# Computer Architecture

## Lecture 28: Systolic Array Architectures

Prof. Onur Mutlu
ETH Zürich

Fall 2023

1 February 2024

# Bachelor's Seminar in Computer Architecture

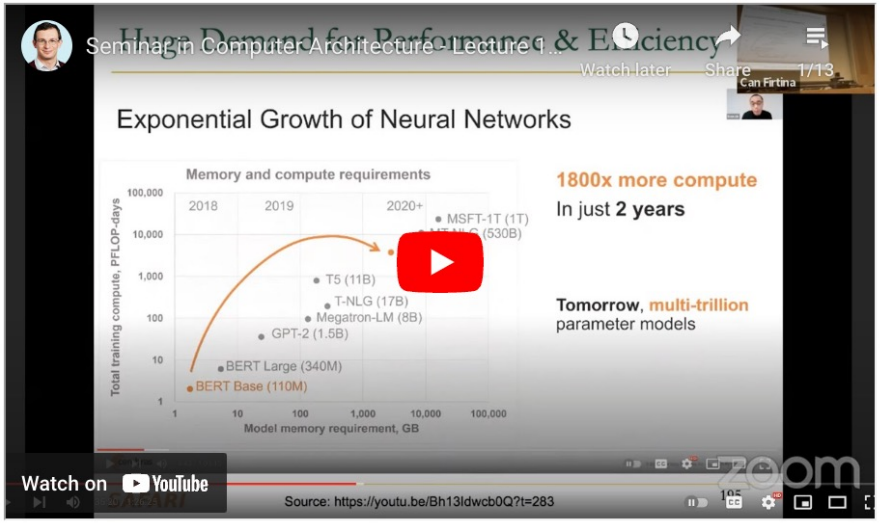
- Fall 2023 (offered every Fall and Spring Semester)
- 2 credit units

- **Rigorous seminar on fundamental and cutting-edge topics in computer architecture**
- Critical paper presentation, review, and discussion of seminal and cutting-edge works in computer architecture
  - We will cover many ideas & issues, analyze their tradeoffs, perform **critical thinking** and **brainstorming**

- Participation, presentation, synthesis report, lots of discussion
- You can register for the course online
- **https://safari.ethz.ch/architecture_seminar**


## Fall 2021 Lectures/Schedule

| Week | Date | Livestream | Lecture | Readings | Assignments |
|---|---|---|---|---|---|
| W1 | 23.09 Thu. | YouTube Live | L1a: Course Logistics (PDF) (PPT) | Suggested | |
| | | | L1b: Introduction and Basics (PDF) (PPT) | Suggested | |
| | | | L1c: Architectural Design Fundamentals (PDF) (PPT) YouTube Video | Suggested | |
| W2 | 30.09 Thu. | YouTube Live | L2: GateKeeper (PDF) (PPT) | Suggested | |
| W3 | 07.10 Thu. | YouTube Live | L3: RowClone (Processing using DRAM) (PDF) (PPT) | Suggested | |
| W4 | 14.10 Thu. | YouTube Live | L4: Memory Channel Partitioning (PDF) (PPT) | Suggested | |
| W5 | 4.11 Thu. | YouTube Live | S1.1: Bottleneck Identification and Scheduling in Multithreaded Applications, ASPLOS 2012. (PDF) (PPT) | Mentioned | |
| W6 | 11.11 Thu. | | S2.1: Profiling a Warehouse-Scale Computer, ISCA 2015. (PDF) (PPT) | Mentioned | |
| | | YouTube Live | S2.2: Understanding Sources of Inefficiency in General-Purpose Chips, ISCA 2010. (PDF) (ODP) (PPT) | Mentioned | |
| W7 | 18.11 Thu. | YouTube Live | S3.1: Pythia: A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning, MICRO2021 (PDF) (PPT) | Mentioned | |
| | | | S3.2: Branch Runahead: An Alternative to Branch Prediction for Impossible to Predict Branches, MICRO 2021 (PDF) (PPT) | Mentioned | |
| W8 | 25.11 Thu. | YouTube Live | S4.1: Very Long Instruction Word Architectures and the ELI-512, ISCA1983 (PDF) (KEY) | Mentioned | |
| | | | S4.2: A Case for Richer Cross-layer Abstractions: Bridging the Semantic Gap with Expressive Memory, ISCA 2018. (PDF) (ODP) | Mentioned | |
| W9 | 02.12 Thu. | YouTube Live | S5.1: Quantifying Server Memory Frequency Margin and Using It to Improve Performance in HPC Systems, ISCA 2021 (PDF) (PPT) | Mentioned | |
| | | | S5.2: SIMDRAM: An End-to-End Framework for Bit-Serial SIMD Computing in DRAM, ASPLOS 2021. (PDF) (KEY) | Mentioned | |
| W10 | 09.12 Thu. | YouTube Live | S6.1: TRRespass: Exploiting the Many Sides of Target Row Refresh, IEEE S&P, 2020 (PDF) | Mentioned | |
| | | | S6.2: BlockHammer: Preventing RowHammer at Low Cost by Blacklisting Rapidly-Accessed DRAM Rows, HPCA 2021. (PDF) (PPT) | Mentioned | |
| W11 | 16.12 Thu. | YouTube Live | S7.1: RAMBleed: Reading Bits in Memory Without Accessing Them, IEEE Symposium on Security and Privacy, 2020. (PPT) (PDF) | Mentioned | |
| | | | S7.2: Using Memory Errors to Attack a Virtual Machine, IEEE S&P 2003. (PPT) (PDF) | Mentioned | |
| W12 | 23.12 Thu. | YouTube Live | S8.1: Drammer: Deterministic Rowhammer Attacks on Mobile Platforms, CCS 2016. (PPT) (PDF) | Mentioned | |
| | | | S8.2: SquiggleFilter: An Accelerator for Portable Virus Detection, MICRO 2021 (PPT) (PDF) | Mentioned | |
| | | | S8.3: Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks, ASPLOS 2018. (PPT) (PDF) | Mentioned | |




## Spring 2022 Lectures/Schedule

| Week | Date | Livestream | Lecture | Readings | Assignments |
|---|---|---|---|---|---|
| W1 | 24.02 Thu. | YouTube Live | L1a: Course Logistics (PDF) (PPT) | Suggested | |
| | | | L1b: Introduction and Basics (PDF) (PPT) | Suggested | |
| | | | L1c: Architectural Design Fundamentals (PDF) (PPT) | Suggested | |
| W2 | 03.03 Thu. | YouTube Live | L2: Memory-Centric Computing (PDF) (PPT) | Suggested | |
| W3 | 10.03 Thu. | YouTube Live | L3: Memory-Centric Computing II (PDF) (PPT) | Suggested | |
| W4 | 17.03 Thu. | YouTube Live | L4: Memory-Centric Computing III (PDF) (PPT) | Suggested | |
| W5 | 24.03 Thu. | YouTube Live | L5: Accelerating Genome Analysis (PDF) (PPT) | Suggested | |
| W6 | 31.03 Thu. | YouTube Live | L6a: Rethinking Virtual Memory I (PDF) (PPT) | Suggested | |
| | | | L6b: Rethinking Virtual Memory II (PDF) (PPT) | Suggested | |
| W7 | 07.04 Thu. | YouTube Live | S1.1: A Logic-in-Memory Computer", IEEE Trans. Comput., 1970 (PDF) (PPT) | | |
| | | | S1.2: SISA: Set-Centric Instruction Set Architecture for Graph Mining on Processing-in-Memory Systems, MICRO 2021 (PDF) (PPT) | | |
| W8 | 14.04 Thu. | | S2.1: Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors, ISCA 2014. (PDF) (PPT) | | |
| | | | S2.2: Uncovering In-DRAM RowHammer Protection Mechanisms: A New Methodology, Custom RowHammer Patterns, and Implications, MICRO 2021 (PDF) (PPT) | | |
| W9 | 28.04 Thu. | YouTube Live | S3.1: ProSE: The Architecture and Design of a Protein Discovery Engine, ASPLOS 2022 (PDF) (PPT) | | |
| | | YouTube Premiere | S3.2: GenStore: a high-performance in-storage processing system for genome sequence analysis, ASPLOS 2022 (PDF) (PPT) | | |
| W10 | 05.05 Thu. | YouTube Live | S4.1: Focusing processor policies via critical-path prediction, ISCA 2001 (PDF) (PPT) | | |
| | | | S4.2: Pythia: A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning, MICRO 2021 (PDF) | | |
| W11 | 12.05 Thu. | YouTube Live | S5.1: Ten Lessons From Three Generations Shaped Google's TPUv4i, ISCA 2021 (PDF) (PPT) | | |
| | | | S5.2: Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks, PACT 2021] (PDF) | | |
| W12 | 19.05 Thu. | YouTube Live | S6.1: Hash, Don't Cache (the Page Table), SIGMETRICS 2016 (PDF) (PPT) | | |
| | | | S6.2: Processing-In-Memory Enabled Graphics Processors for 3D Rendering, HPCA 2017 (PDF) | | |
| W13 | 02.06 Thu. | YouTube Live | S7.1: QUAC-TRNG: High-Throughput True Random Number Generation Using Quadruple Row Activation in Commodity DRAM Chips, ISCA 2021 (PDF) (PPT) | | |
| | | | S7.2: A Deeper Look into RowHammer's Sensitivities: Experimental Analysis of Real DRAM Chips and Implications on Future Attacks and Defenses, MICRO 2021 (PDF) (PPT) | | |
| | | | S7.3: A2: Analog malicious hardware, IEEE Symposium on Security and Privacy 2016 (PDF) (PPT) | | |

# Research Opportunities

- If you are interested in doing research in Computer Architecture, Security, Systems & Bioinformatics:
  - Email me with your interest (CC: Mohammad, Juan)
  - Take the seminar course and the "Computer Architecture" course
  - Do readings and assignments on your own & **talk with us**

- There are many exciting projects and research positions, e.g.:
  - Novel memory/storage/computation/communication systems
  - New execution paradigms (e.g., in-memory computing)
  - Hardware security, safety, reliability, predictability
  - GPUs, TPUs, FPGAs, PIM, heterogeneous systems, …
  - Security-architecture-reliability-energy-performance interactions
  - Architectures for genomics/proteomics/medical/health/AI/ML
  - A limited list is here: https://safari.ethz.ch/theses/

**https://people.inf.ethz.ch/omutlu/projects.htm**

# For More Information: SAFARI Website

SAFARI

SAFARI Research Group

safari.ethz.ch

Think BIG, Aim HIGH!

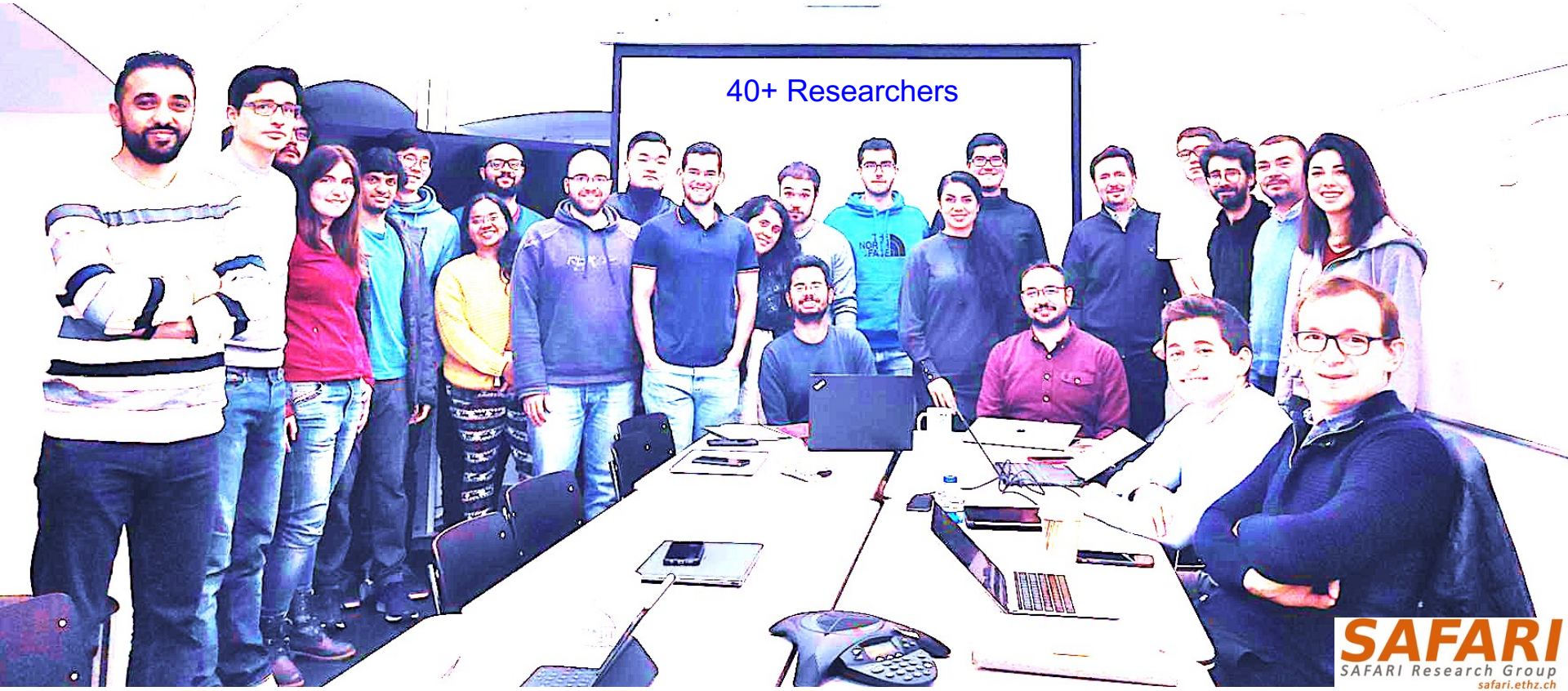
https://safari.ethz.ch

https://people.inf.ethz.ch/omutlu/projects.htm

# Onur Mutlu's SAFARI Research Group

***Computer architecture, HW/SW, systems, bioinformatics, security, memory***

https://safari.ethz.ch/safari-newsletter-january-2021/




Think BIG, Aim HIGH!

SAFARI

https://safari.ethz.ch

# SAFARI Newsletter December 2021 Edition

- https://safari.ethz.ch/safari-newsletter-december-2021/

SAFARI
SAFARI Research Group

*Think Big, Aim High*

ETH zürich

View in your browser
December 2021

# SAFARI Introduction & Research

***Computer architecture, HW/SW, systems, bioinformatics, security, memory***


SAFARI
SAFARI Research Group
safari.ethz.ch

Think BIG, Aim HIGH!

SAFARI

https://www.youtube.com/watch?v=mV2OuB2djEs

# SAFARI PhD and Post-Doc Alumni

- https://safari.ethz.ch/safari-alumni/

- Hasan Hassan (Rivos), EDAA Outstanding Dissertation Award 2023; S&P 2020 Best Paper Award, 2020 Pwnie Award, IEEE Micro TP HM 2020
- Christina Giannoula (Univ. of Toronto)
- Minesh Patel (ETH Zurich), DSN Carter Award for Best Thesis 2022; ETH Medal 2023; MICRO'20 & DSN'20 Best Paper Awards; ISCA HoF 2021
- Damla Senol Cali (Bionano Genomics), SRC TECHCON 2019 Best Student Presentation Award; RECOMB-Seq 2018 Best Poster Award
- Nastaran Hajinazar (Intel)
- Gagandeep Singh (AMD/Xilinx), FPL 2020 Best Paper Award Finalist
- Amirali Boroumand (Stanford Univ → Google), SRC TECHCON 2018 Best Presentation Award
- Jeremie Kim (Apple), EDAA Outstanding Dissertation Award 2020; IEEE Micro Top Picks 2019; ISCA/MICRO HoF 2021
- Nandita Vijaykumar (Univ. of Toronto, Assistant Professor), ISCA Hall of Fame 2021
- Kevin Hsieh (Microsoft Research, Senior Researcher)
- Justin Meza (Facebook), HiPEAC 2015 Best Student Presentation Award; ICCD 2012 Best Paper Award
- Mohammed Alser (ETH Zurich), IEEE Turkey Best PhD Thesis Award 2018
- Yixin Luo (Google), HPCA 2015 Best Paper Session
- Kevin Chang (Facebook), SRC TECHCON 2016 Best Student Presentation Award
- Rachata Ausavarungnirun (KMUNTB, Assistant Professor), NOCS 2015 and NOCS 2012 Best Paper Award Finalist
- Gennady Pekhimenko (Univ. of Toronto, Assistant Professor), ISCA Hall of Fame 2021; ASPLOS 2015 SRC Winner
- Vivek Seshadri (Microsoft Research)
- Donghyuk Lee (NVIDIA Research, Senior Researcher), HPCA Hall of Fame 2018
- Yoongu Kim (Software Robotics → Google), TCAD'19 Top Pick Award; IEEE Micro Top Picks'10; HPCA'10 Best Paper Session
- Lavanya Subramanian (Intel Labs → Facebook)

- Samira Khan (Univ. of Virginia, Assistant Professor), HPCA 2014 Best Paper Session
- Saugata Ghose (Univ. of Illinois, Assistant Professor), HPCA 2015 Best Paper Session, DFRWS-EU 2017 Best Paper Award
- Jawad Haj-Yahya (Huawei Research Zurich, Principal Researcher)
- Lois Orosa (Galicia Supercomputing Center, Director)
- Jisung Park (POSTECH, Assistant Professor)
- Gagandeep Singh (AMD/Xilinx, Researcher), FPL 2020 Best Paper Award Finalist

# You Can Join Us!

- **https://safari.ethz.ch/apply/**

**SAFARI Researcher Applications**

Sign in

This is the application submission site to be considered for being a researcher in the SAFARI Research Group, directed by Professor Onur Mutlu (Publications and Teaching).

