UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Accelerators/Specialization/ Emerging Architectures

ECE 411 Spring 2024

Guest Lecture: Muhammad Husnain Mubarik

04/11/2024

Outline of Lecture

- **The Six Ideas of Computer Architecture**
- **Systolic Arrays**
- **Systolic Arrays: Motivation**
- **Systolic Architectures**
- **Systolic Computational Kernels**
- **Dataflows**
- **Common Optimizations**
- **Design Considerations**

Some slides are inspired by lectures of Prof. Onur Mutlu (ETH Zürich)

The Six Ideas in Computer Architecture

- **Surprisingly, there are only six distinct concepts used to design computers, apart from technology advances, that reappear in many guises over and over again:**

  - **Locality – Spatial & Temporal – Caches**
  - **Prediction – No State Change – Branch Prediction**
  - **Speculation – State Change – Prefetching**
  - **Indirection – Virtualization**
  - **Parallelism – Pipelining, OoO, Vectors**
  - **Specialization – GPUs, Accelerators**

*** from Trevor Mudge (Michigan) - https://tnm.engin.umich.edu/**

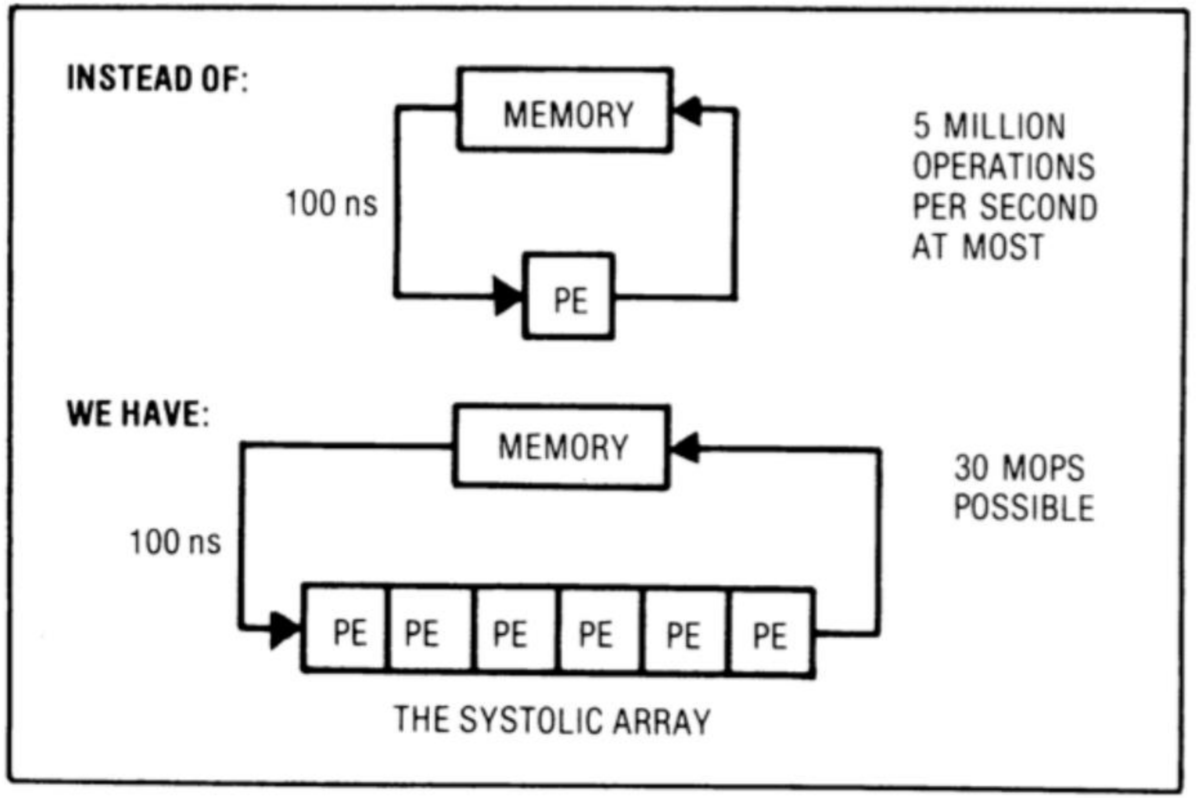
Systolic Arrays

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

Why Systolic Architectures?

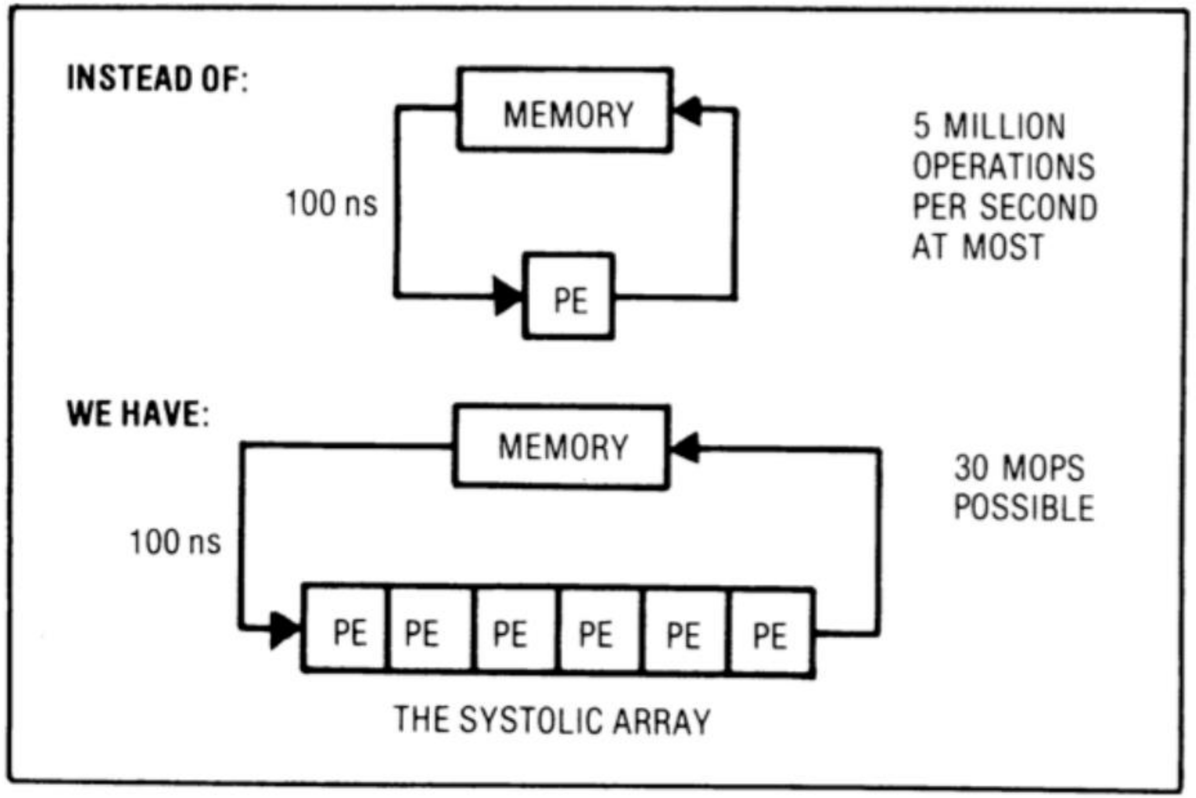
- **Idea: Data flows from the computer memory in a rhythmic fashion, passing through many processing elements before it returns to memory**
- **Special purpose accelerators/architectures need**
  - **Simple, regular design (keep # unique parts small and regular)**
  - **High concurrency → high performance**
  - **Balanced computation and I/O (memory) bandwidth**



