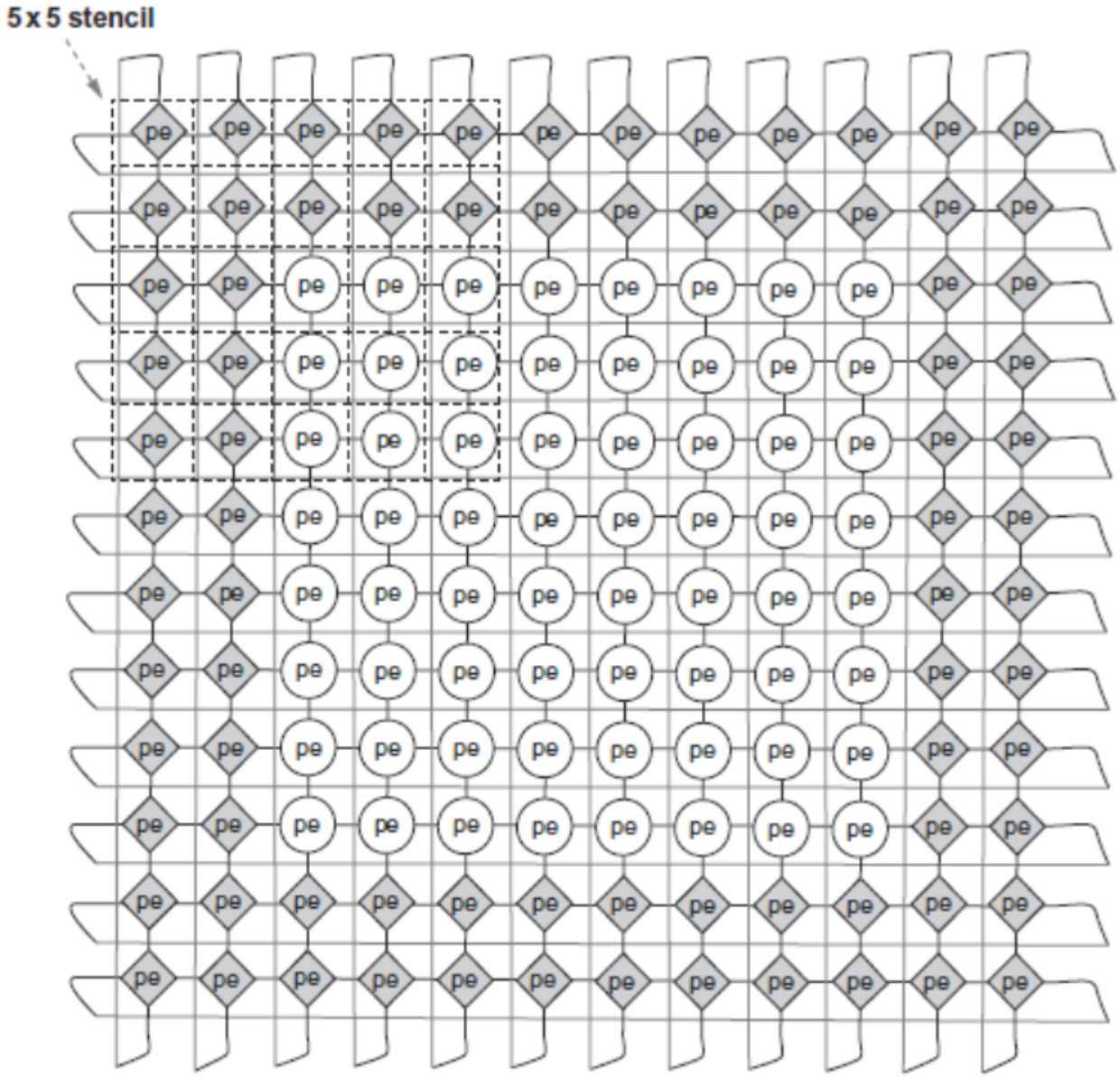
Pixel Visual Core

- 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

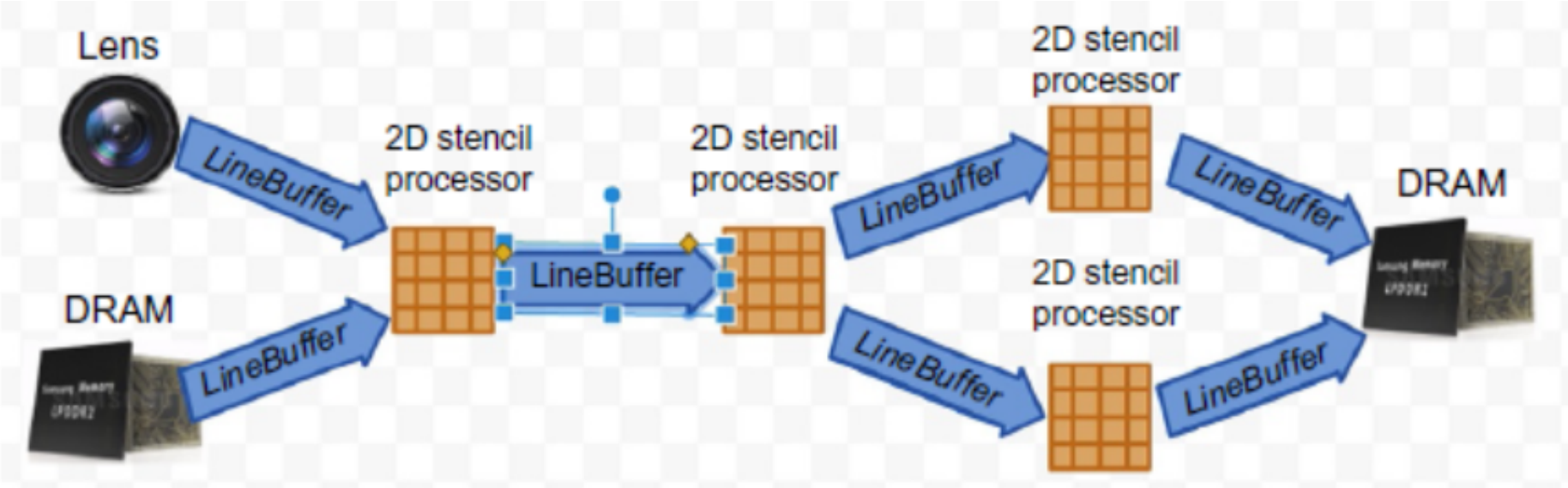
Pixel Visual Core



Pixel Visual Core



Pixel Visual Core



Visual Core and the 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

Fallacies and Pitfalls

- It costs $100 million to design a custom chip
- Performance counters added as an afterthought
- Architects are tackling the right DNN tasks
- For DNN hardware, inferences per second (IPS) is a fair summary performance metric
- Being ignorant of architecture history when designing an DSA