If you are interested in doing research in the SAFARI Research Group, please make sure you apply through this submissions site and supply as many of the requested documents and information as possible. Please read and follow the provided instructions and submit as complete an application as possible (given the position you are applying for).

We suggest studying the following materials before submission:
SAFARI Publications and Courses
Onur Mutlu's Online Lectures and Course Materials

We strongly recommend that you read and analyze critically as many recent papers from our group as possible. This is the best way to prepare for the application process. Our recommendation is that you use professor Mutlu's methodology for critically analyzing papers.
Guide On Reviewing Papers

Good luck!

Welcome to the SAFARI at ETH Zurich -- PhD, Postdoc, Internship, Visiting Researcher Applications (SAFARI Researcher Applications) submissions site.

# Approaches to (Instruction-Level) Concurrency

- Pipelining
- Fine-Grained Multithreading
- Out-of-order Execution
- Dataflow (at the ISA level)
- Superscalar Execution
- VLIW
- Systolic Arrays
- Decoupled Access Execute
- SIMD Processing (Vector and array processors, GPUs)

# Readings for Today

- **Required**
  - H. T. Kung, "Why Systolic Architectures?," IEEE Computer 1982.

- **Recommended**
  - Jouppi et al., "In-Datacenter Performance Analysis of a Tensor Processing Unit", ISCA 2017.

# Readings for Next Week

- **Required**
  - Lindholm et al., "NVIDIA Tesla: A Unified Graphics and Computing Architecture," IEEE Micro 2008.

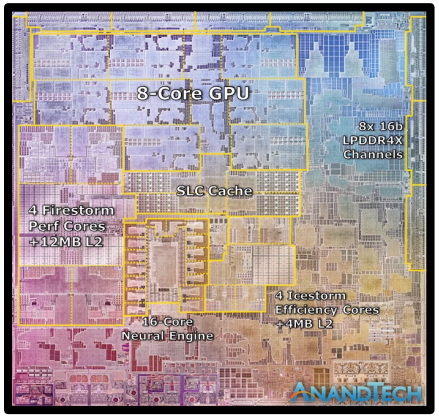
- **Recommended**
  - Peleg and Weiser, "MMX Technology Extension to the Intel Architecture," IEEE Micro 1996.

# Systolic Arrays

# General Purpose vs. Special Purpose Systems

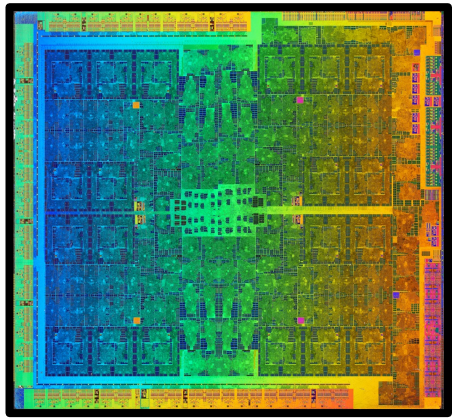
**General Purpose**

**CPUs**



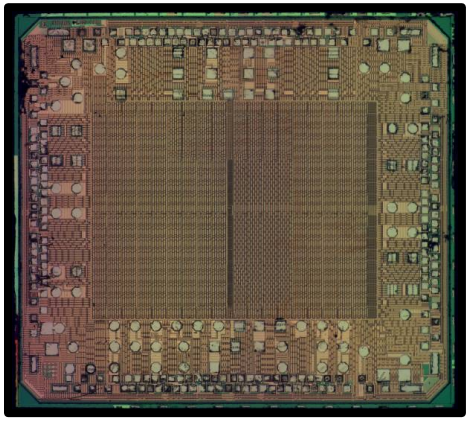

Apple M1

**GPUs**



Nvidia GTX 1070

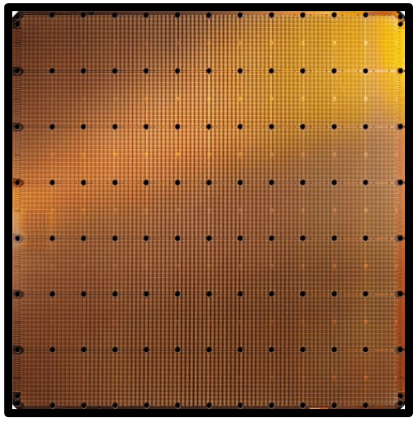
**FPGAs**



Xilinx Spartan

**Special Purpose**

**ASICs**



Cerebras WSE-2

**Flexible: Can execute any program**

**Easy to program & use**

**Not the best performance & efficiency**

**Efficient & High performance**

**(Usually) Difficult to program & use**

**Inflexible: Limited set of programs**

# Systolic Arrays: Specialized Accelerators

- Systolic arrays comprise an execution model
  - They implement systolic computation
  - Different from **von Neumann**, different from **dataflow**
- They are initially designed as **special-purpose accelerators**
  - For convolutions, filtering, pattern matching, special-purpose matrix/vector computations in image & vision processing, signal processing, pattern recognition, etc.
- They are currently used heavily in machine learning
  - Specialized machine learning accelerators
- Their general execution model can be (and is) generalized
  - As we will see later in this lecture

# Systolic Arrays: Motivation

- Goal: design an accelerator that has
  - Simple, regular design (keep # unique parts small and regular)
  - High concurrency → high performance
  - Balanced computation and I/O (memory) bandwidth

- Idea: Replace a single processing element (PE) with a regular array of PEs and carefully orchestrate flow of data between the PEs
  - such that they collectively transform a piece of input data before outputting it to memory

- Benefit: Maximizes computation done on a single piece of data element brought from memory

# Systolic Arrays

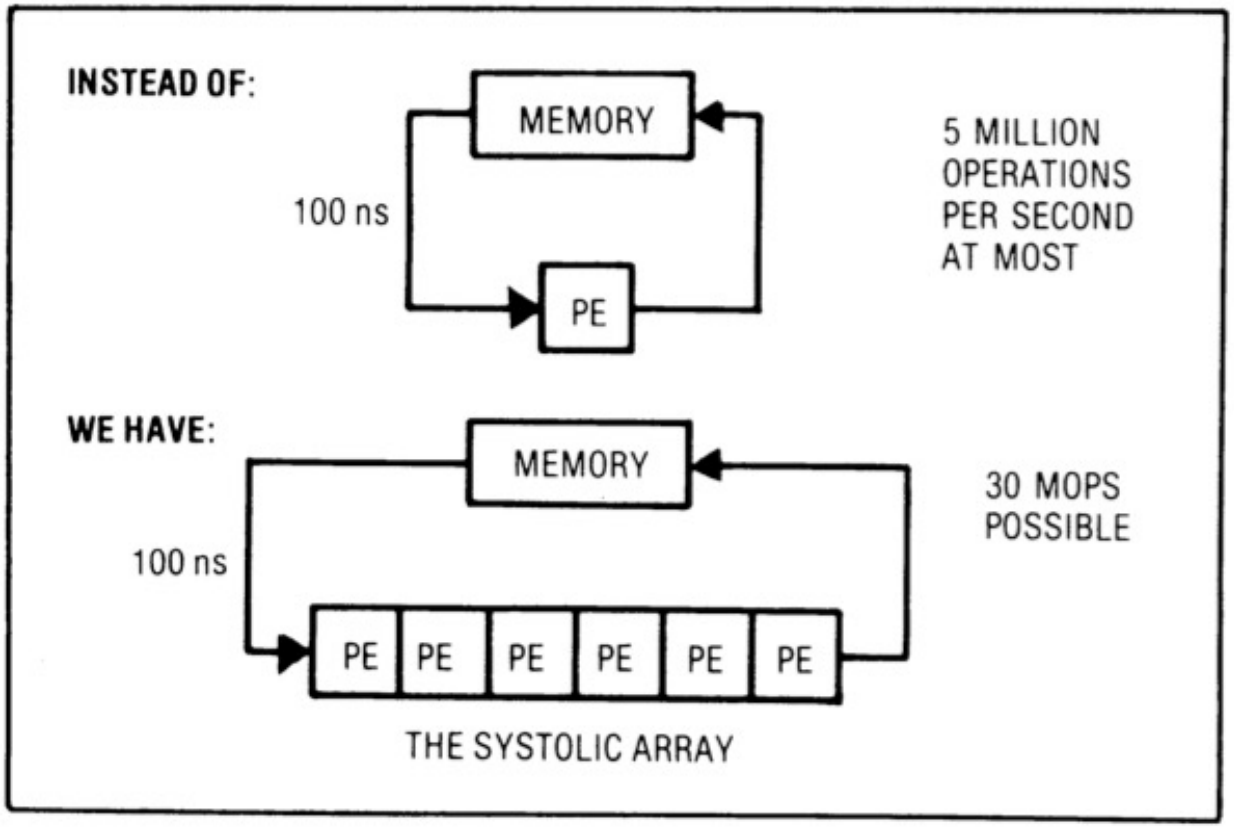



**Figure 1. Basic principle of a systolic system.**

Memory: heart
Data: blood
PEs: cells

Memory pulses data through PEs

- H. T. Kung, "Why Systolic Architectures?," IEEE Computer 1982.

# Why Systolic Architectures?

- Idea: Data flows from the computer memory in a rhythmic fashion, passing through many processing elements before it returns to memory

- Similar to blood flow: heart → many cells → heart
  - Different cells "process" the blood
  - Many veins operate simultaneously
  - Can be many-dimensional

- Why? Special purpose accelerators/architectures need
  - Simple, regular design (keep # unique parts small and regular)
  - High concurrency → high performance
  - Balanced computation and I/O (memory) bandwidth

# Systolic Architectures

- Basic principle: Replace a single PE with a regular array of PEs and carefully orchestrate flow of data between the PEs
  - Balance computation and memory bandwidth

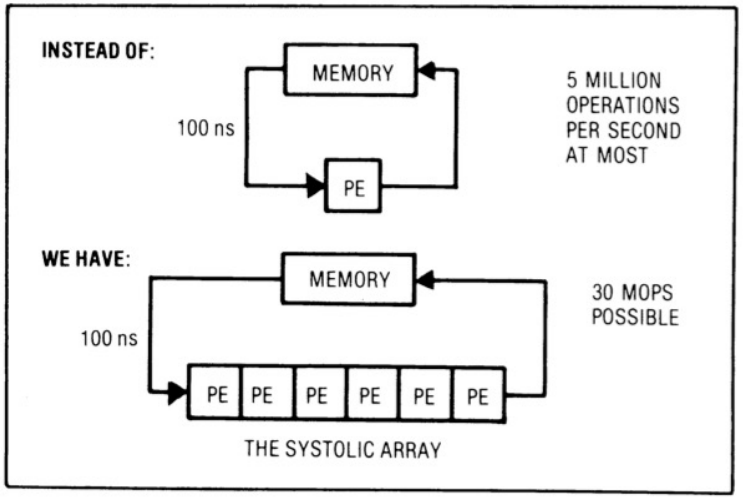



Figure 1. Basic principle of a systolic system.

- Differences from pipelining:
  - These are individual PEs
  - Array structure can be non-linear and multi-dimensional
  - PE connections can be multidirectional (and different speed)
  - PEs can have local memory and execute kernels (rather than a piece of the instruction)

# Systolic Computation Example

- Convolution
  - Used in filtering, pattern matching, correlation, polynomial evaluation, etc …
  - Many image processing tasks
  - Machine learning: up to hundreds of convolutional layers in Convolutional Neural Networks (CNN)

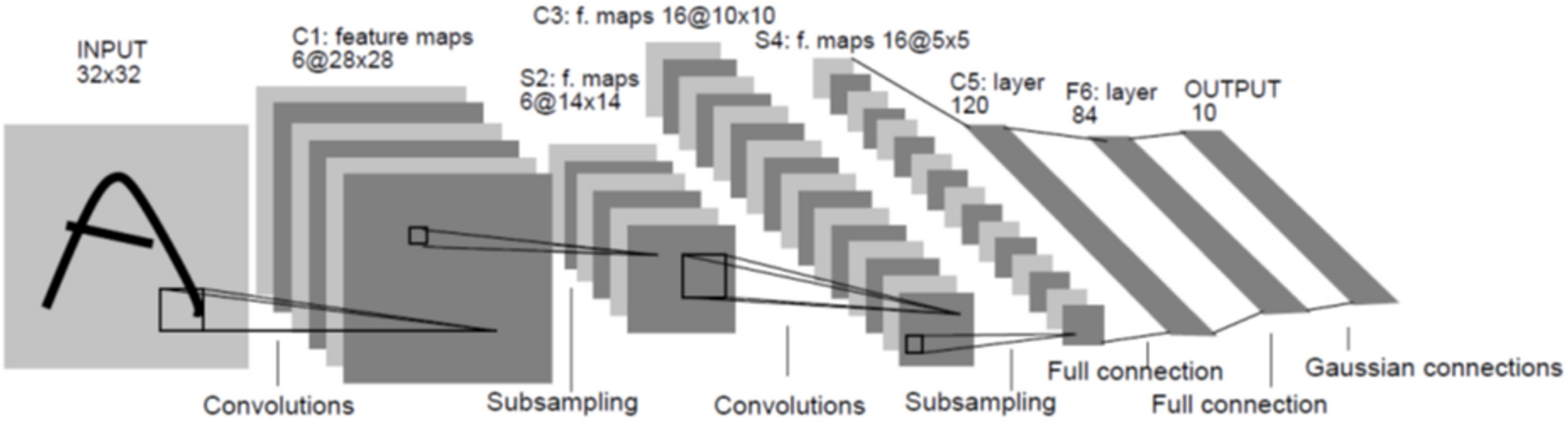
**Given** the sequence of weights $\{w_1, w_2, \ldots, w_k\}$
and the input sequence $\{x_1, x_2, \ldots, x_n\}$,

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

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

# LeNet-5, a Convolutional Neural Network for Hand-Written Digit Recognition

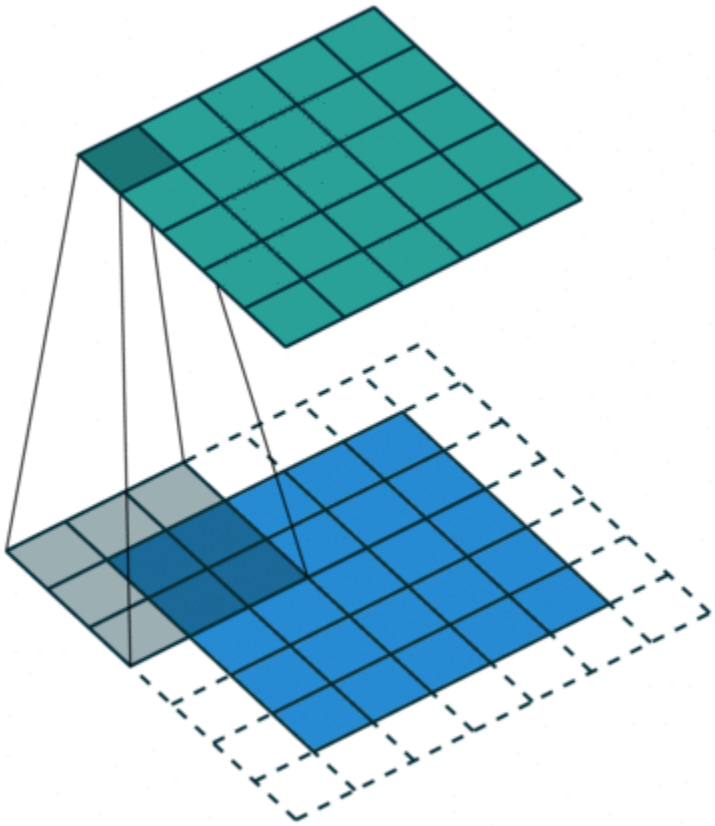
This is a 1024*8 bit input, which will have a truth table of $2^{8196}$ entries




Slide credit: Hwu & Kirk

# An Example of 2D Convolution

Output feature map



Input feature map

**Structure information**

Input: 5*5 (blue)

Kernel (filter): 3*3 (grey)

Output: 5*5 (green)