H. T. Kung, “Why Systolic Architectures?,” IEEE Computer 1982.

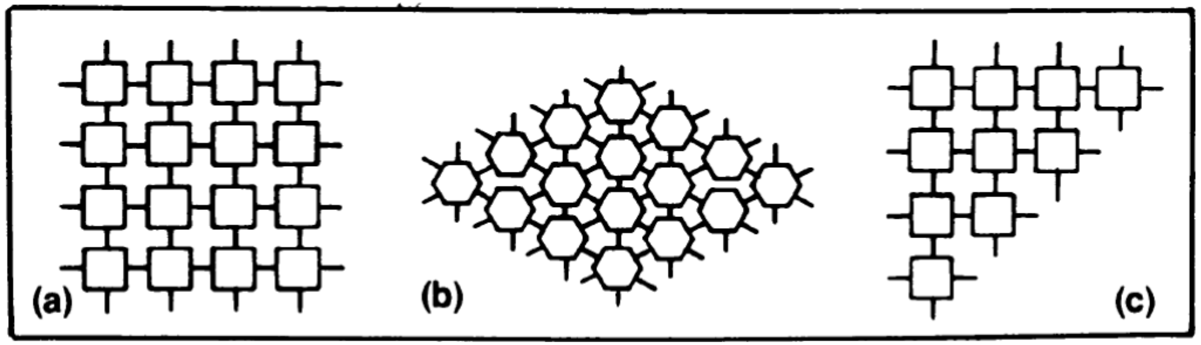
Systolic Architectures

- **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**
- **Differences from pipelining:**
  - **These are individual PEs**
  - **Array structure can be non-linear and multi-dimensional**



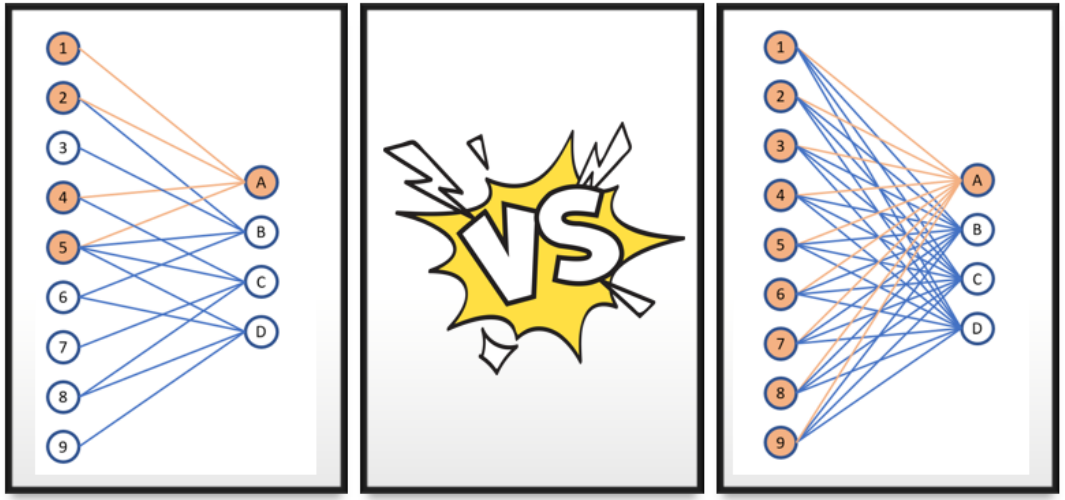
Systolic Architectures

- **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**
- **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)**

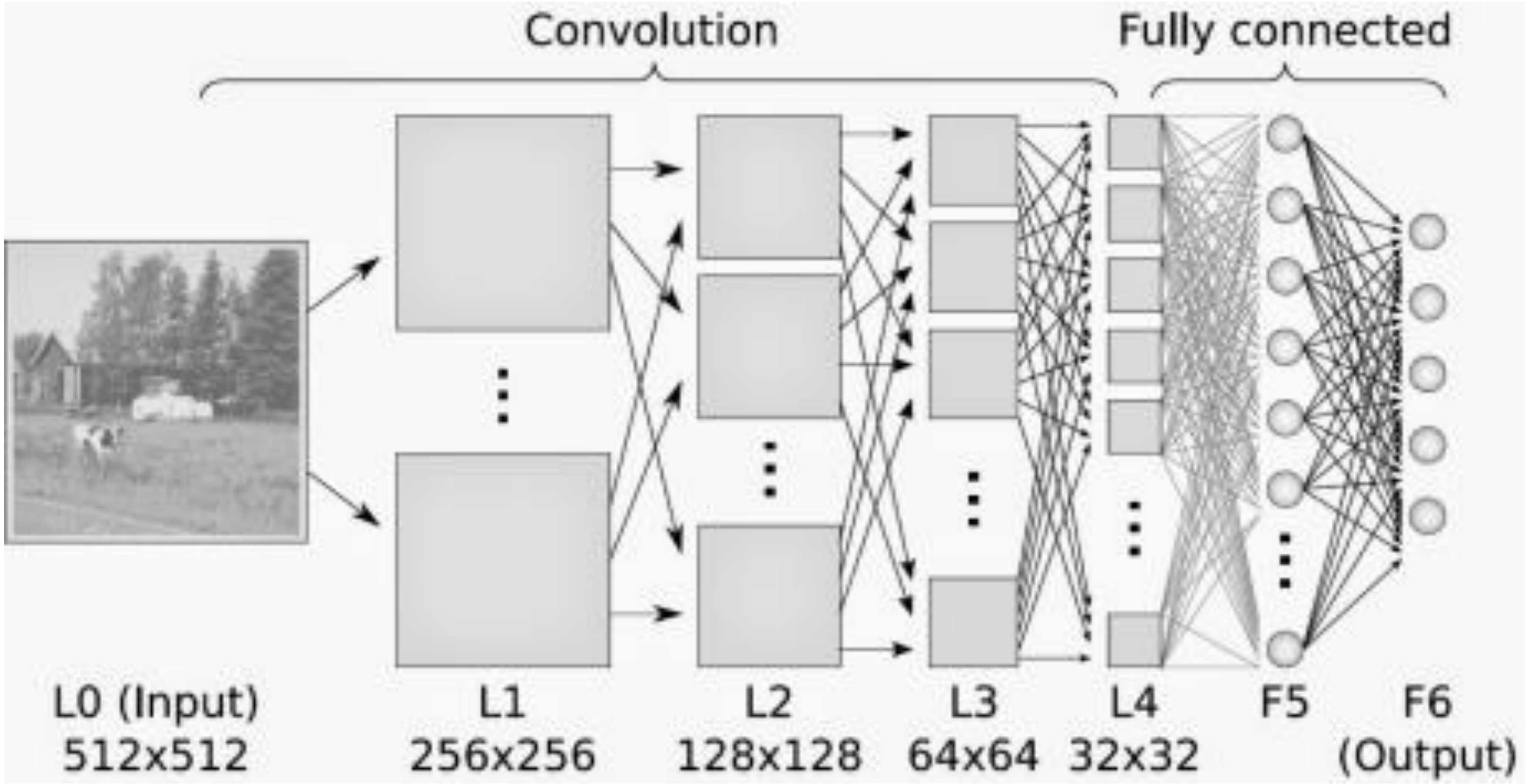


Systolic Computation Example

- **Convolution**
  - **Sparse connections**
  - **Convolutional Neural Networks (CNN)**
  - **Information density**
- **Fully Connected Layers**
  - **Feed forward, dense connections**
  - **Multilayer Perceptron (MLP)**
  - **Information density**
- **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)**



A Typical DNN Architecture



Systolic Arrays: Pros and Cons

- **Advantages:**
  - **Principled: Efficiently makes use of limited memory bandwidth, balances computation to I/O bandwidth availability**
  - **Specialized (computation needs to fit PE organization/functions)**
    - **improved efficiency, simple design, high concurrency / performance**
    - **good to do more with less memory bandwidth requirement**
- **Downsides:**
  - **Not good at exploiting irregular parallelism**
  - **Specialized**
    - **not generally applicable because computation needs to fit the PE functions/organization**

An Example Modern Systolic Array: TPU

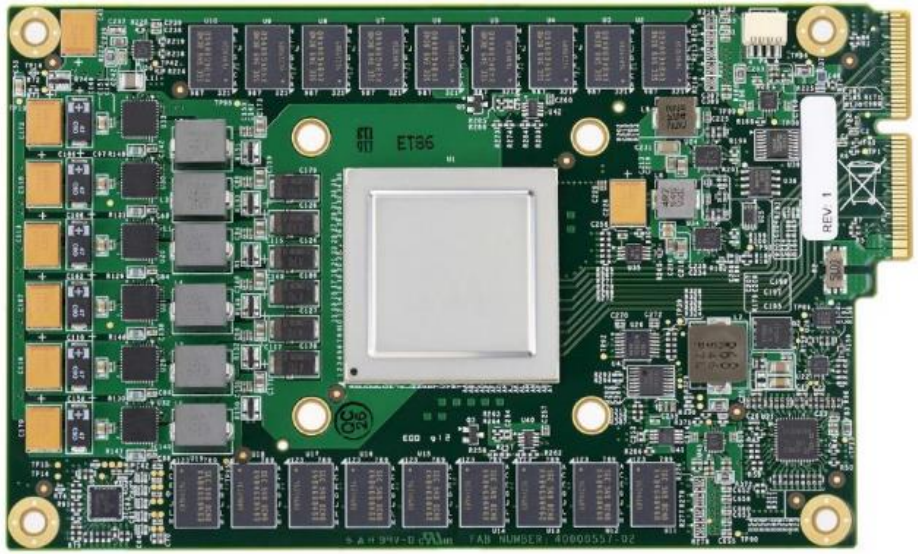


Figure 3. TPU Printed Circuit Board. It can be inserted in the slot for an SATA disk in a server, but the card uses PCIe Gen3 x16.

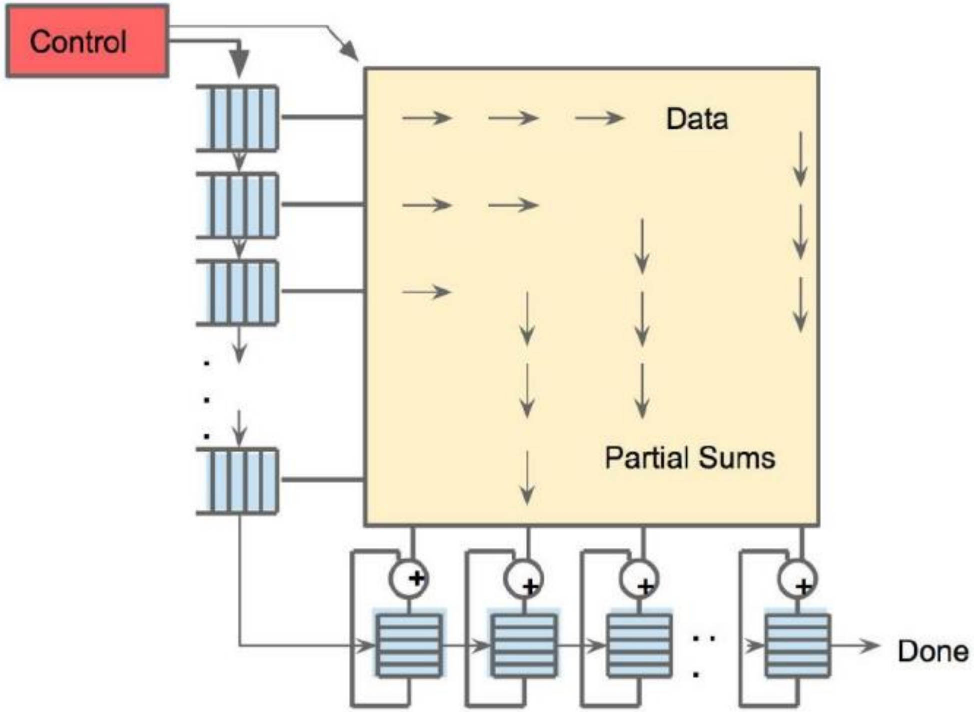


Figure 4. Systolic data flow of the Matrix Multiply Unit. Software has the illusion that each 256B input is read at once, and they instantly update one location of each of 256 accumulator RAMs.

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

TPU: Second Generation

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

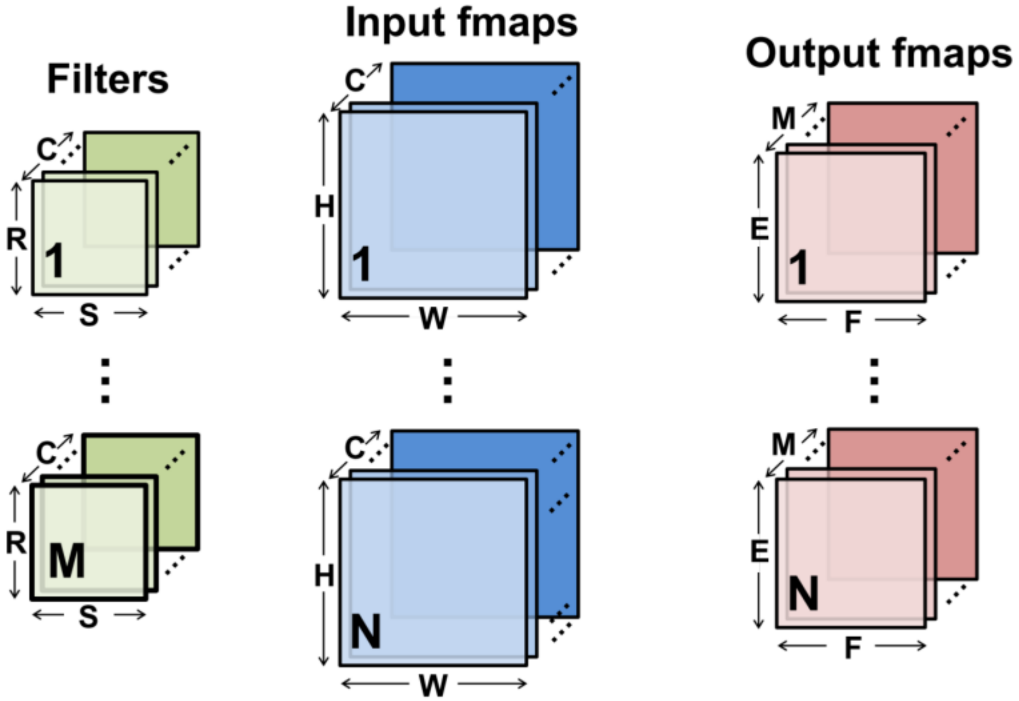
45 TFLOPS per chip
vs 23 TOPS

Designed for training
and inference
vs only inference

Mapping Kernels on Systolic Arrays

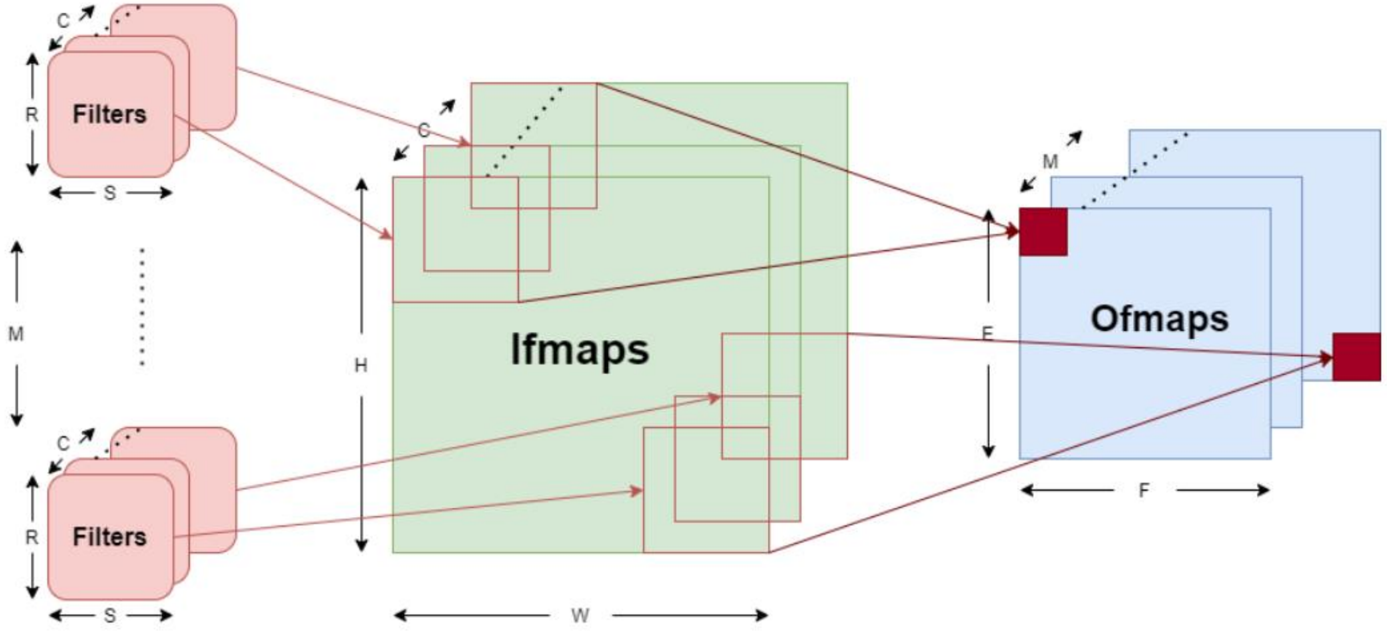
Systolic Computation Example -- Convolution