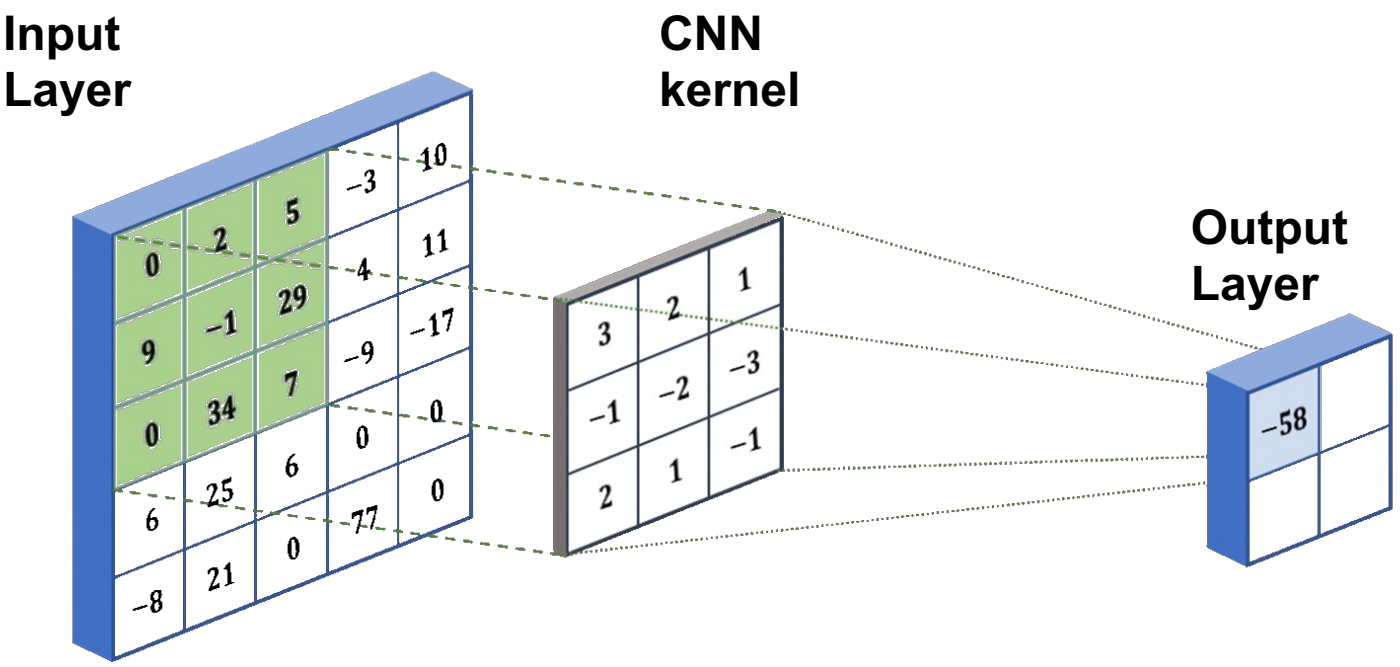
**Computation information**

Stride: 1

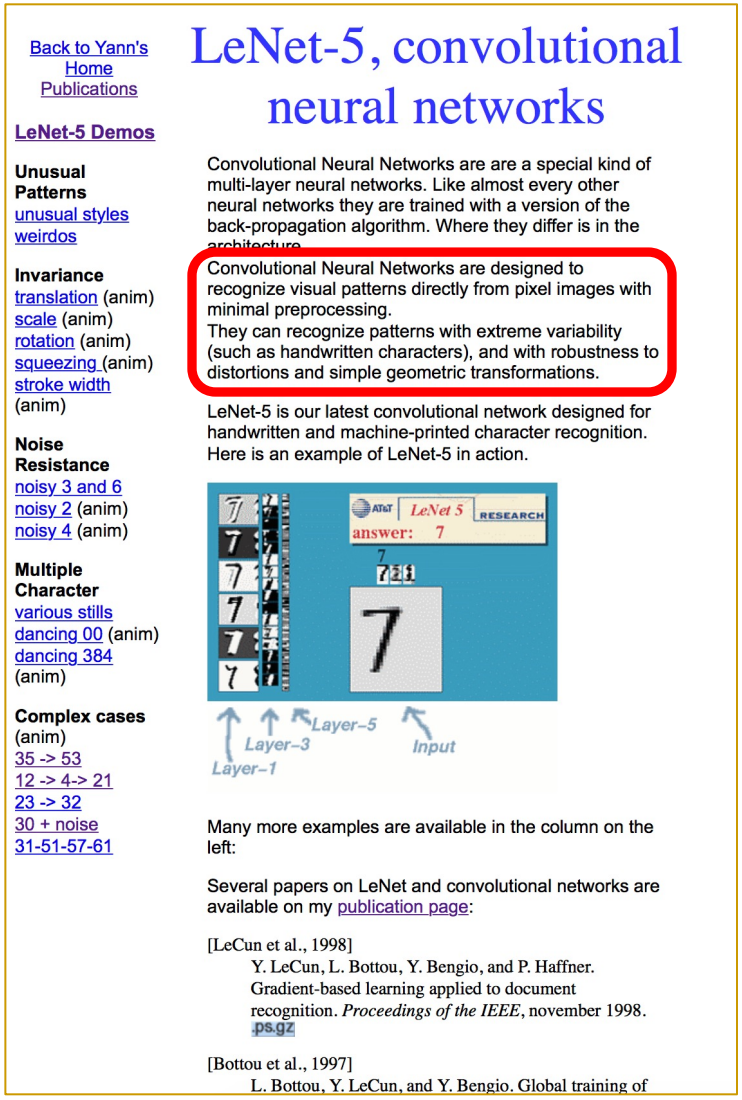
Padding: 1 (white)

Output Dim = (Input + 2*Padding - Kernel) / Stride + 1

# An Example of 2D Convolution



# Convolutional Neural Networks: Demo



Back to Yann's Home Publications

LeNet-5 Demos

**Unusual Patterns**
unusual styles
weirdos

**Invariance**
translation (anim)
scale (anim)
rotation (anim)
squeezing (anim)
stroke width (anim)

**Noise Resistance**
noisy 3 and 6
noisy 2 (anim)
noisy 4 (anim)

**Multiple Character**
various stills
dancing 00 (anim)
dancing 384 (anim)

**Complex cases** (anim)
35 -> 53
12 -> 4-> 21
23 -> 32
30 + noise
31-51-57-61

## LeNet-5, convolutional neural networks

Convolutional Neural Networks are are a special kind of multi-layer neural networks. Like almost every other neural networks they are trained with a version of the back-propagation algorithm. Where they differ is in the architecture.

Convolutional Neural Networks are designed to recognize visual patterns directly from pixel images with minimal preprocessing.
They can recognize patterns with extreme variability (such as handwritten characters), and with robustness to distortions and simple geometric transformations.

LeNet-5 is our latest convolutional network designed for handwritten and machine-printed character recognition. Here is an example of LeNet-5 in action.

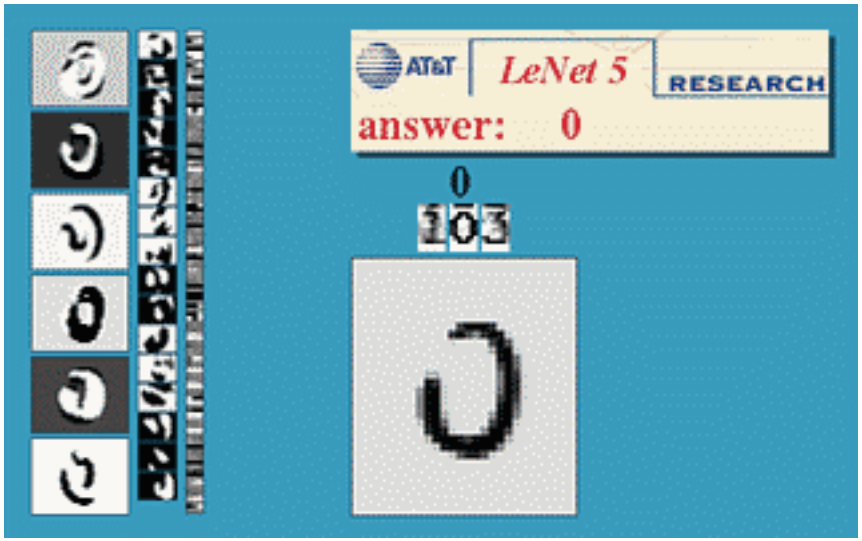



Many more examples are available in the column on the left:

Several papers on LeNet and convolutional networks are available on my publication page:

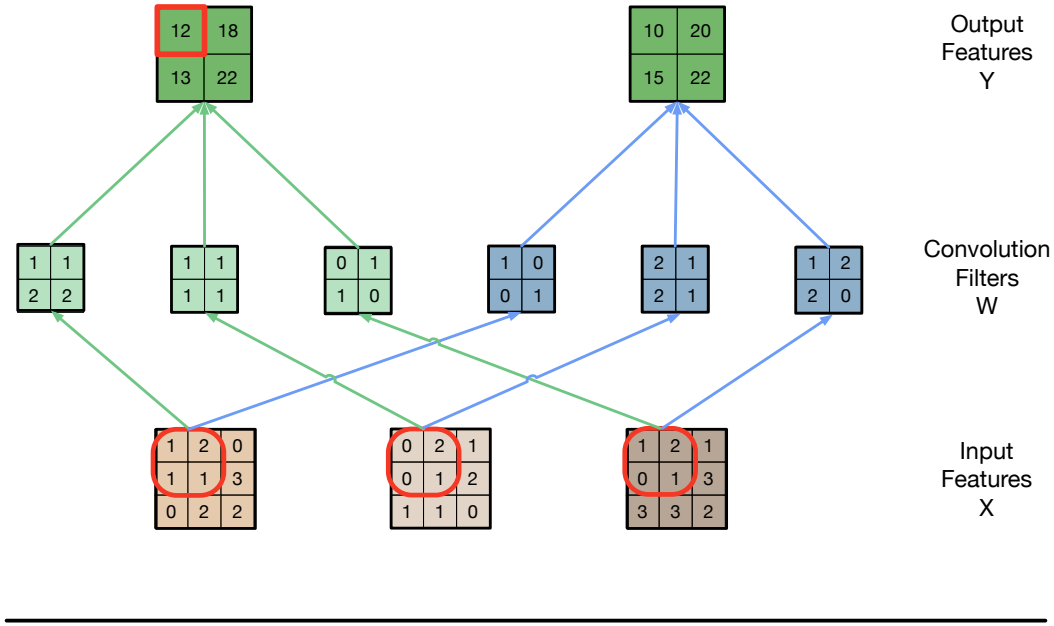
[LeCun et al., 1998]
Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, november 1998. ps.gz

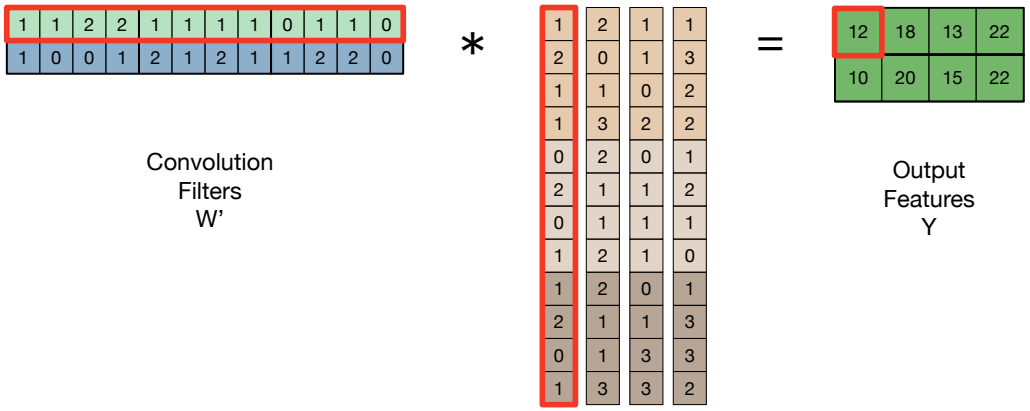
[Bottou et al., 1997]
L. Bottou, Y. LeCun, and Y. Bengio. Global training of




http://yann.lecun.com/exdb/lenet/index.html

# Implementing a Convolutional Layer with Matrix Multiplication



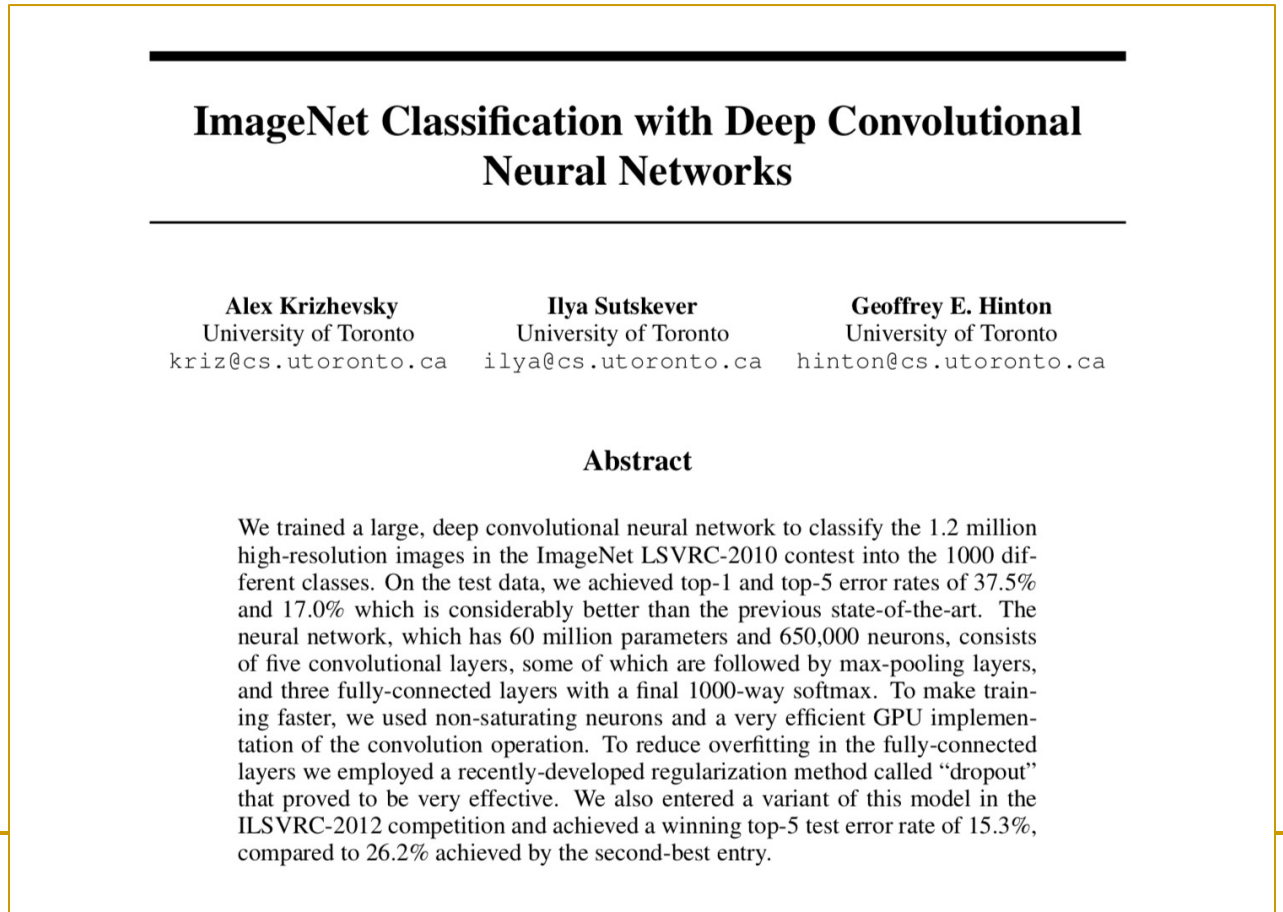
# Power of **Convolutions** and **Applied Courses**

- In 2010, Prof. Andreas Moshovos adopted Professor Hwu's ECE498AL Programming Massively Parallel Processors Class
- Several of Prof. Geoffrey Hinton's graduate students took the course
- These students developed the GPU implementation of the Deep CNN that was trained with 1.2M images to win the ImageNet competition

Slide credit: Hwu & Kirk

# Example: AlexNet (2012)

- AlexNet wins the ImageNet classification competition with ~10% points higher accuracy than state-of-the-art
  - Krizhevsky et al., "ImageNet Classification with Deep Convolutional Neural Networks", NIPS 2012.



**ImageNet Classification with Deep Convolutional Neural Networks**

**Alex Krizhevsky**
University of Toronto
kriz@cs.utoronto.ca

**Ilya Sutskever**
University of Toronto
ilya@cs.utoronto.ca

**Geoffrey E. Hinton**
University of Toronto
hinton@cs.utoronto.ca

**Abstract**

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called "dropout" that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

# Example: GoogLeNet (2014)

- Google improves accuracy by adding more network layers
  - From 8 in AlexNet to 22 in GoogLeNet
  - Szegedy et al., "Going Deeper with Convolutions", CVPR 2015.