- **H – Height of input fmap (activations)**
- **W – Width of input fmap (activations)**
- **C – Number of 2-D input fmaps /filters (channels)**
- **R – Height of 2-D filter (weights)**
- **S – Width of 2-D filter (weights)**
- **M – Number of 2-D output fmaps (channels)**
- **E – Height of output fmap (activations)**
- **F – Width of output fmap (activations)**
- **N – Number of input fmaps/output fmaps (batch size)**



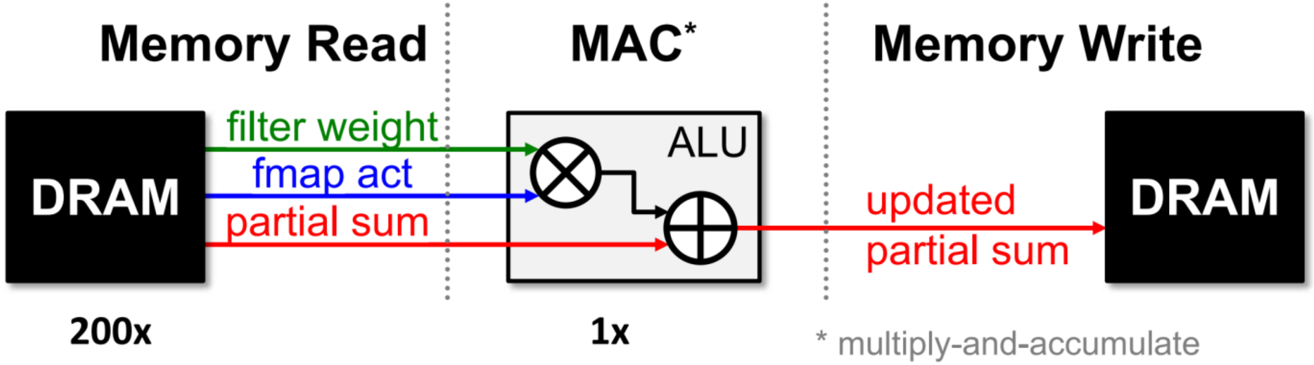
Systolic Computation Example -- Convolution

- **Element-wise multiplication**
- **Partial sum accumulation**
- **Sliding window processing**



Properties We Can Leverage

- **Operations exhibit high parallelism**
  - **High throughput possible**
- **Memory Access is the Bottleneck**

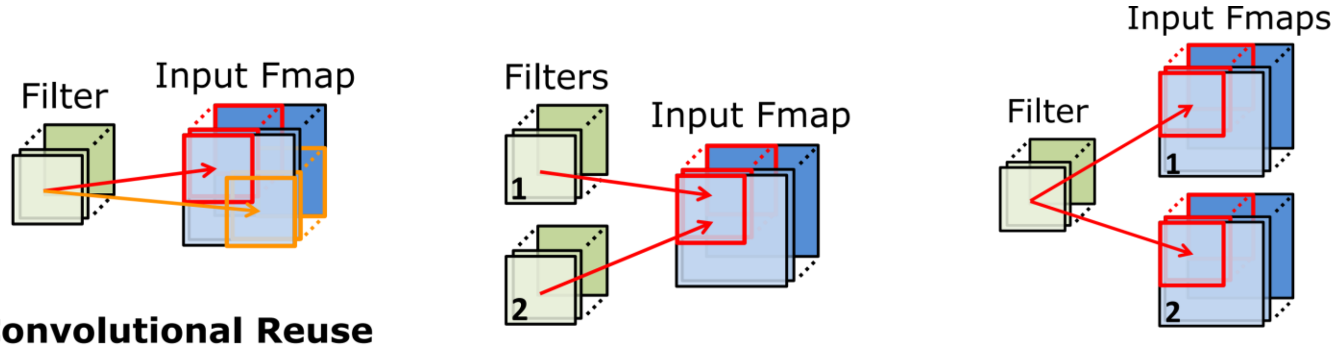


Worst Case: all memory R/W are **DRAM** accesses

Example: AlexNet has **724M** MACs → **2896M** DRAM accesses required

Properties We Can Leverage

- **Operations exhibit high parallelism**
  - **High throughput possible**
- **Input data reuse opportunities (e.g., up to 500x for AlexNet)**
  - **Exploit low-cost memory**



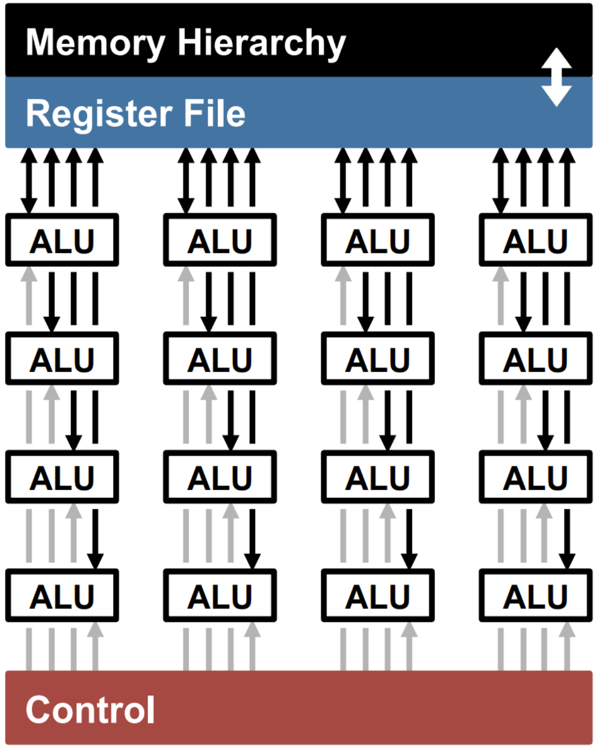
Convolutional Reuse
(Activations, Weights)
CONV layers only
(sliding window)

Fmap Reuse
(Activations)
CONV and FC layers

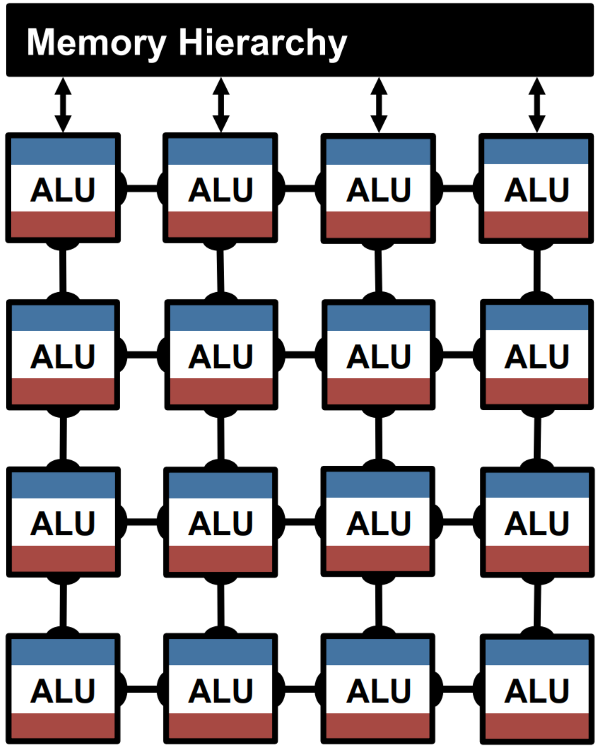
Filter Reuse
(Weights)
CONV and FC layers
(batch size > 1)