**Going Deeper with Convolutions**

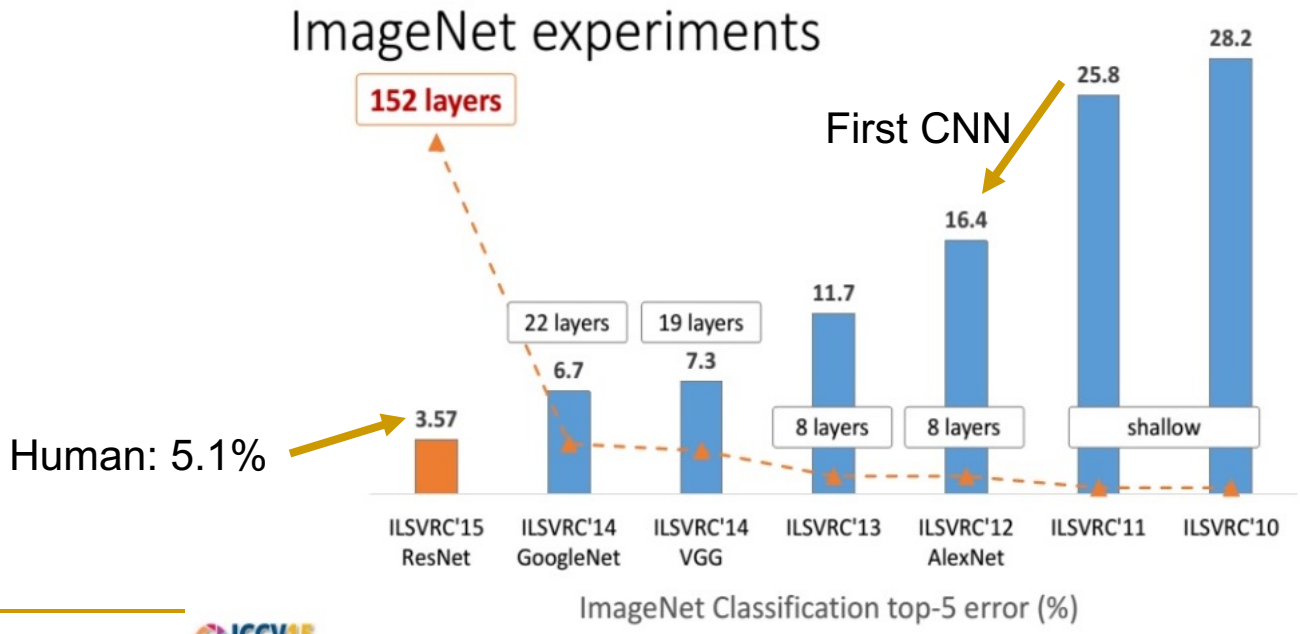
Christian Szegedy[1], Wei Liu[2], Yangqing Jia[1], Pierre Sermanet[1], Scott Reed[3],
Dragomir Anguelov[1], Dumitru Erhan[1], Vincent Vanhoucke[1], Andrew Rabinovich[4]
[1]Google Inc. [2]University of North Carolina, Chapel Hill
[3]University of Michigan, Ann Arbor [4]Magic Leap Inc.
[1]{szegedy,jiayq,sermanet,dragomir,dumitru,vanhoucke}@google.com
[2]wliu@cs.unc.edu, [3]reedscott@umich.edu, [4]arabinovich@magicleap.com

# Example: ResNet (2015)

- He et al., "Deep Residual Learning for Image Recognition", CVPR 2016.

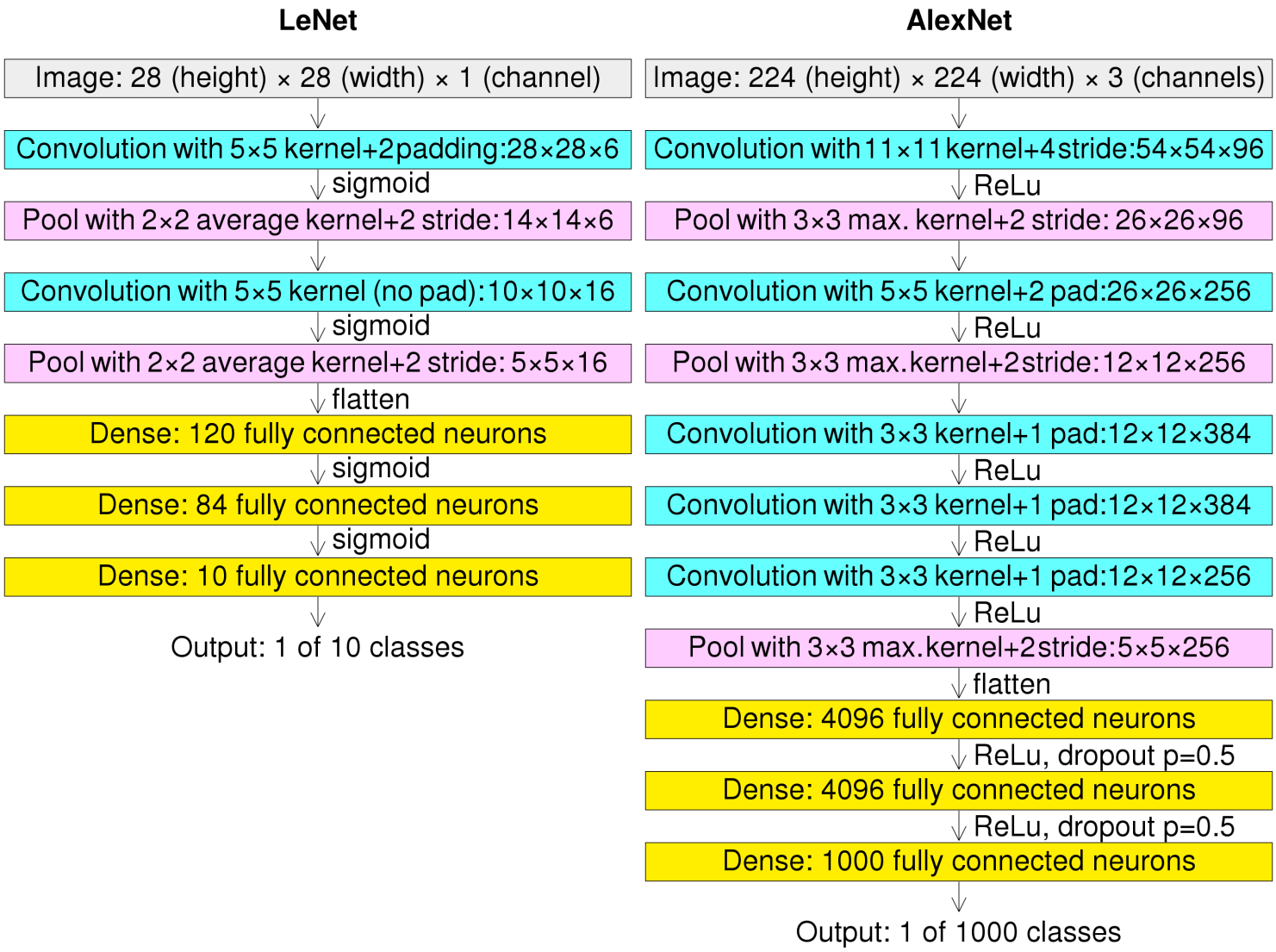
**Deep Residual Learning for Image Recognition**

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com




Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

# Neural Network Layer Examples



# Systolic Computation Example: Convolution (I)

- Convolution
  - Used in filtering, pattern matching, correlation, polynomial evaluation, etc …
  - Many image processing tasks
  - Machine learning: up to hundreds of convolutional layers in Convolutional Neural Networks (CNN)

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

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

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

# Systolic Computation Example: Convolution (II)

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

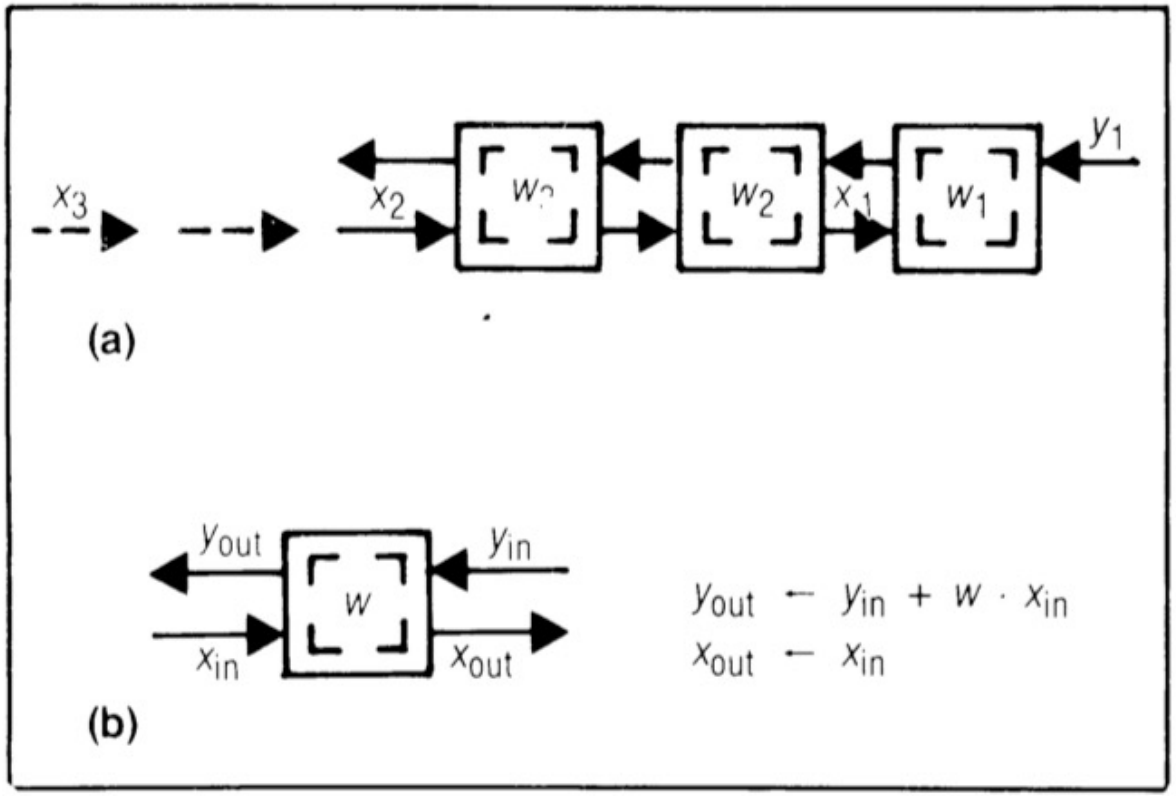



**Figure 8. Design W1: systolic convolution array (a) and cell (b) where $w_i$'s stay and $x_i$'s and $y_i$'s move systolically in opposite directions.**

# Systolic Computation Example: Convolution (III)

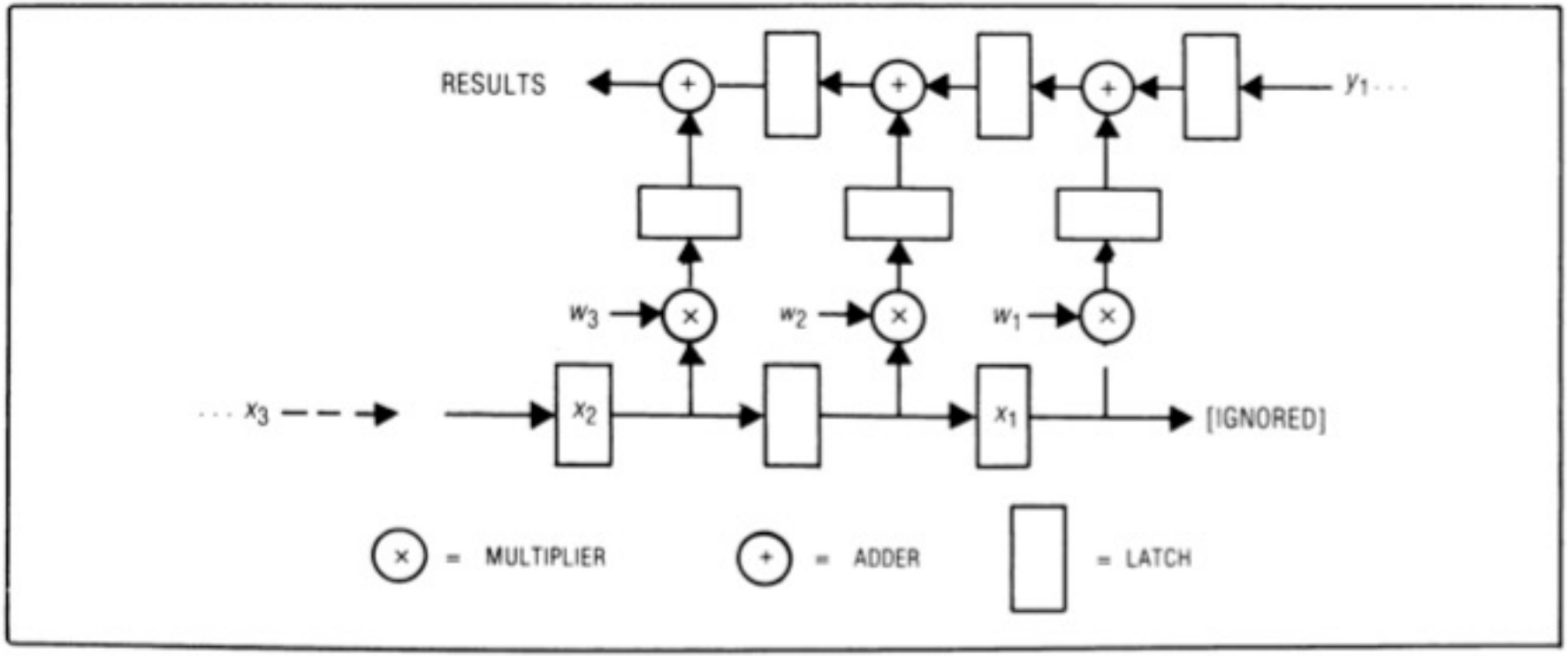



**Figure 10. Overlapping the executions of multiply and add in design W1.**

- Worthwhile to implement adder and multiplier separately to allow overlapping of add/mul executions

# Systolic Computation Example: Convolution (IV)

- One needs to carefully orchestrate when data elements are input to the array
- And when output is buffered

- This gets more involved when
  - Array dimensionality increases
  - PEs are less predictable in terms of latency

# Example 2D Systolic Array Computation

- Multiply two 3x3 matrices (inputs)
  - Keep the final result in PE accumulators

$$\begin{bmatrix} c_{00} & c_{01} & c_{02} \\ c_{10} & c_{11} & c_{12} \\ c_{20} & c_{21} & c_{22} \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} \times \begin{bmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \\ b_{20} & b_{21} & b_{22} \end{bmatrix}$$

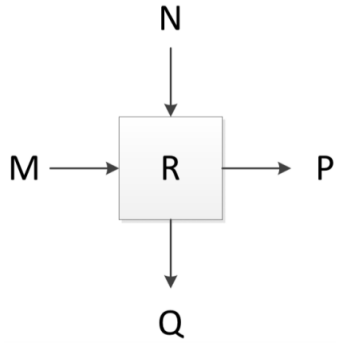
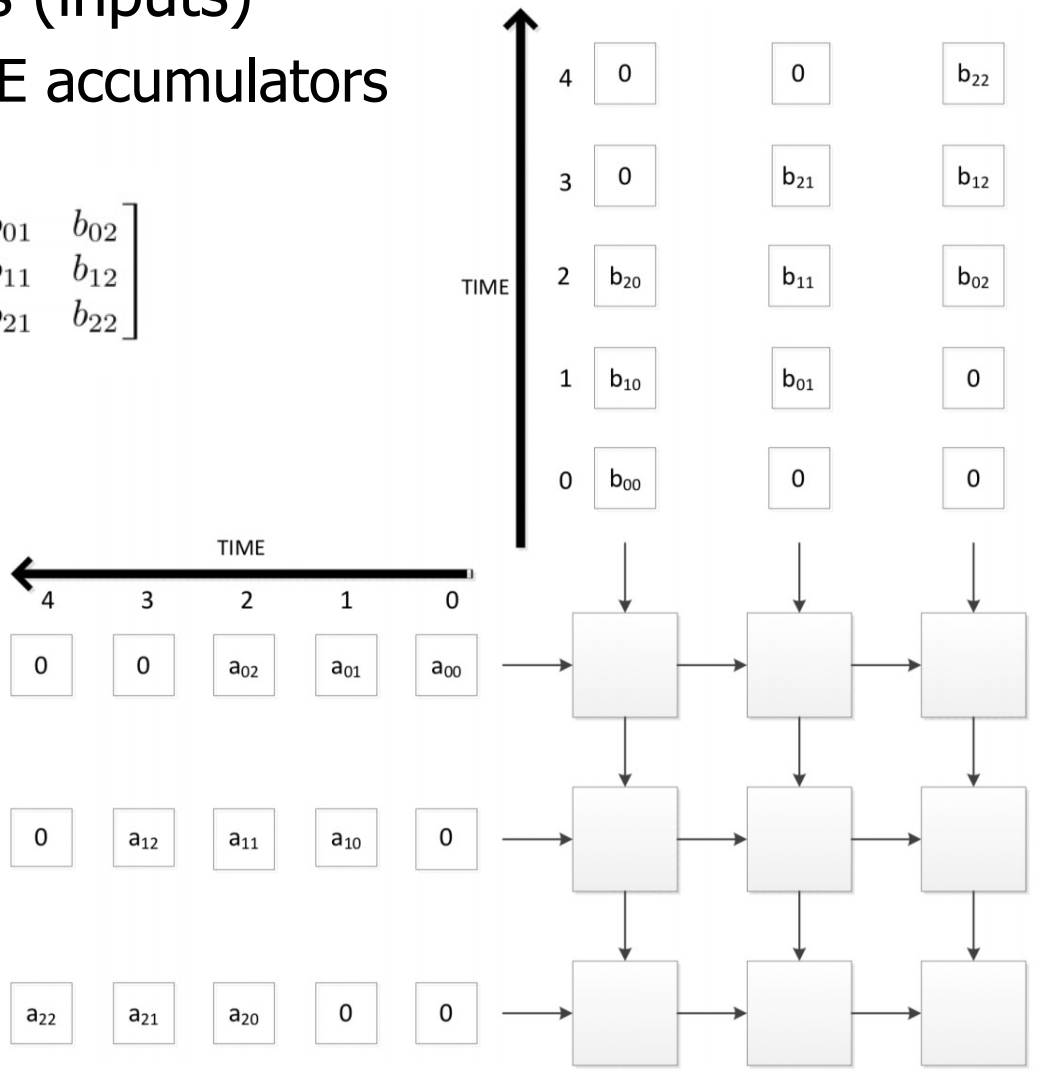