Highly-Parallel Compute Paradigms

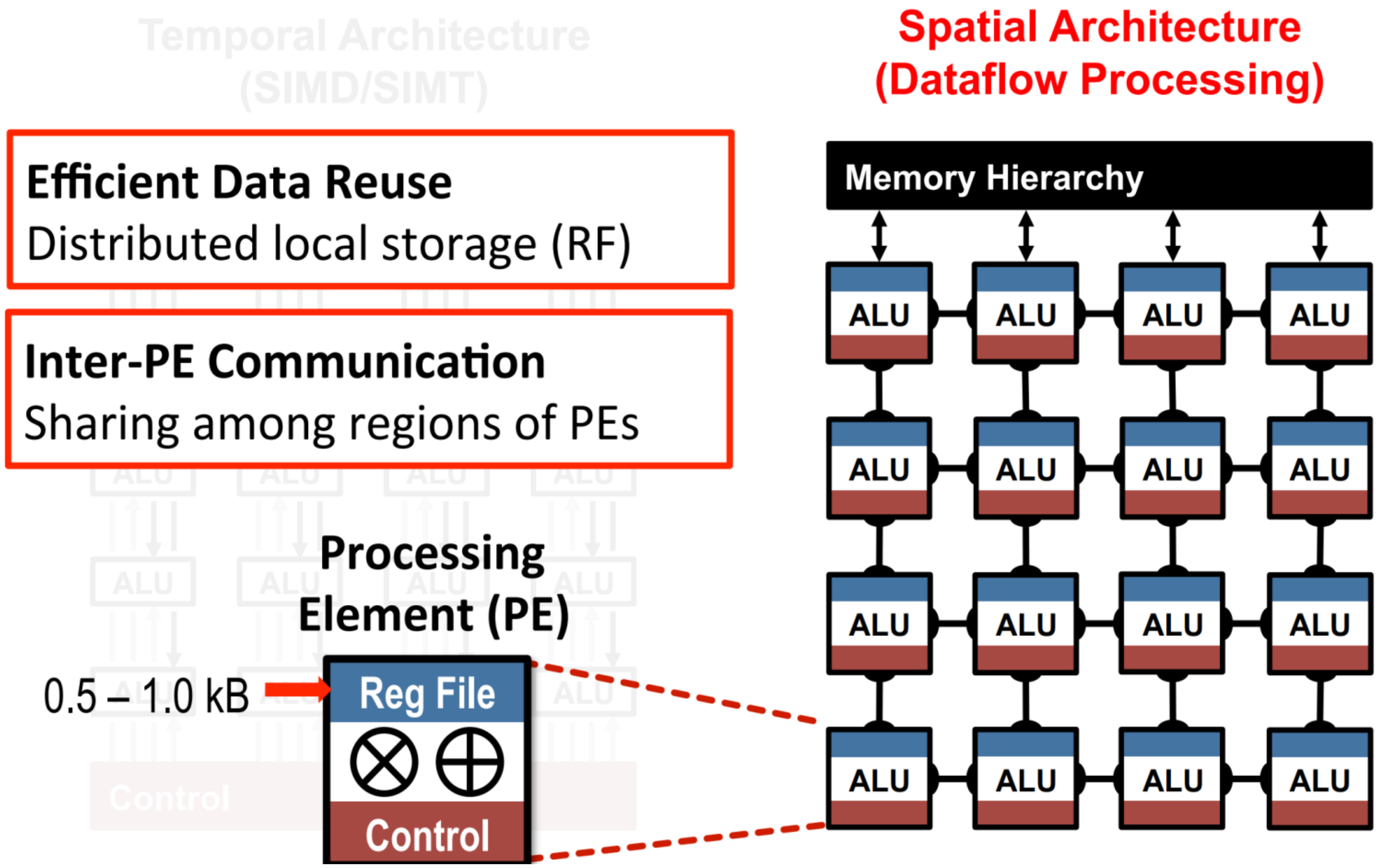
Temporal Architecture (SIMD/SIMT)



Spatial Architecture (Dataflow Processing)



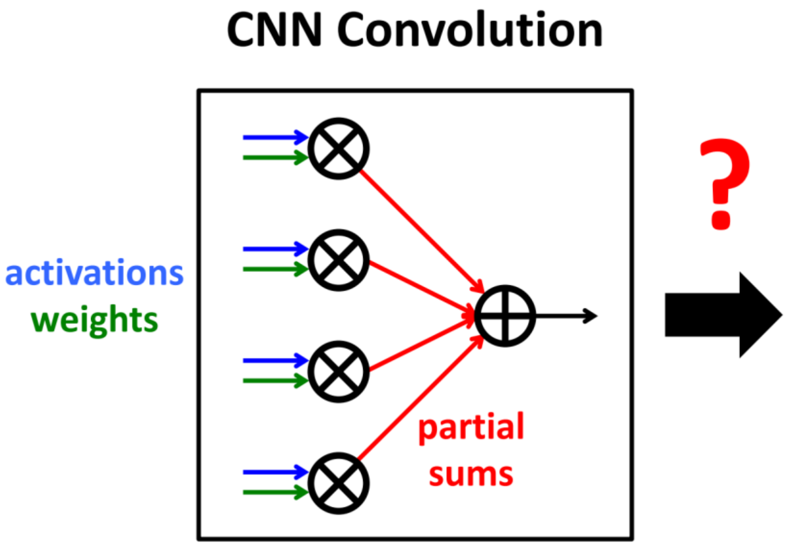
Advantages of Spatial Architecture

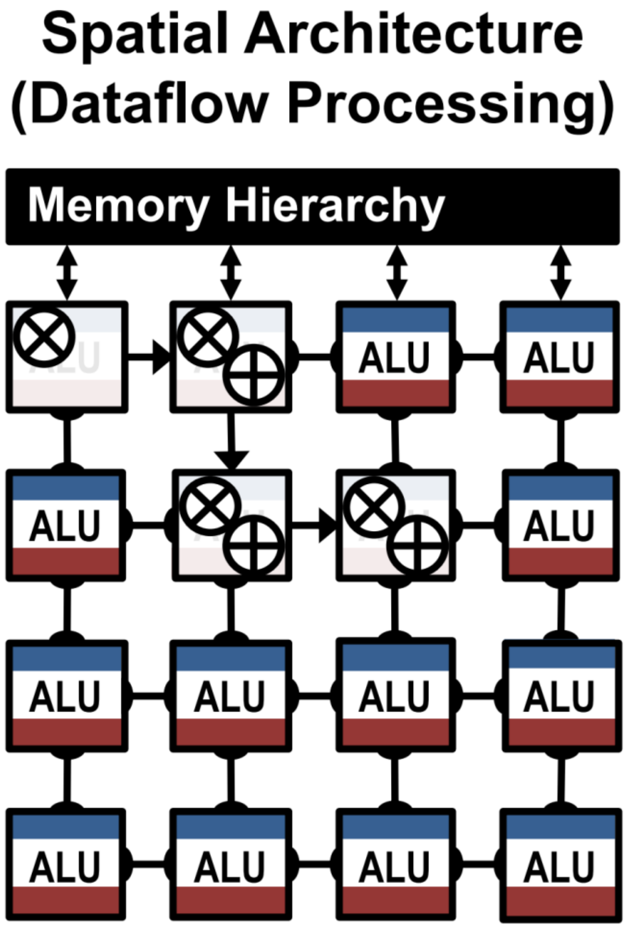


Efficient Data Reuse
Distributed local storage (RF)

Inter-PE Communication
Sharing among regions of PEs

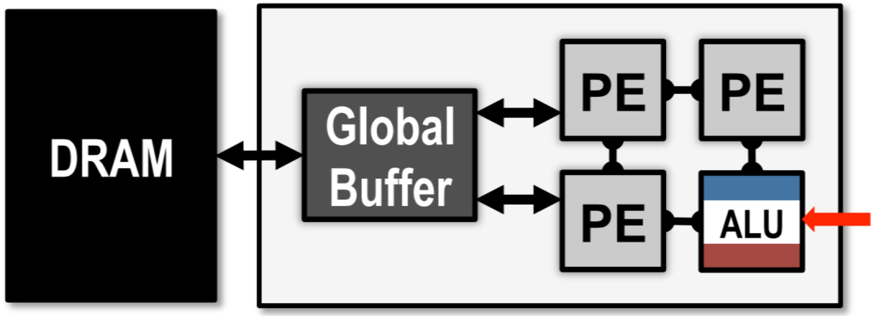
How to Map the Dataflow?





Goal: Increase reuse of input data (weights and activations) and local partial sums accumulation

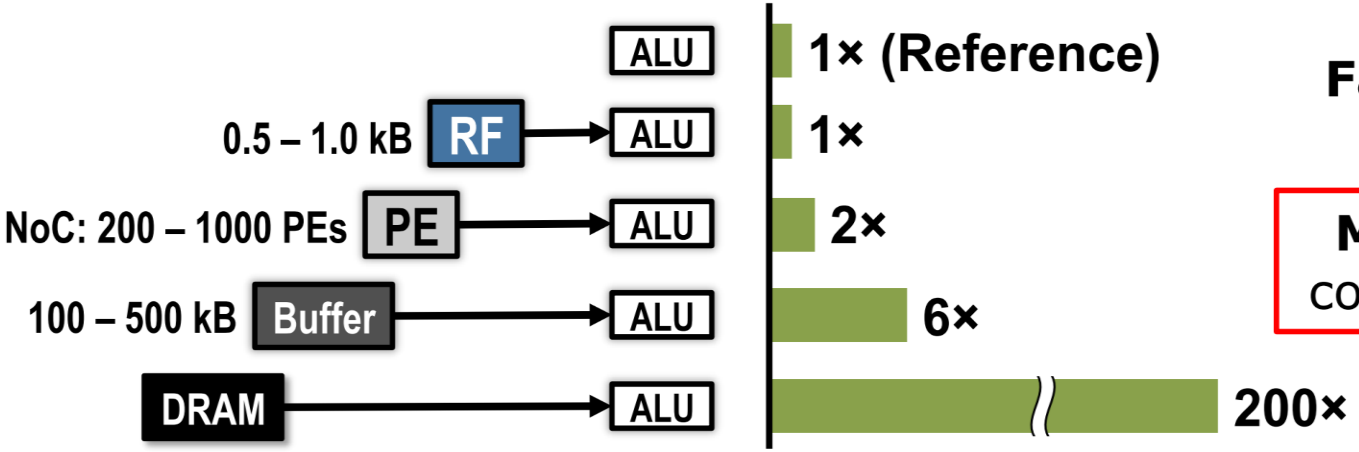
Data Movement is Expensive



fetch data to run
a MAC here

Specialized hardware with small (< 1kB)
low cost memory near compute

Normalized Energy Cost*



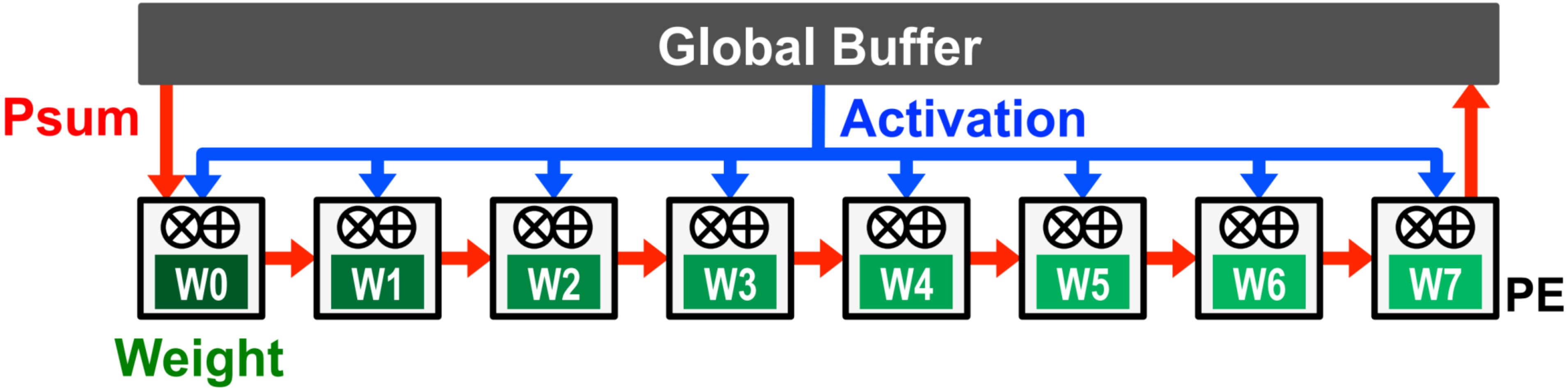
Farther and **larger** memories
consume more power

Maximize data reuse at low
cost levels of memory hierarchy

Data Flows

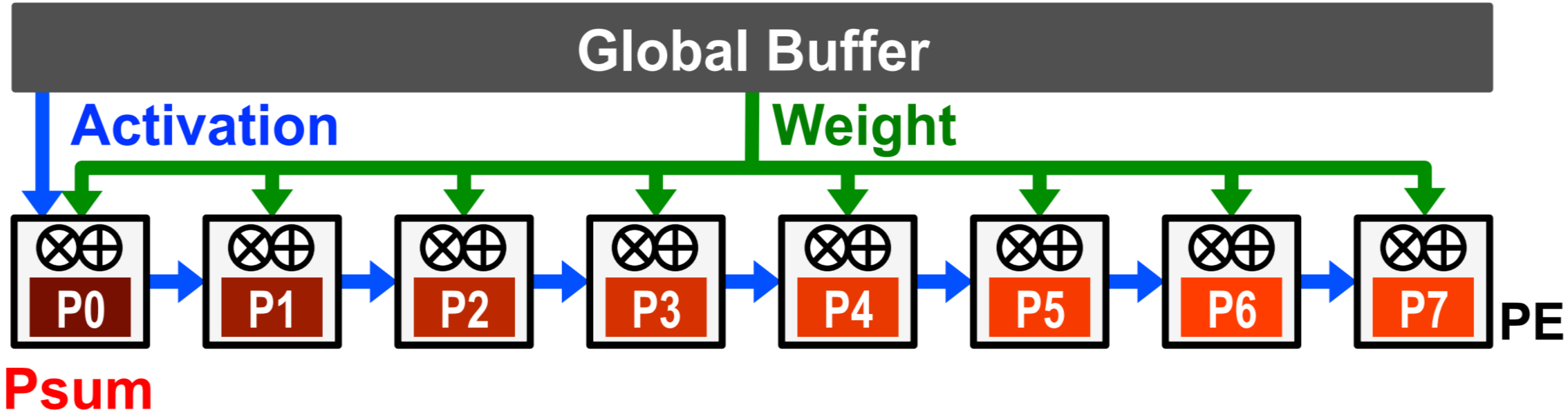
Weight Stationary (WS)

- Minimize weight read energy consumption
    - Maximize convolutional and filter reuse of weights
- Broadcast activations and accumulate partial sums spatially across the PE array
- Examples: TPU [Jouppi, ISCA 2017], NVDLA