Figure 1: A systolic array processing element

P = M

Q = N

R = R + M*N



# Two-Dimensional Systolic Arrays

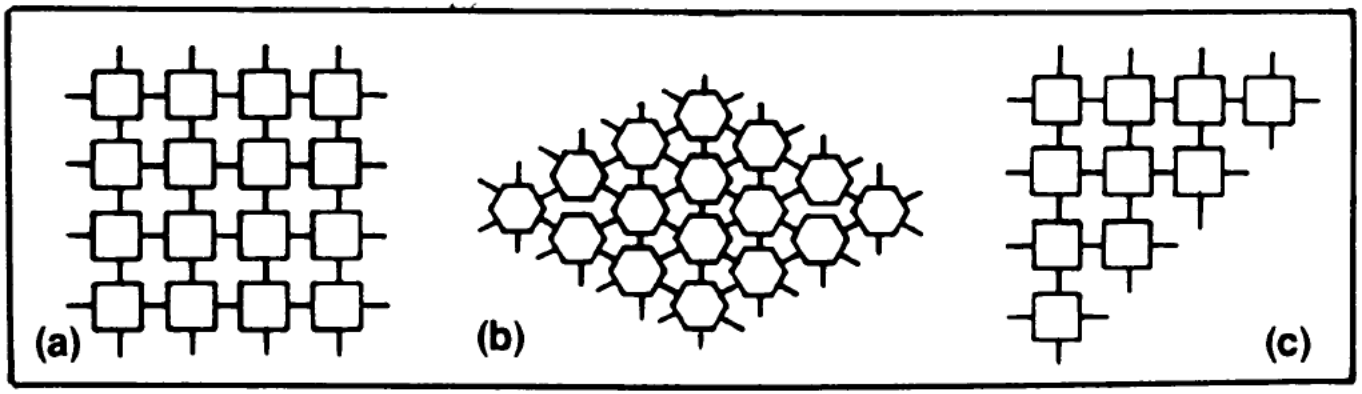



**Figure 11. Two-dimensional systolic arrays: (a) type R, (b) type H, and (c) type T.**

To a given problem there could be both one- and two-dimensional systolic array solutions. For example, two-dimensional convolution can be performed by a one-dimensional systolic array[24,25] or a two-dimensional systolic array.[6] When the memory speed is more than cell speed, two-dimensional systolic arrays such as those depicted in Figure 11 should be used. At each cell cycle, all the I/O ports on the array boundaries can input or output data items to or from the memory; as a result, the available memory bandwidth can be fully utilized. Thus, the choice of a one- or two-dimensional scheme is very dependent on how cells and memories will be implemented.

# Combinations

- Systolic arrays can be chained together to form powerful systems
- This systolic array is capable of producing on-the-fly least-squares fit to all the data that has arrived up to any given moment

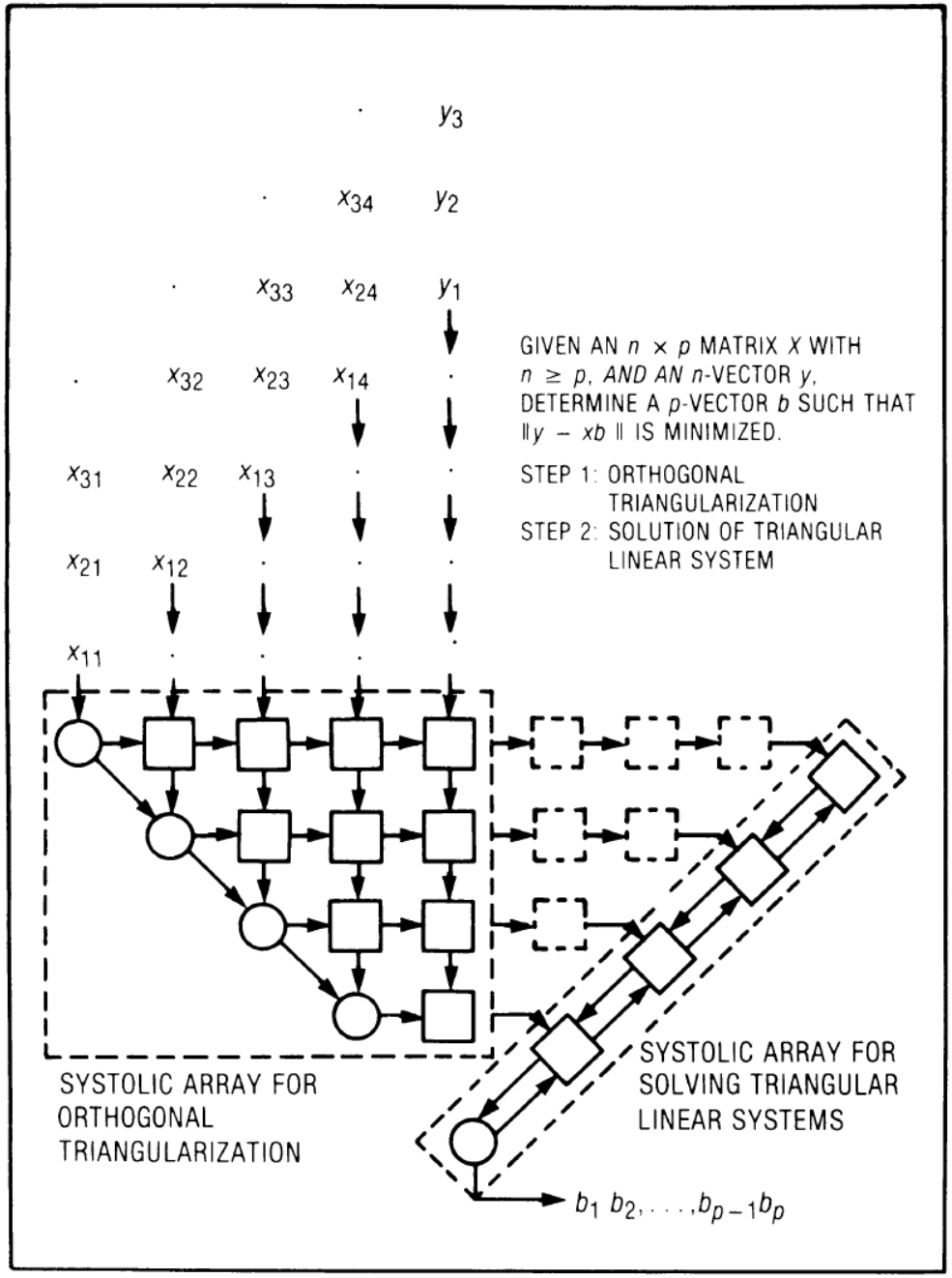



**Figure 12. On-the-fly least-squares solutions using one- and two-dimensional systolic arrays, with $p = 4$.**

# Systolic Arrays: Pros and Cons

- Advantages:
  - Principled: Efficiently makes use of limited memory bandwidth, balances computation to I/O bandwidth availability
  - Specialized (computation should fit the PE organization/functions)

    → improved efficiency, simple design, high concurrency/ performance

    → good to do more with less memory bandwidth requirement

- Downside:
  - Specialized

    → not generally applicable because computation needs to fit the PE functions/organization

# More Programmability in Systolic Arrays

- Each PE in a systolic array
  - Can store multiple "weights"
  - Weights can be selected on the fly
  - Eases implementation of, e.g., adaptive filtering

- Taken further
  - Each PE can have its own data and instruction memory
  - Data memory → to store partial/temporary results, constants
  - Leads to stream processing, pipeline parallelism
    - More generally, staged execution

# Pipeline-Parallel (Pipelined) Programs

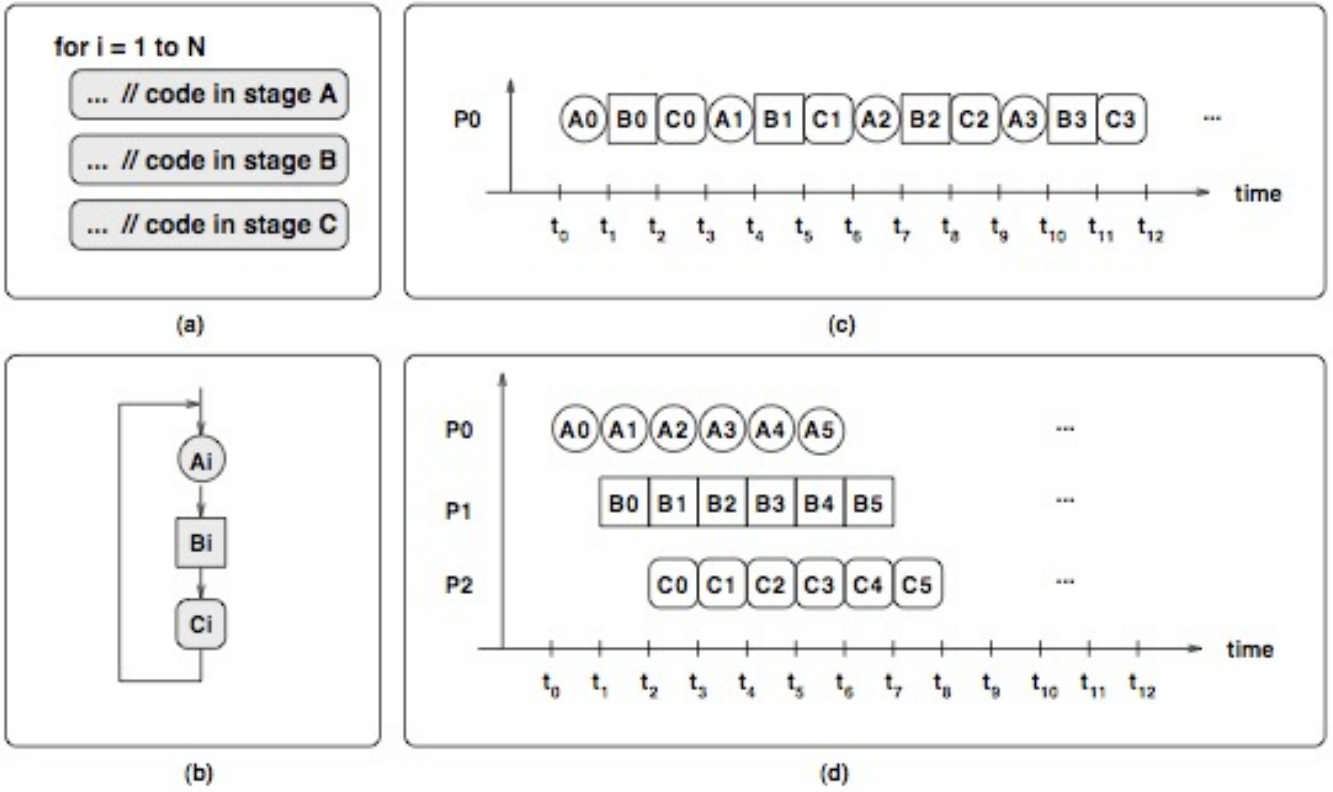



**Figure 1. (a) The code of a loop, (b) Each iteration is split into 3 pipeline stages: A, B, and C. Iteration i comprises Ai, Bi, Ci. (c) Sequential execution of 4 iterations. (d) Parallel execution of 6 iterations using pipeline parallelism on a three-core machine. Each stage executes on one core.**

Suleman+, "Data Marshaling for Multi-core Architectures," ISCA 2010.

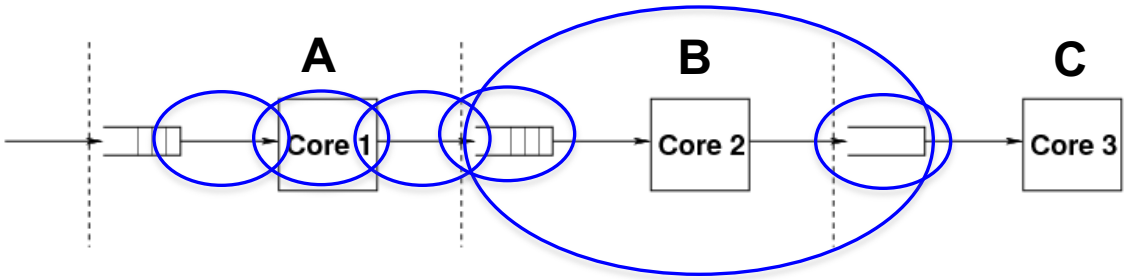
# Stages of Pipelined Programs

- Loop iterations are divided into code segments called **stages**
- Threads execute stages on different cores
  - Cores can be specialized for the computations that execute on them

```
loop {
  Compute1   A

  Compute2   B

  Compute3   C
}
```



# Pipelined File Compression Example

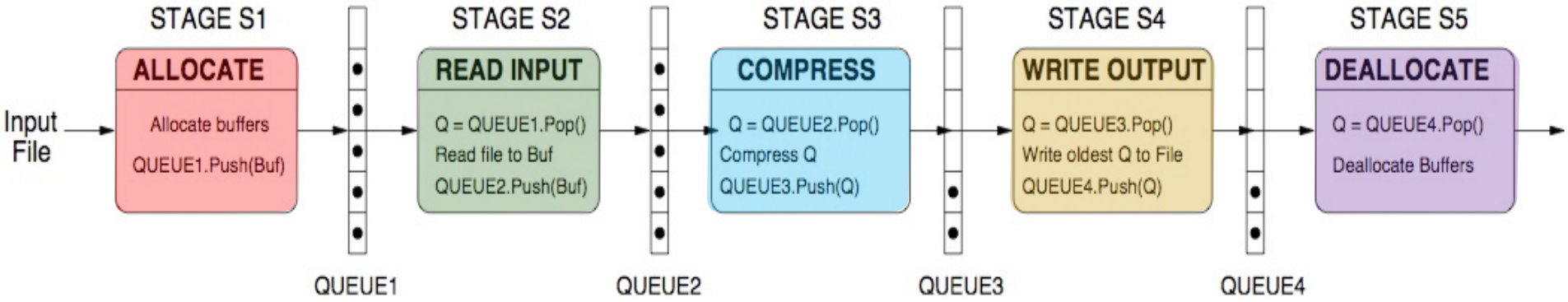



**Figure 3. File compression algorithm executed using pipeline parallelism**

# Systolic Array: Advantages & Disadvantages

- Advantages
  - Special purpose → high efficiency
  - Makes multiple uses of each data item → reduced need for fetching/refetching → better use of memory bandwidth
  - High concurrency
  - Regular design (both data and control flow) → easier to implement

- Disadvantages
  - Not good at exploiting irregular parallelism
  - Special purpose → not generally applicable
    - Needs software, programmer support to become more general purpose
  - Difficult to program if problem does not fit (well)

# Example Systolic Array: The WARP Computer

- HT Kung, CMU, 1984-1988

- Linear array of 10 cells, each cell a 10 Mflop programmable processor
- Attached to a general purpose host machine
- HLL and optimizing compiler to program the systolic array
- Used extensively to accelerate vision and robotics tasks

- Annaratone et al., "Warp Architecture and Implementation," ISCA 1986.
- Annaratone et al., "The Warp Computer: Architecture, Implementation, and Performance," IEEE TC 1987.

# The WARP Computer

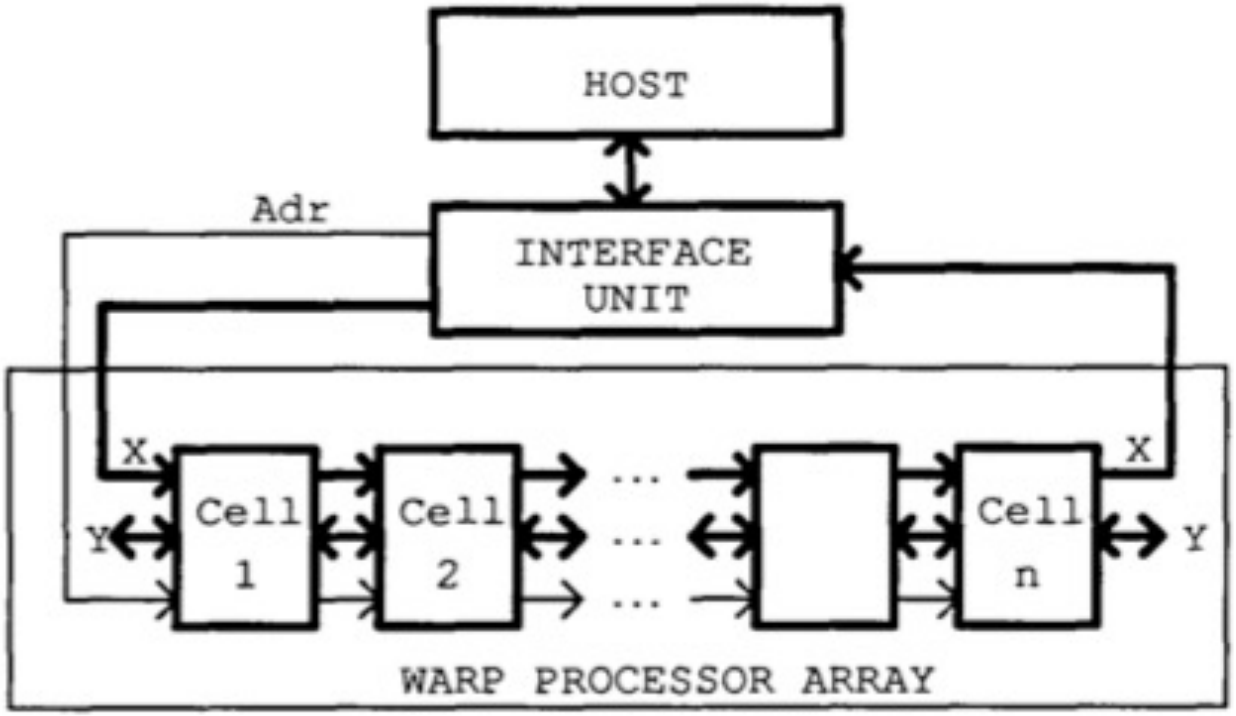



**Figure 1:** Warp system overview

# The WARP Cell

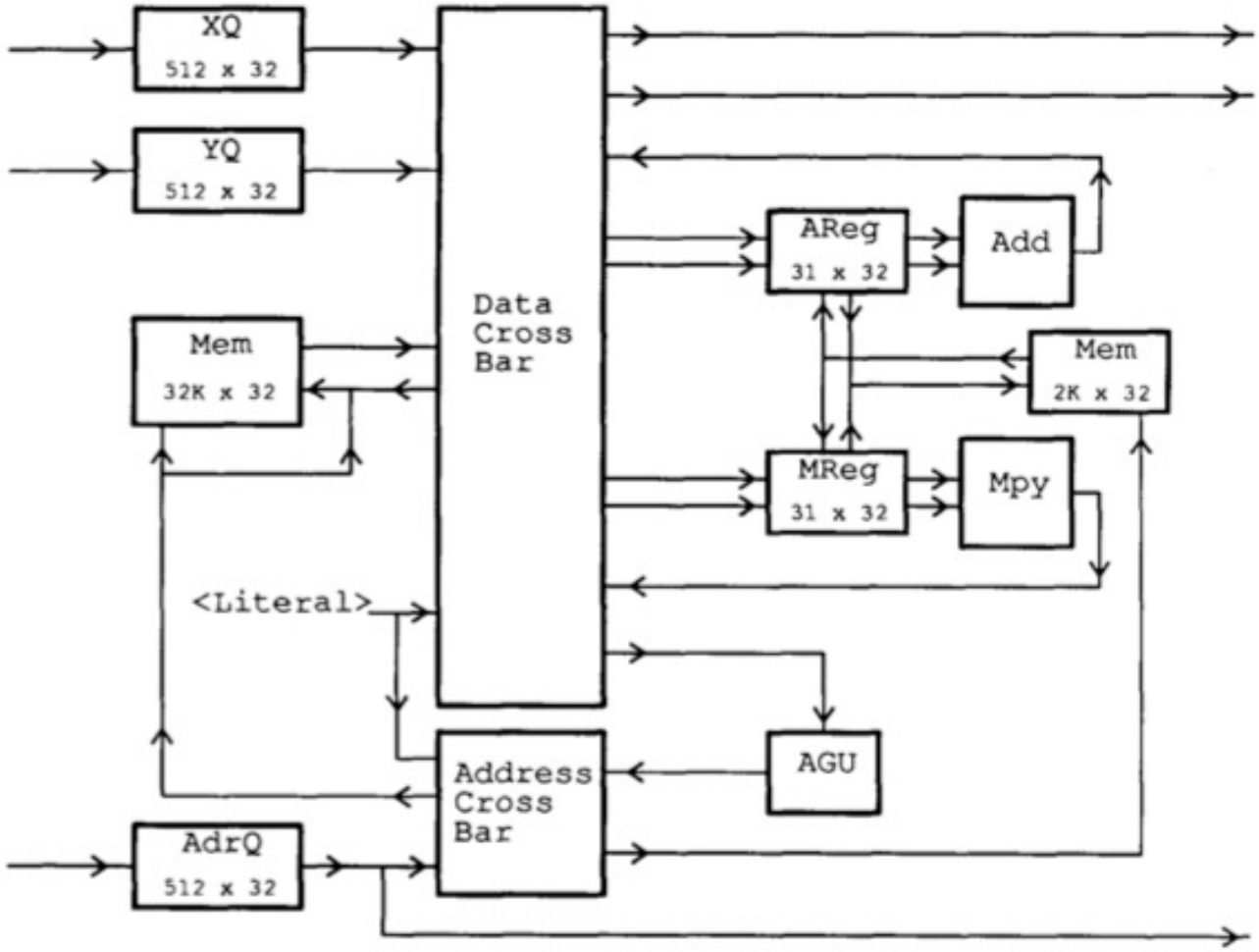



**Figure 2:** Warp cell data path

# An Example Modern Systolic Array: TPU (I)

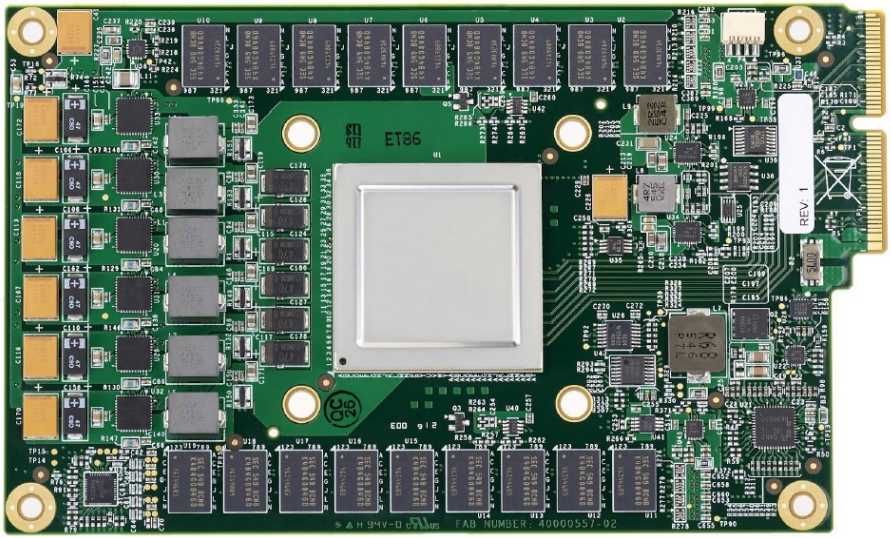


**Figure 3.** TPU Printed Circuit Board. It can be inserted in the slot for an SATA disk in a server, but the card uses PCIe Gen3 x16.

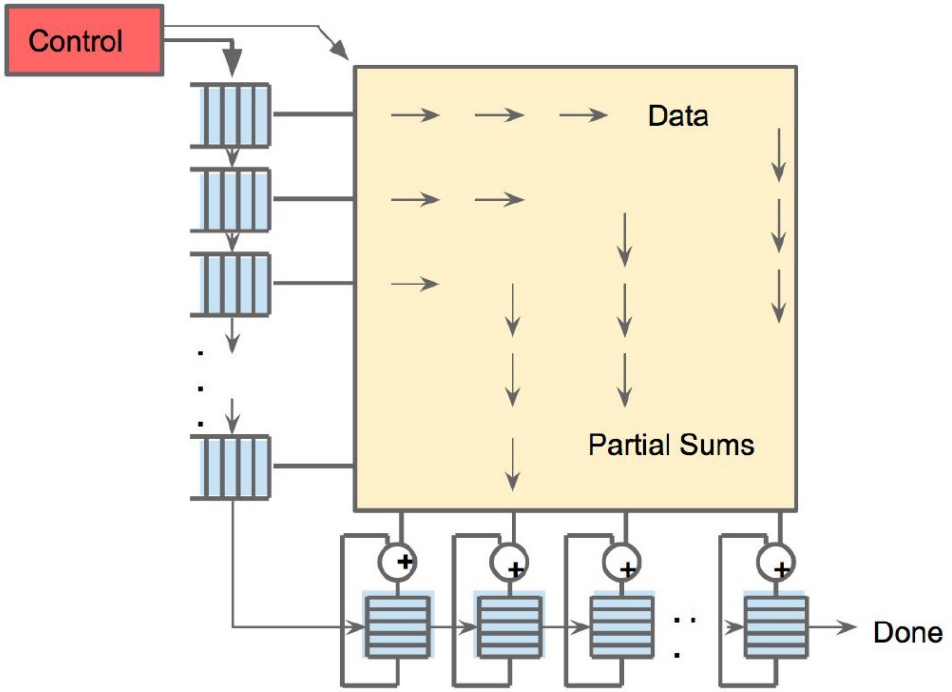
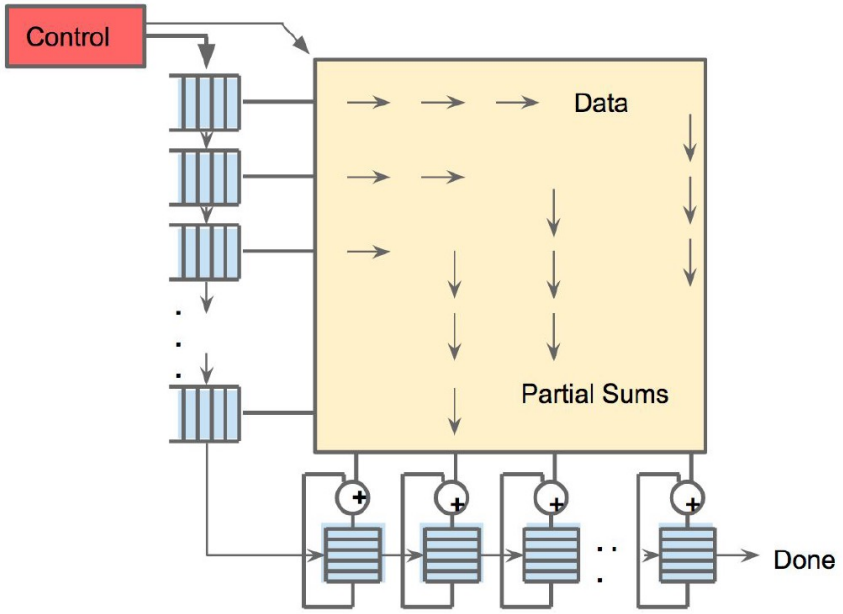



**Figure 4.** Systolic data flow of the Matrix Multiply Unit. Software has the illusion that each 256B input is read at once, and they instantly update one location of each of 256 accumulator RAMs.

Jouppi et al., "In-Datacenter Performance Analysis of a Tensor Processing Unit", ISCA 2017.

# An Example Modern Systolic Array: TPU (II)

As reading a large SRAM uses much more power than arithmetic, the matrix unit uses systolic execution to save energy by reducing reads and writes of the Unified Buffer [Kun80][Ram91][Ovt15b]. Figure 4 shows that data flows in from the left, and the weights are loaded from the top. A given 256-element multiply-accumulate operation moves through the matrix as a diagonal wavefront. The weights are preloaded, and take effect with the advancing wave alongside the first data of a new block. Control and data are pipelined to give the illusion that the 256 inputs are read at once, and that they instantly update one location of each of 256 accumulators. From a correctness perspective, software is unaware of the systolic nature of the matrix unit, but for performance, it does worry about the latency of the unit.




Jouppi et al., "In-Datacenter Performance Analysis of a Tensor Processing Unit", ISCA 2017.

# Recall: Example 2D Systolic Array Computation

- Multiply two 3x3 matrices (inputs)
  - Keep the final result in PE accumulators

$$\begin{bmatrix} c_{00} & c_{01} & c_{02} \\ c_{10} & c_{11} & c_{12} \\ c_{20} & c_{21} & c_{22} \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} \times \begin{bmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \\ b_{20} & b_{21} & b_{22} \end{bmatrix}$$

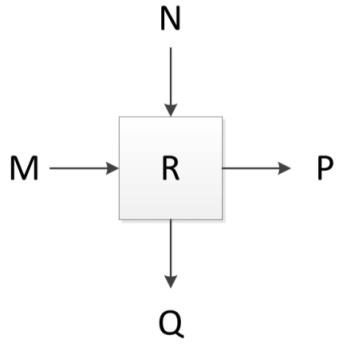
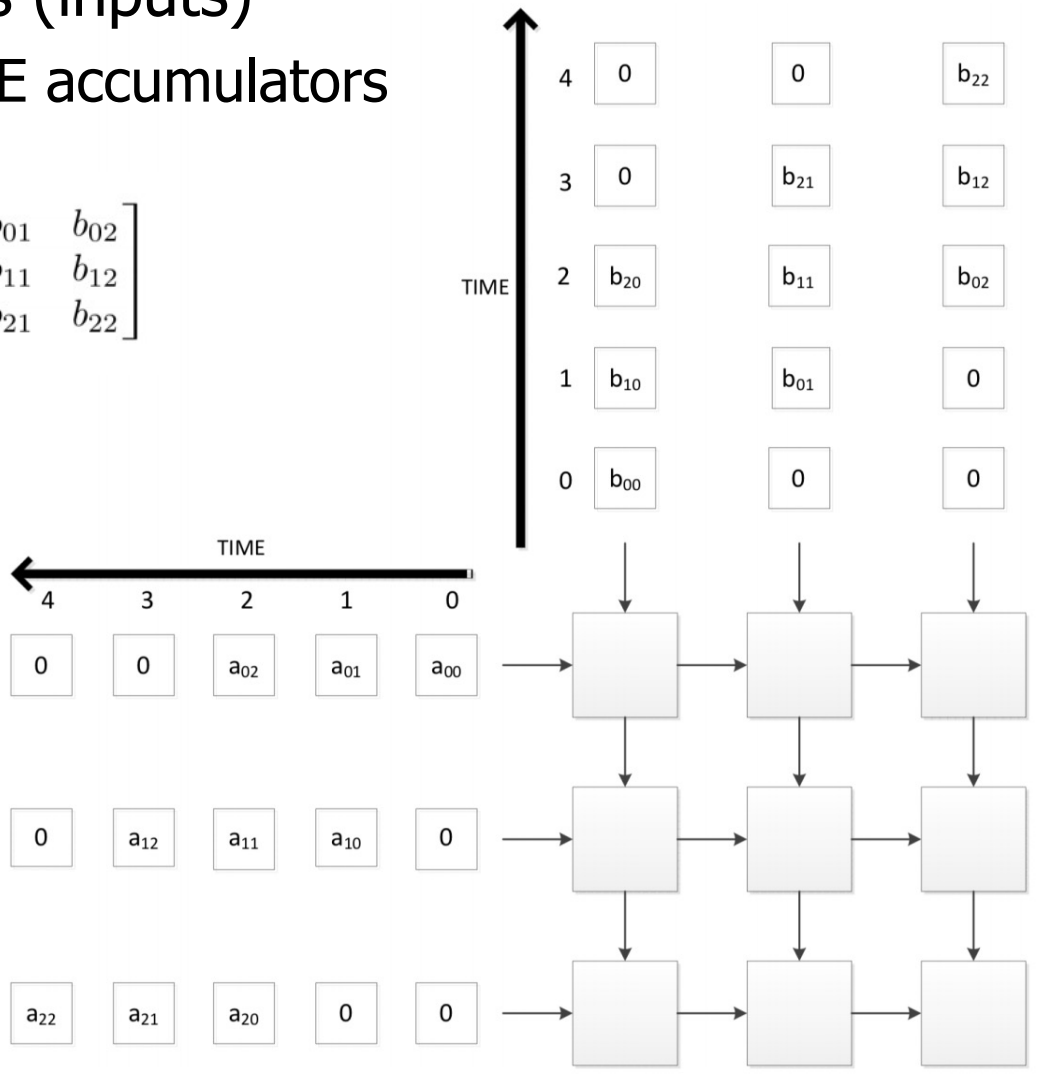