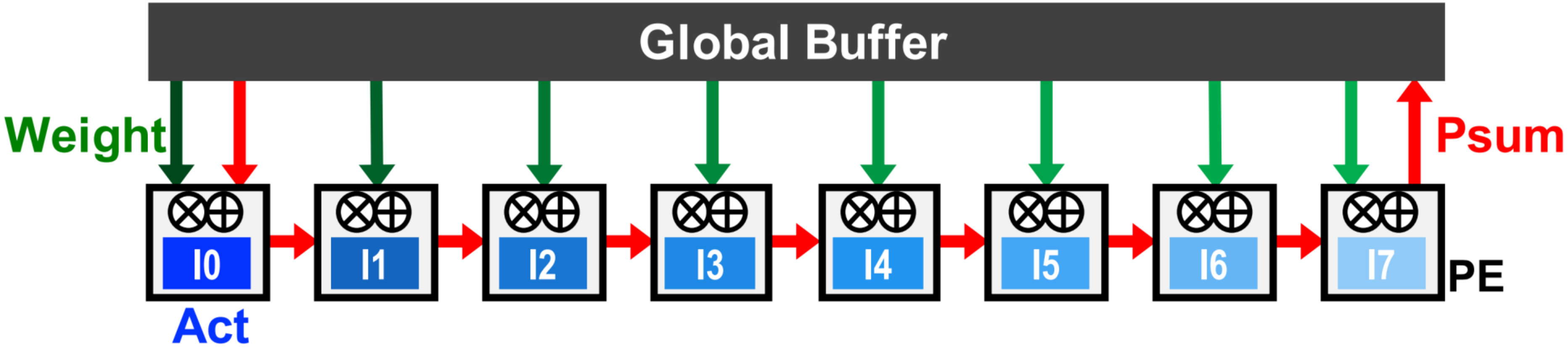
Output Stationary (OS)

- Minimize partial sum R/W energy consumption
  - Maximize local accumulation
- Broadcast/Multicast filter weights and reuse activations spatially across the PE array
- Examples: [Moons, VLSI 2016], [Thinker, VLSI 2017]



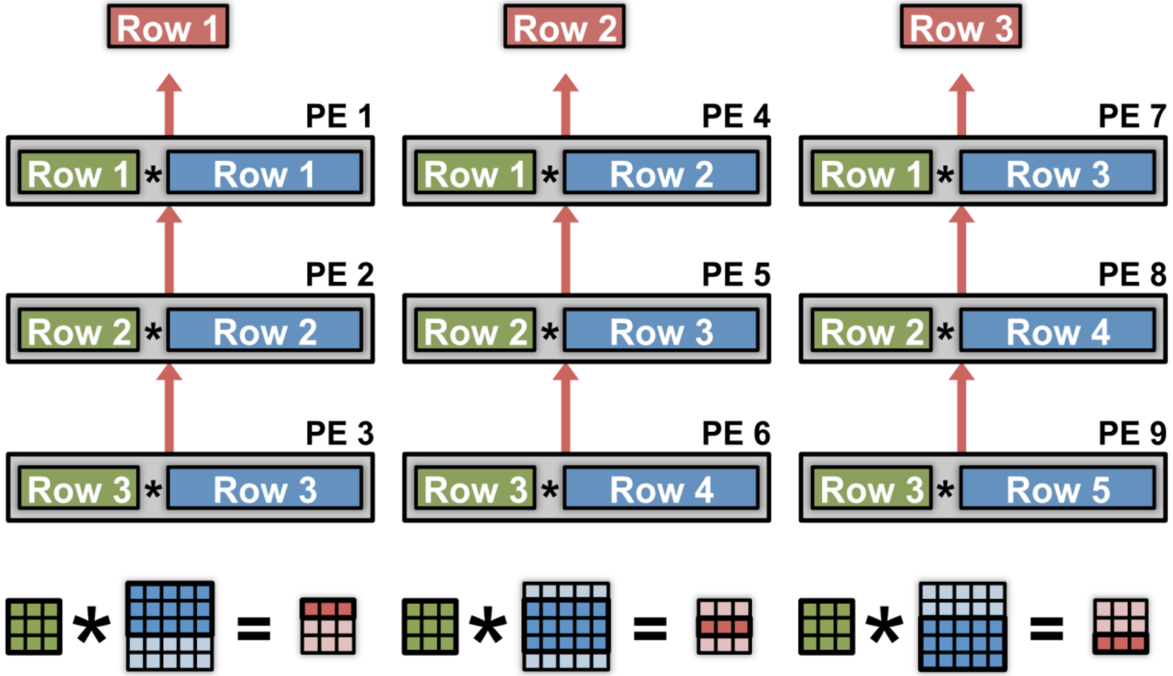
Input Stationary (IS)

- **Minimize activation read energy consumption**
  - **Maximize convolutional and fmap reuse of activations**
- **Unicast weights and accumulate partial sums spatially across the PE array**
- **Example: [SCNN, ISCA 2017]**



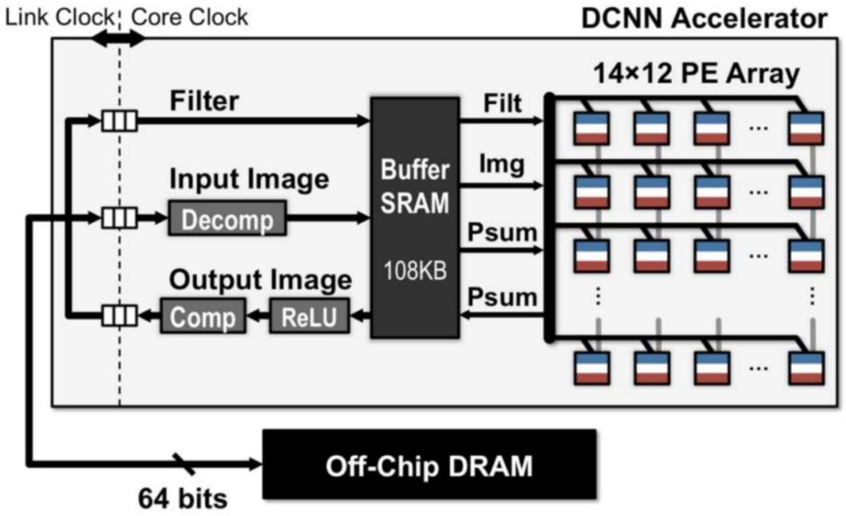
Row Stationary Dataflow

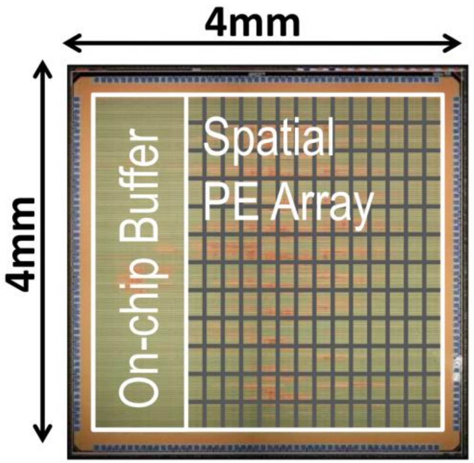
- **Maximize row convolutional reuse in RF**
  - **Keep a filter row and fmap sliding window in RF**
- **Maximize row psum accumulation in RF**
- **Optimize for overall energy efficiency instead for only a certain data type**



Eyeriss: Deep Neural Network Accelerator

- **Exploits data reuse for 100x reduction in memory accesses from global buffer and 1400x reduction in memory accesses from off-chip DRAM**
- **Overall >10x energy reduction compared to a mobile GPU (Nvidia TK1)**