Figure 1: A systolic array processing element

P = M

Q = N

R = R + M*N



# An Example Modern Systolic Array: TPU (III)

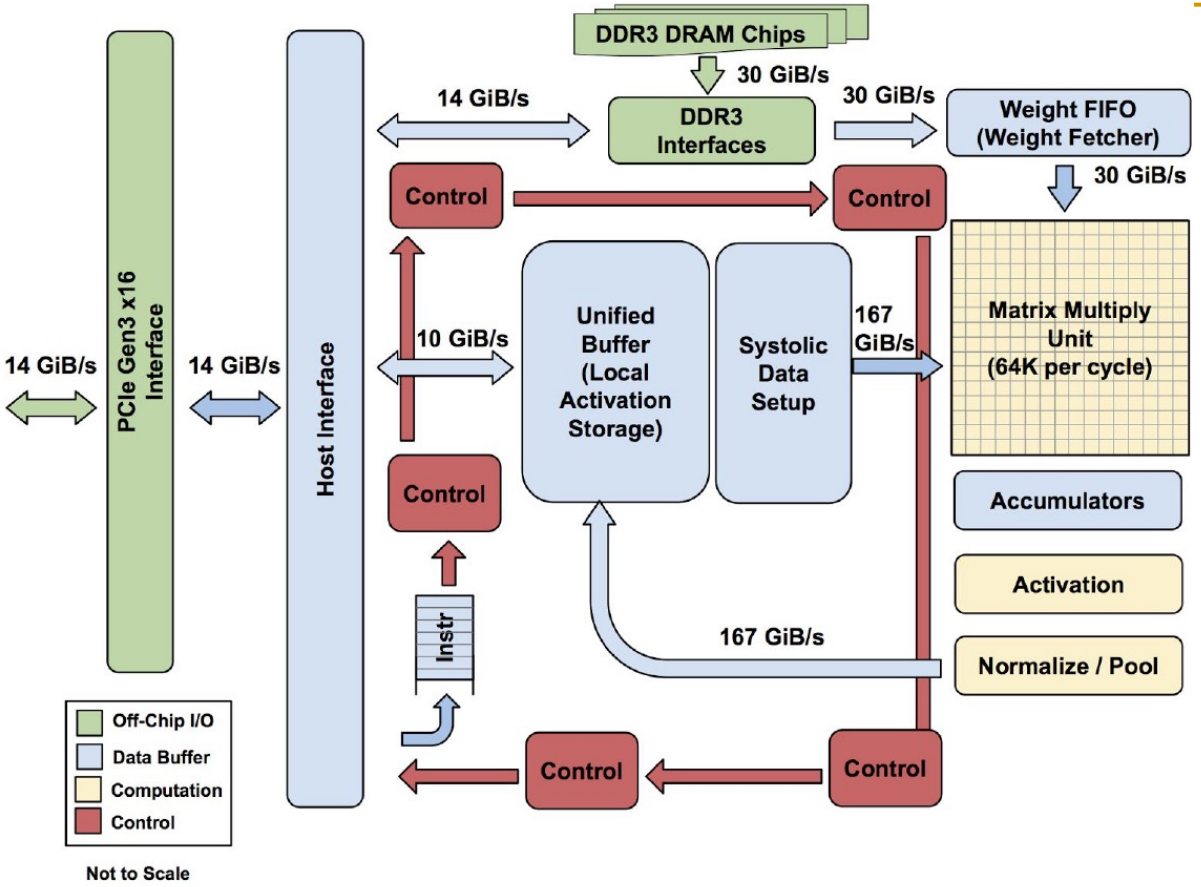
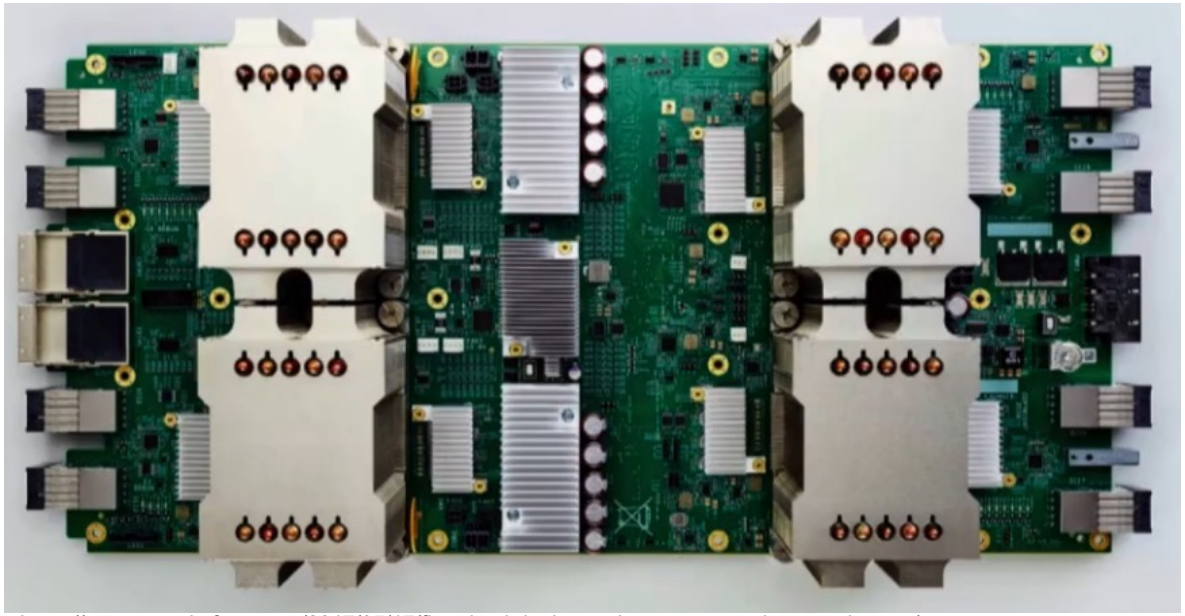



**Figure 1.** TPU Block Diagram. The main computation part is the yellow Matrix Multiply unit in the upper right hand corner. Its inputs are the blue Weight FIFO and the blue Unified Buffer (UB) and its output is the blue Accumulators (Acc). The yellow Activation Unit performs the nonlinear functions on the Acc, which go to the UB.

# An Example Modern Systolic Array: TPU2



https://www.nextplatform.com/2017/05/17/first-depth-look-googles-new-second-generation-tpu/

4 TPU chips
vs 1 chip in TPU1

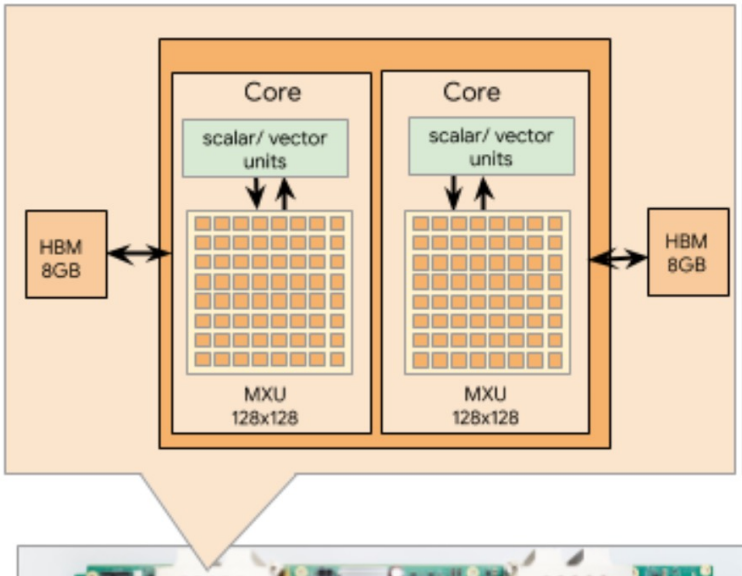
High Bandwidth Memory
vs DDR3

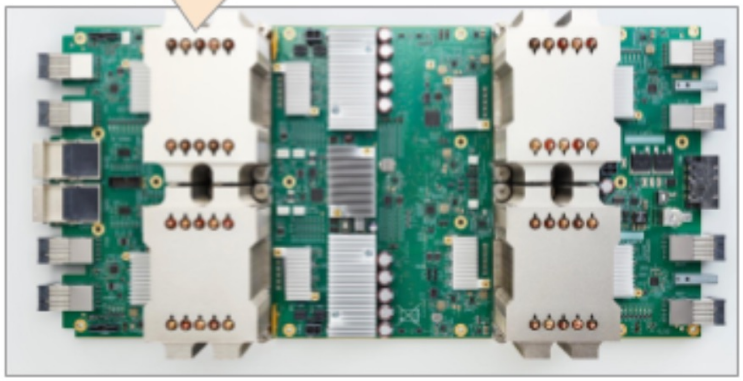
Floating point operations
vs FP16

45 TFLOPS per chip
vs 23 TOPS

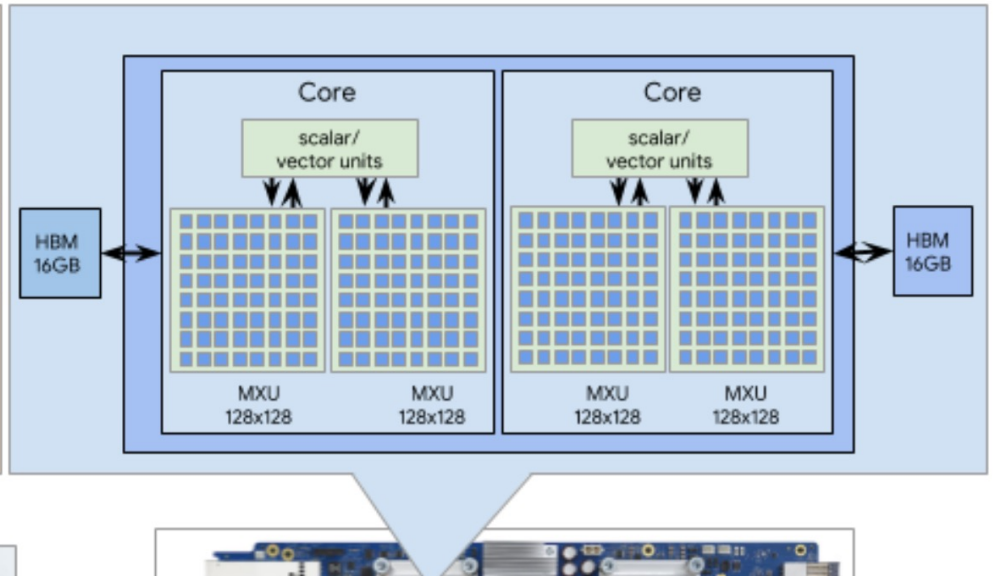
Designed for training
and inference
vs only inference

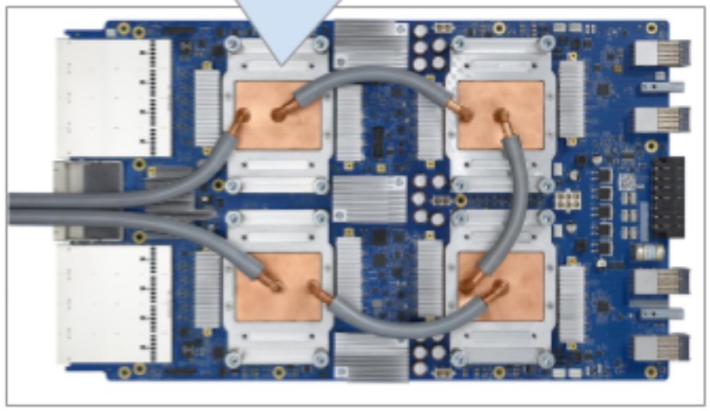
# An Example Modern Systolic Array: TPU3



TPU v2 - 4 chips, 2 cores per chip



TPU v3 - 4 chips, 2 cores per chip

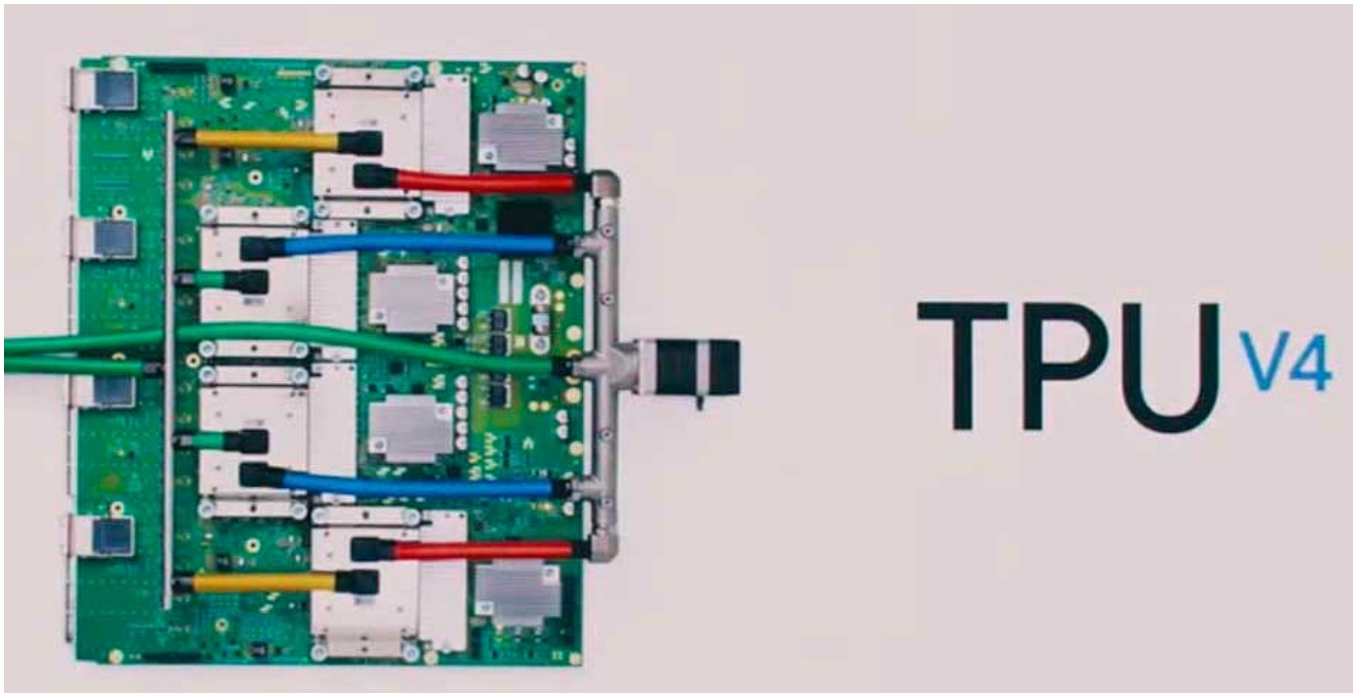
32GB HBM per chip
vs 16GB HBM in TPU2

4 Matrix Units per chip
vs 2 Matrix Units in TPU2

90 TFLOPS per chip
vs 45 TFLOPS in TPU2

https://cloud.google.com/tpu/docs/system-architecture

# An Example Modern Systolic Array: TPU4




New ML applications (vs. TPU3):

- Computer vision
- Natural Language Processing (NLP)
- Recommender system
- Reinforcement learning that plays Go

250 TFLOPS per chip in 2021
vs 90 TFLOPS in TPU3

1 ExaFLOPS per board

https://spectrum.ieee.org/tech-talk/computing/hardware/heres-how-googles-tpu-v4-ai-chip-stacked-up-in-training-tests

# A Summary Reading of Google TPUs

## Ten Lessons From Three Generations Shaped Google's TPUv4i

**Industrial Product**

Norman P. Jouppi, Doe Hyun Yoon, Matthew Ashcraft, Mark Gottscho, Thomas B. Jablin, George Kurian, James Laudon, Sheng Li, Peter Ma, Xiaoyu Ma, Thomas Norrie, Nishant Patil, Sushma Prasad, Cliff Young, Zongwei Zhou, and David Patterson, Google LLC

https://safari.ethz.ch/architecture_seminar/spring2022/lib/exe/fetch.php?media=ten_lessons_from_three_generations_shaped_google_s_tpuv4i.pdf

# A Recent Reading on the TPUv4

**TPU v4: An Optically Reconfigurable Supercomputer for Machine Learning with Hardware Support for Embeddings**

**Industrial Product***

Norman P. Jouppi, George Kurian, Sheng Li, Peter Ma, Rahul Nagarajan, Lifeng Nai, Nishant Patil, Suvinay Subramanian, Andy Swing, Brian Towles, Cliff Young, Xiang Zhou, Zongwei Zhou, and David Patterson
Google, Mountain View, CA

https://arxiv.org/pdf/2304.01433.pdf

# An Analysis of the Google Edge TPU

- Amirali Boroumand, Saugata Ghose, Berkin Akin, Ravi Narayanaswami, Geraldo F. Oliveira, Xiaoyu Ma, Eric Shiu, and Onur Mutlu,
  **"Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks"**
  *Proceedings of the 30th International Conference on Parallel Architectures and Compilation Techniques (**PACT**)*, Virtual, September 2021.
  [Slides (pptx) (pdf)]
  [Talk Video (14 minutes)]

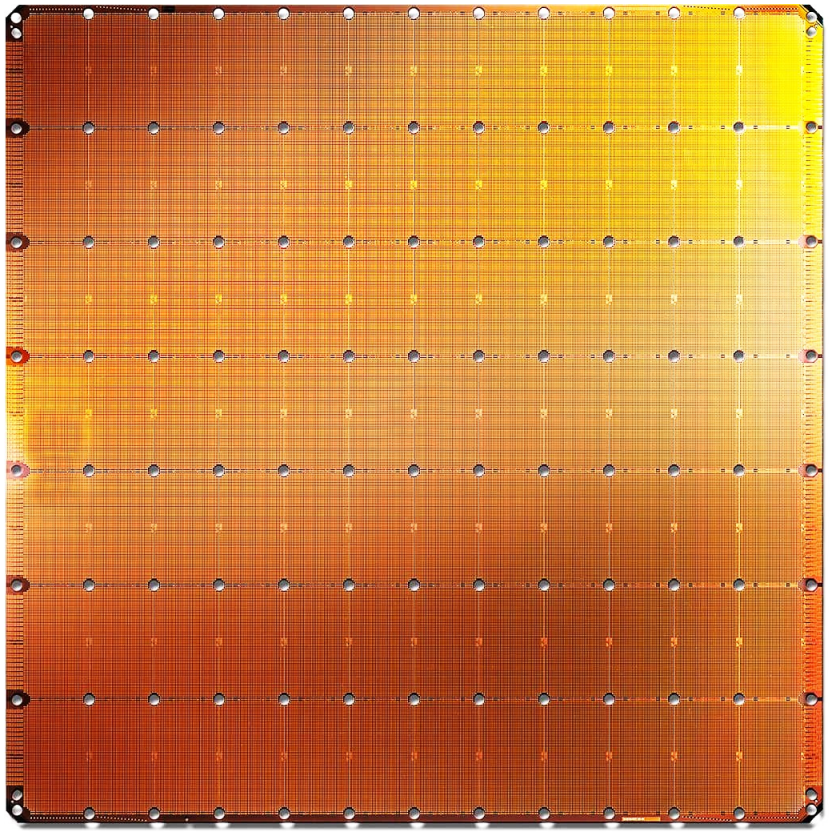
**> 90% of the total system energy is spent on memory in large ML models**

**Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks**

Amirali Boroumand[†◇] Saugata Ghose[‡] Berkin Akin[§] Ravi Narayanaswami[§]
Geraldo F. Oliveira[⋆] Xiaoyu Ma[§] Eric Shiu[§] Onur Mutlu[⋆†]

[†]*Carnegie Mellon Univ.* [◇]*Stanford Univ.* [‡]*Univ. of Illinois Urbana-Champaign* [§]*Google* [⋆]*ETH Zürich*

# Cerebras's Wafer Scale Engine (2019)



- The largest ML accelerator chip
- 400,000 cores

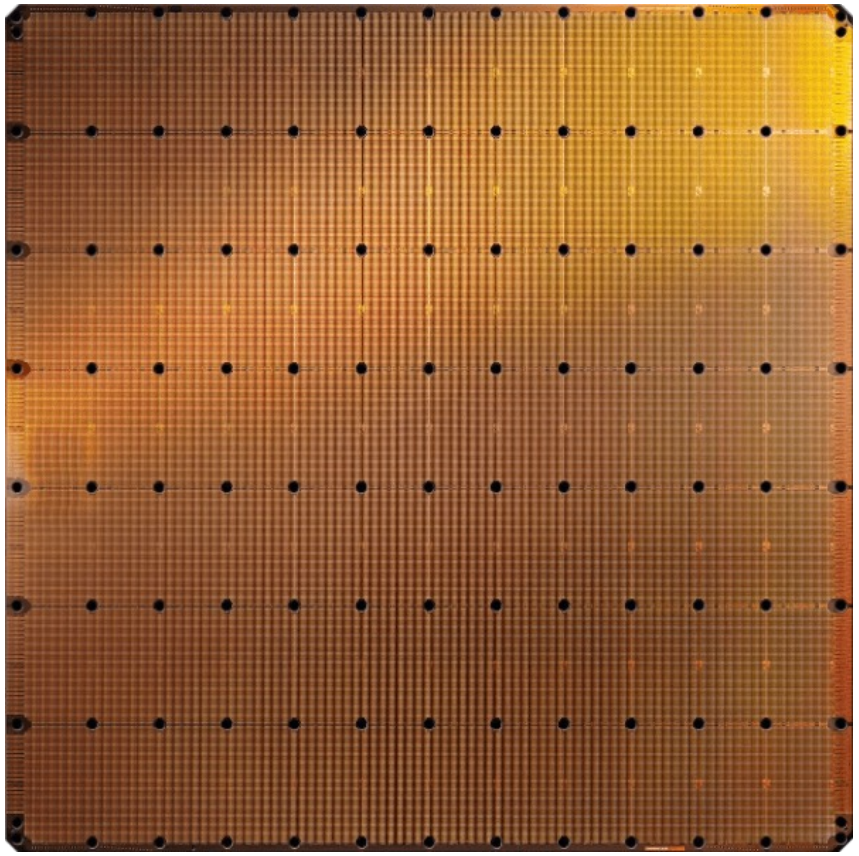

**Cerebras WSE**
1.2 Trillion transistors
46,225 mm$^2$

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

https://www.anandtech.com/show/14758/hot-chips-31-live-blogs-cerebras-wafer-scale-deep-learning

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

# Cerebras's Wafer Scale Engine-2 (2021)



- The largest ML accelerator chip
- 850,000 cores


**Cerebras WSE-2**
2.6 Trillion transistors
46,225 mm$^2$

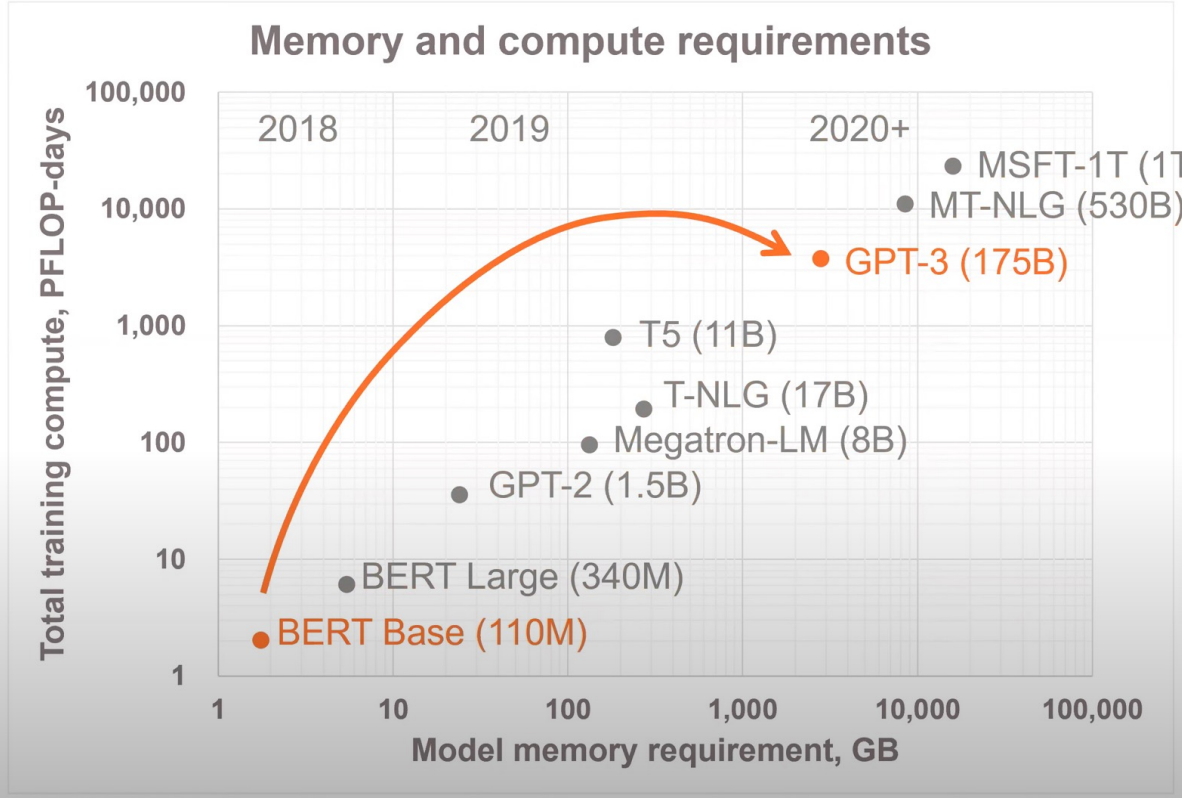
**Largest GPU**
54.2 Billion transistors
826 mm$^2$
**NVIDIA** Ampere GA100

https://www.anandtech.com/show/14758/hot-chips-31-live-blogs-cerebras-wafer-scale-deep-learning

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

# Huge Demand for Performance & Efficiency

## Exponential Growth of Neural Networks




**1800x more compute**

In just **2 years**

**Tomorrow**, **multi-trillion** parameter models

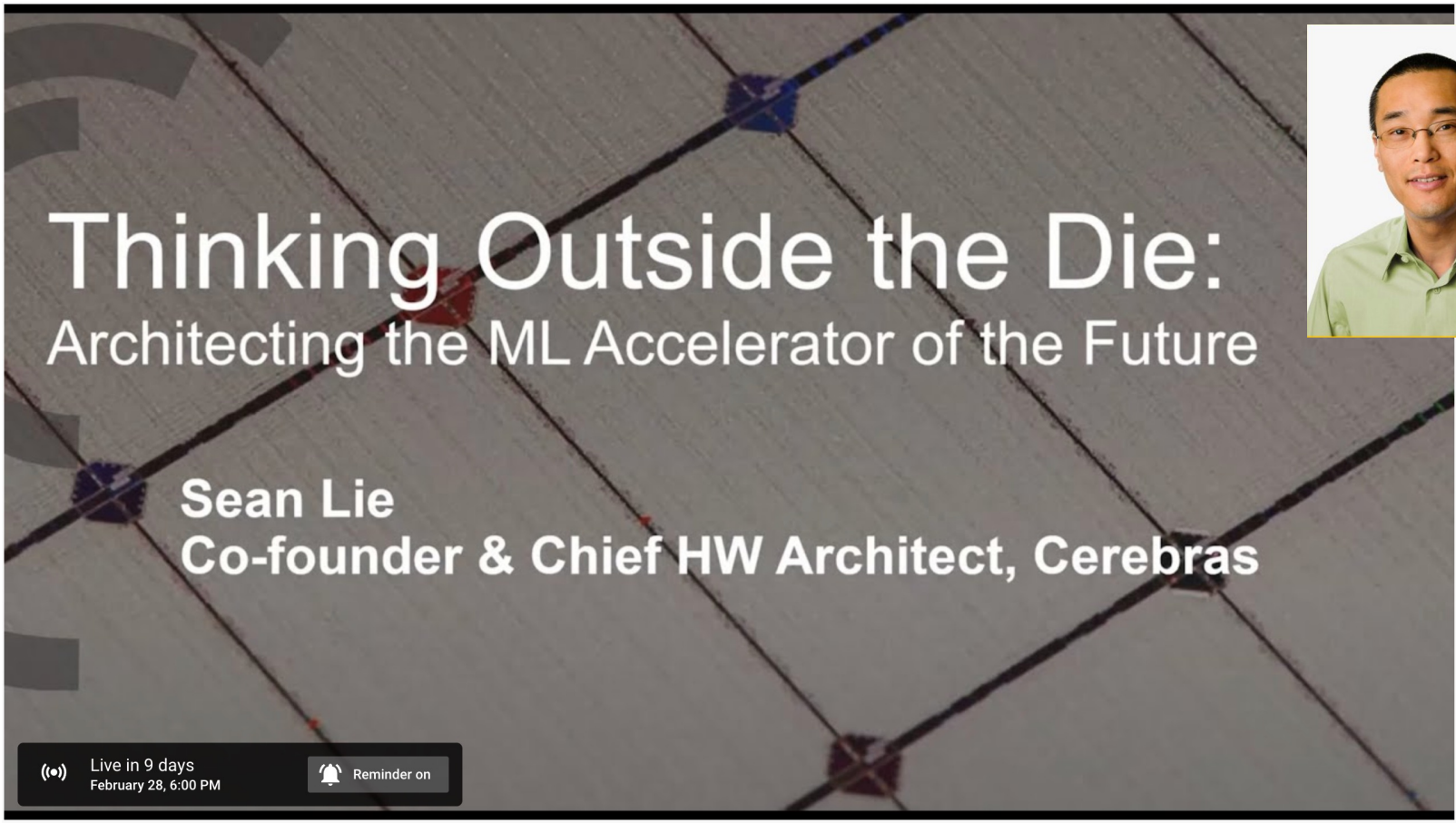




https://www.youtube.com/watch?v=x2-qB0J7KHw

# More on the Cerebras WSE

**https://www.youtube.com/watch?v=x2-qB0J7KHw**



# Microsoft Brainwave FPGA Acceleration

## Serving DNNs in Real Time at Datacenter Scale with Project Brainwave

**Eric Chung, Jeremy Fowers, Kalin Ovtcharov, Michael Papamichael, Adrian Caulfield, Todd Massengill, Ming Liu, Daniel Lo, Shlomi Alkalay, Michael Haselman, Maleen Abeydeera, Logan Adams, Hari Angepat, Christian Boehn, Derek Chiou, Oren Firestein, Alessandro Forin, Kang Su Gatlin, Mahdi Ghandi, Stephen Heil, Kyle Holohan, Ahmad El Husseini, Tamas Juhasz, Kara Kagi, Ratna K. Kovvuri, Sitaram Lanka, Friedel van Megen, Dima Mukhortov, Prerak Patel, Brandon Perez, Amanda Grace Rapsang, Steven K. Reinhardt, Bita Darvish Rouhani, Adam Sapek, Raja Seera, Sangeetha Shekar, Balaji Sridharan, Gabriel Weisz, Lisa Woods, Phillip Yi Xiao, Dan Zhang, Ritchie Zhao, and Doug Burger**
Microsoft Corporation

To meet the computational demands required of deep learning, cloud operators are turning toward specialized hardware for improved efficiency and performance. Project Brainwave, Microsoft's principal infrastructure for AI serving in real time, accelerates deep neural network (DNN) inferencing in major services such as Bing's intelligent search features and Azure. Exploiting distributed model parallelism and pinning over low-latency hardware microservices, Project Brainwave serves state-of-the-art, pre-trained DNN models with high efficiencies at low batch sizes. A high-performance, precision-adaptable FPGA soft processor is at the heart of the system, achieving up to 39.5 teraflops (Tflops) of effective performance at Batch 1 on a state-of-the-art Intel Stratix 10 FPGA.

The recent successes of machine learning, enabled largely by the rise of DNNs, have fueled a growing demand for ubiquitous AI, ranging from conversational agents to object recognition to intelligent search. While state-of-the-art DNNs continue to deliver major breakthroughs in challenging AI domains such as computer vision and natural language processing, their computational demands have steadily outpaced the performance growth rate of standard CPUs. These trends have spurred a

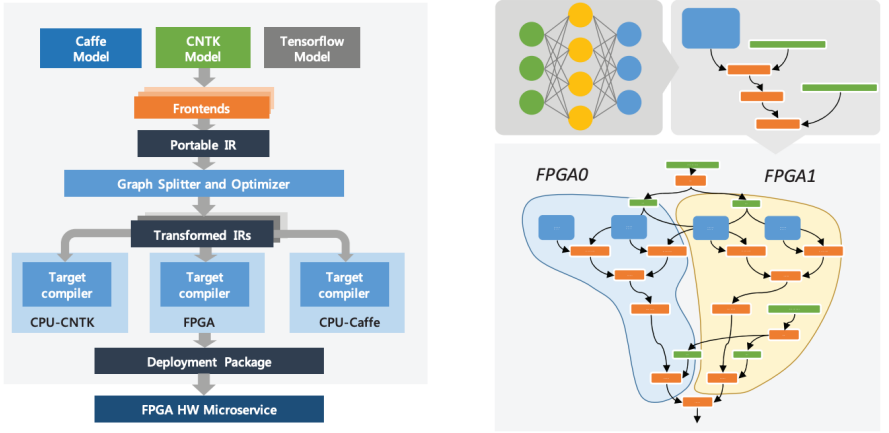



Figure 3. The framework-neutral Brainwave tool flow accepts models from different DNN toolchains and exports them into a common intermediate graph representation. Tool flow optimizes the intermediate representation and partitions it into sub-graphs assigned to different CPUs and FPGAs. Device-specific backends generate device assembly and are linked together by a federated runtime that gets deployed into a live FPGA hardware microservice.

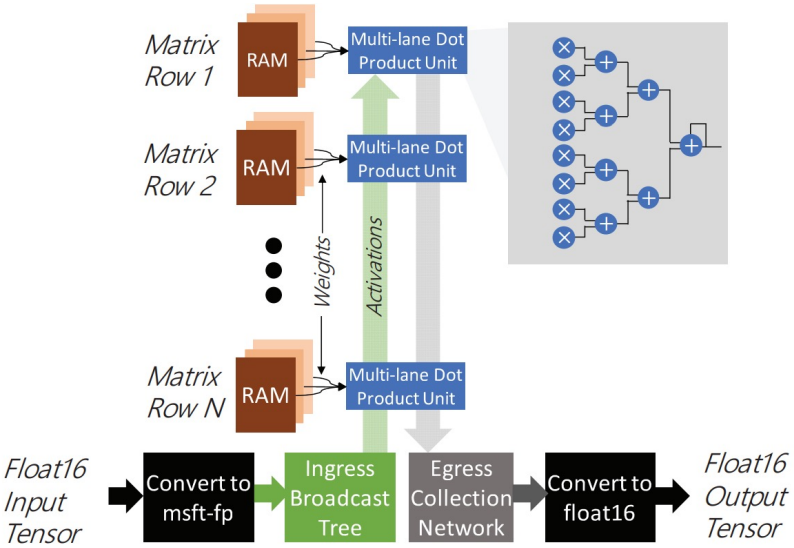



Figure 5. An independent SRAM memory port is dedicated to every lane of a multi-lane vector dot product unit within the MVU, allowing up to 80,000 MACs on a Stratix 10 280 to be fed with independent weights. As a result, FPGA can achieve near-peak utilization on Batch 1-oriented matrix-vector multiplication.

# Computer Architecture

## Lecture 28: Systolic Array Architectures

Prof. Onur Mutlu
ETH Zürich

Fall 2023

1 February 2024