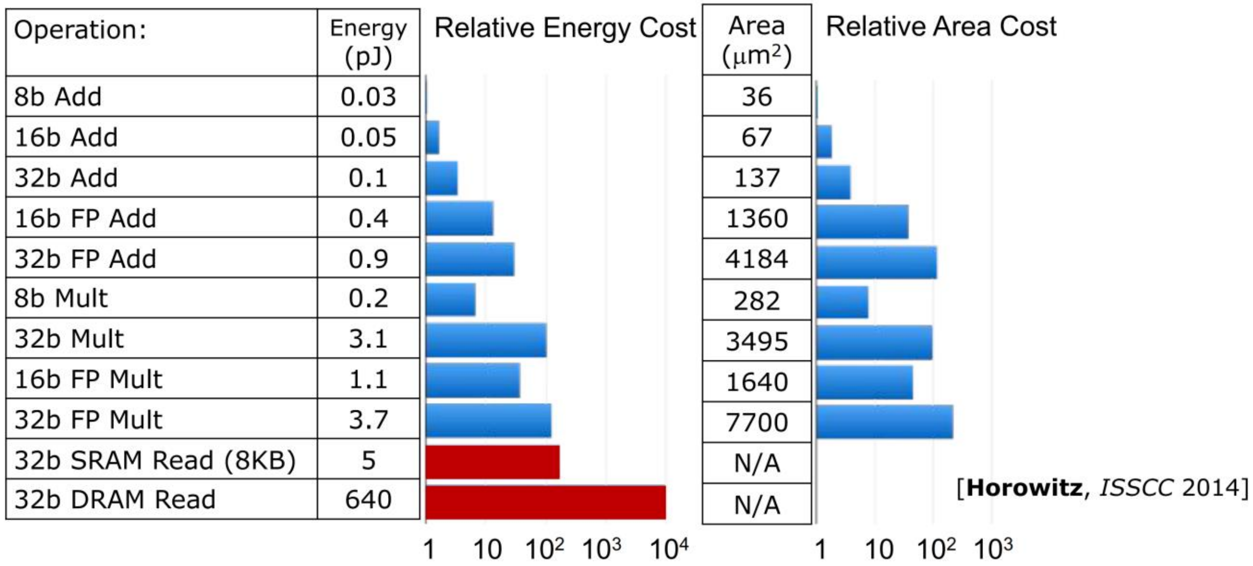
[**Chen**, *ISSCC* 2016]

Hardware / Software Co-Design

- Co-design algorithm + hardware -> better than what each could achieve alone
- A few popular co-design approaches:
  - Reduce size of operands for storage/compute (Reduced Precision)
  - Reduce number of operations for storage/compute (Sparsity and Efficient Network Architecture)
- Hardware support required to increase savings in latency and energy
  - Ensure that overhead of hardware support does not exceed benefits
- Unlike previously discussed approaches, these approaches can affect accuracy!
  - Evaluate tradeoff between accuracy and other metrics

Reduced Precision

- **Reduce data movement and storage cost for inputs and outputs of MAC**
- **Smaller memory -> lower energy**
- **Reduce cost of MAC**
- **Cost of multiply increases with bit width (n) -> energy and area by O(n2); delay by O(n)**



Operation:	Energy (pJ)	Area (μm^2)
8b Add	0.03	36
16b Add	0.05	67
32b Add	0.1	137
16b FP Add	0.4	1360
32b FP Add	0.9	4184
8b Mult	0.2	282
32b Mult	3.1	3495
16b FP Mult	1.1	1640
32b FP Mult	3.7	7700
32b SRAM Read (8KB)	5	N/A
32b DRAM Read	640	N/A

[**Horowitz**, *ISSCC* 2014]

Sparsity / Pruning

- **Reduce number of MACs**
  - **Anything multiplied by zero is zero -> avoid performing unnecessary MACs**
  - **Reduce energy consumption and latency**
- **Reduce data movement**
  - **If one of the inputs to MAC is zero, can avoid reading the other input**
  - **Compress data by only sending non-zero values**
- **Example: AlexNet weight reduction by pruning**
  - **CONV layers 2.7x; FC layers 9.9x**
  - **Overall Reduction: Weights 9x, MACs 3x**

Design Considerations for Sparsity

- **Impact on accuracy**
  - **Must consider difficulty of dataset, task, and DNN model**
    - **e.g., AlexNet and VGG known to be over parameterized and thus easy to prune weights; does method work on efficient DNN models?**
- **Does hardware cost exceed benefits?**
  - **Need extra hardware to identify sparsity**
    - **e.g., Additional logic to identify non-zeros and store non-zero locations**
  - **Accounting for sparsity in both weights and activations is challenging**
    - **Need to compute intersection of two data streams rather than find next non-zero in one**
  - **Compressed data will be variable length**
    - **Reduced flexibility in access order -> random access will have significant overhead**

Design Considerations for Co-Design

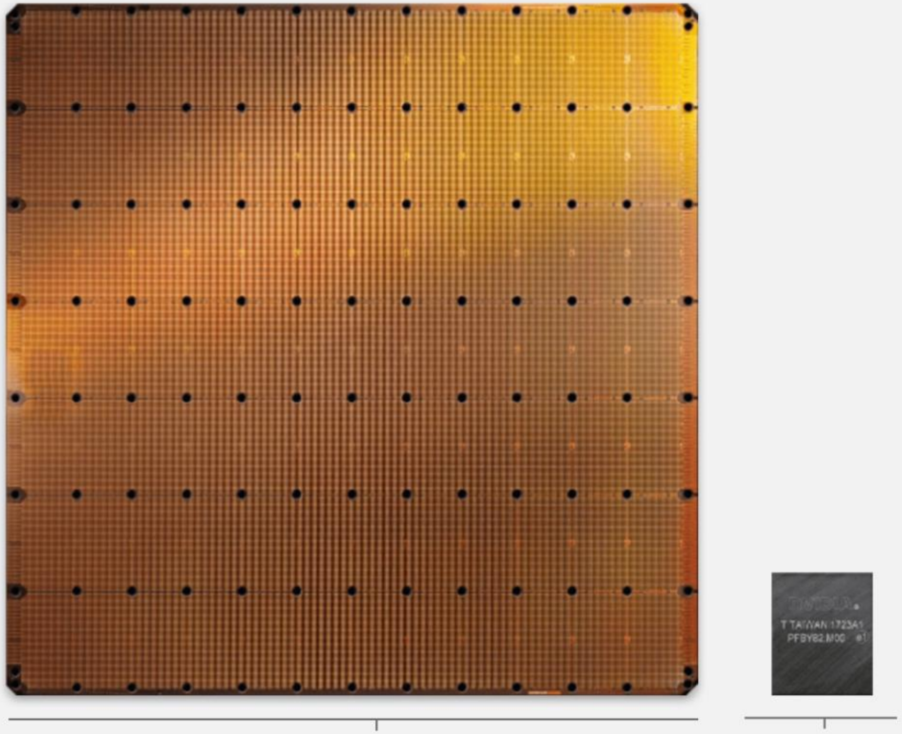
- **Impact on accuracy**
  - **Consider quality of baseline (initial) DNN model, difficulty of task and dataset**
  - **Sweep curve of accuracy versus latency/energy to see the full tradeoff**
- **Does hardware cost exceed benefits?**
  - **Need extra hardware to support variable precision and shapes or to identify sparsity**
- **Time required to perform co-design**
  - **E.g., Difficulty of tuning affected by**
    - **Number of hyperparameters**
    - **Uncertainty in relationship between hyperparameters and impact on performance**
- **How does the approach perform on different platforms?**
  - **Is the approach a general method, or applicable on specific hardware?**

Design Considerations for ASIC

- **Increase PE utilization**
  - **Flexible mapping and on-chip network for different DNN models -> requires additional hardware**
- **Reduce data movement**
  - **Custom memory hierarchy and dataflows that exploit data reuse**
  - **Apply compression to exploit redundancy in data -> requires additional hardware**
- **Reduce time and energy per MAC**
  - **Reduce precision -> if precision varies, requires additional hardware; impact on accuracy**
- **Reduce unnecessary MACs**
  - **Exploit sparsity -> requires additional hardware; impact on accuracy**
  - **Exploit redundant operations -> requires additional hardware**

Cerebras’s Wafer Scale Engine – 2

- **Cerebras’s WSE-2**
  - **Largest ML accelerator chip (2021)**
  - **850,000 cores**
  - **2.6 Trillion transistors vs (54.2B - A100)**
  - **46,225 mm2 vs (826mm2 - A100)**
  - **Map NN on whole wafer**
  - **MIMD machine**
    - **Each processor is SIMD processor**
    - **4-way SIMD for 16-bit FP operands**
    - **48 KB of local SRAM**



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

Largest GPU
826mm^2 Silicon
54.2 Billion transistors

The Grainger College of Engineering

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN