Deep Learning Processors
Data Flow &
Domain Specific Architecture

Chixiao Chen

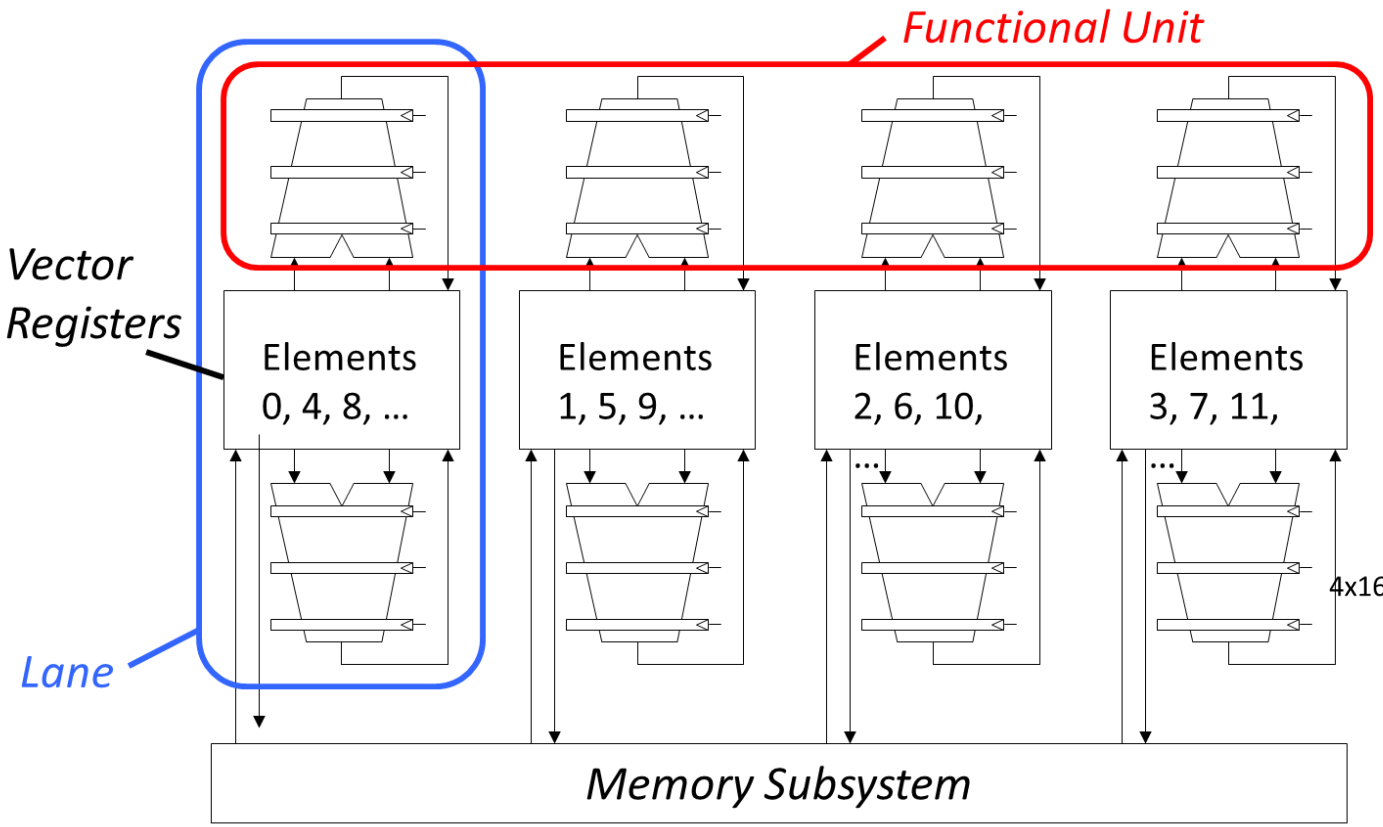
Announcement

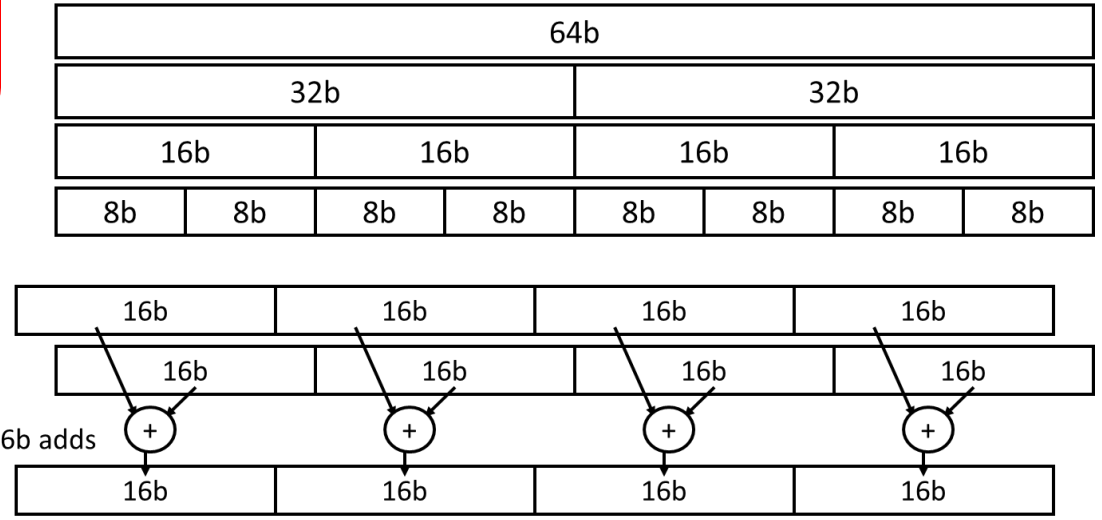
- We do not have class on 5/23, 5/20.
  6/6's class is on pending

- HW 4 is merged with Final Project
- HW4 & Final PJ will be on-line this week end.
  - Option I: Implementing a SIMD instruction to perform a NN (RTL) & propose a new technique (not mentioned from class) to further increase the performance (reference and slides)
  - Option II: Implementing an **advanced** instruction to realize the NN

Lecture Last Time

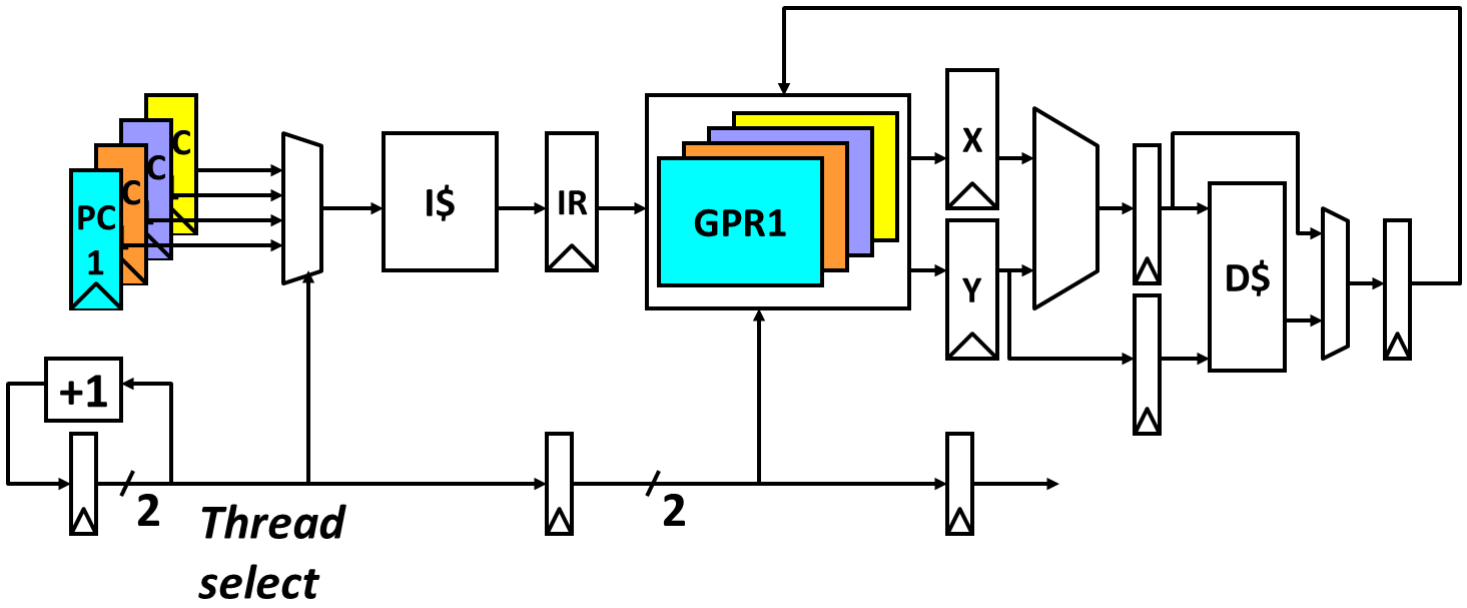
- Vector Processor and SIMD





Lecture Last Time

- Multithread Pipeline Architecture & GPU

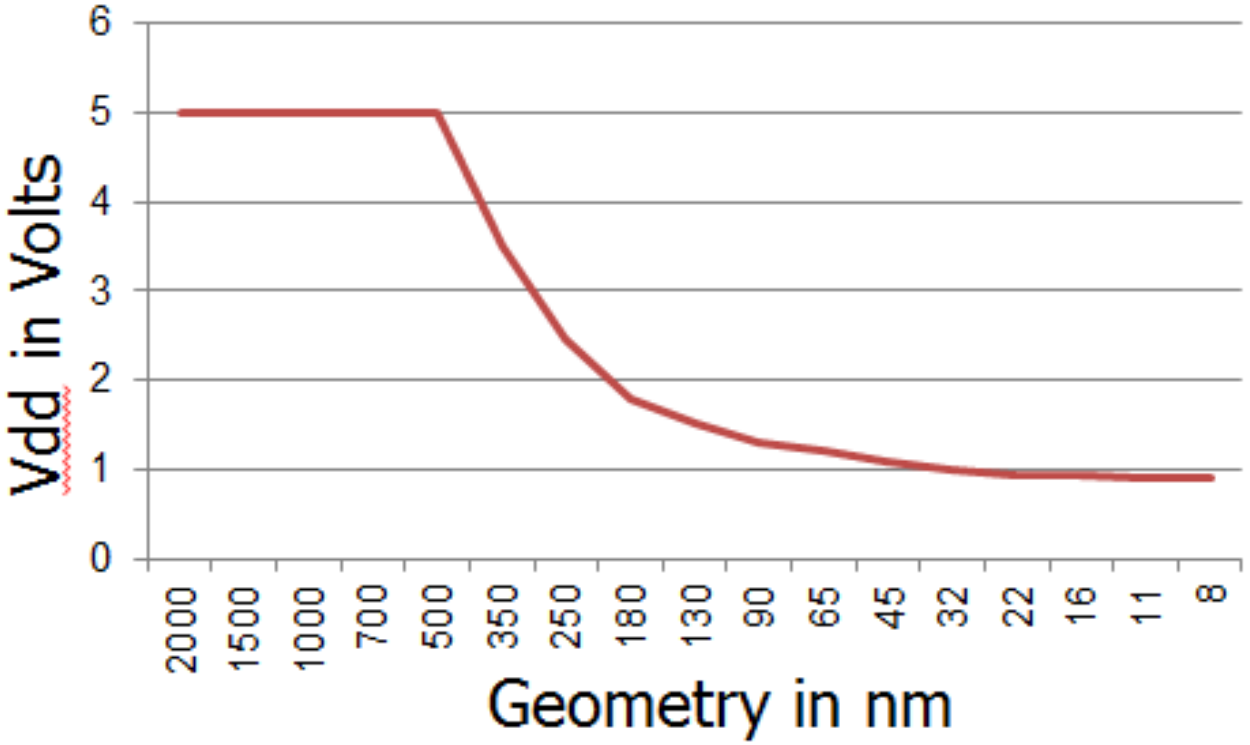


Overview

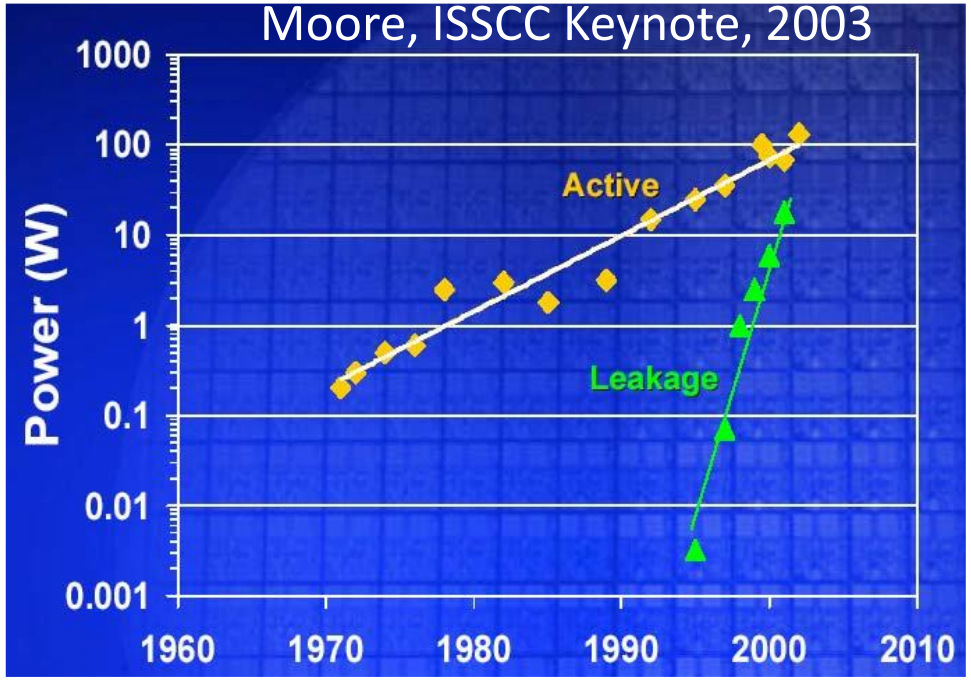
- Deep learning hardware not using instructions
- Reconfigurable hardware
  - FPGA
  - CGRA
- Data flows
  - Systolic arrays
  - Weight/output/**Row** stationary

End of Instruction-based Processor ?

- Moore's Law, especially Dennard Scaling, is ending.



Data courtesy S. Borkar/Intel 2011

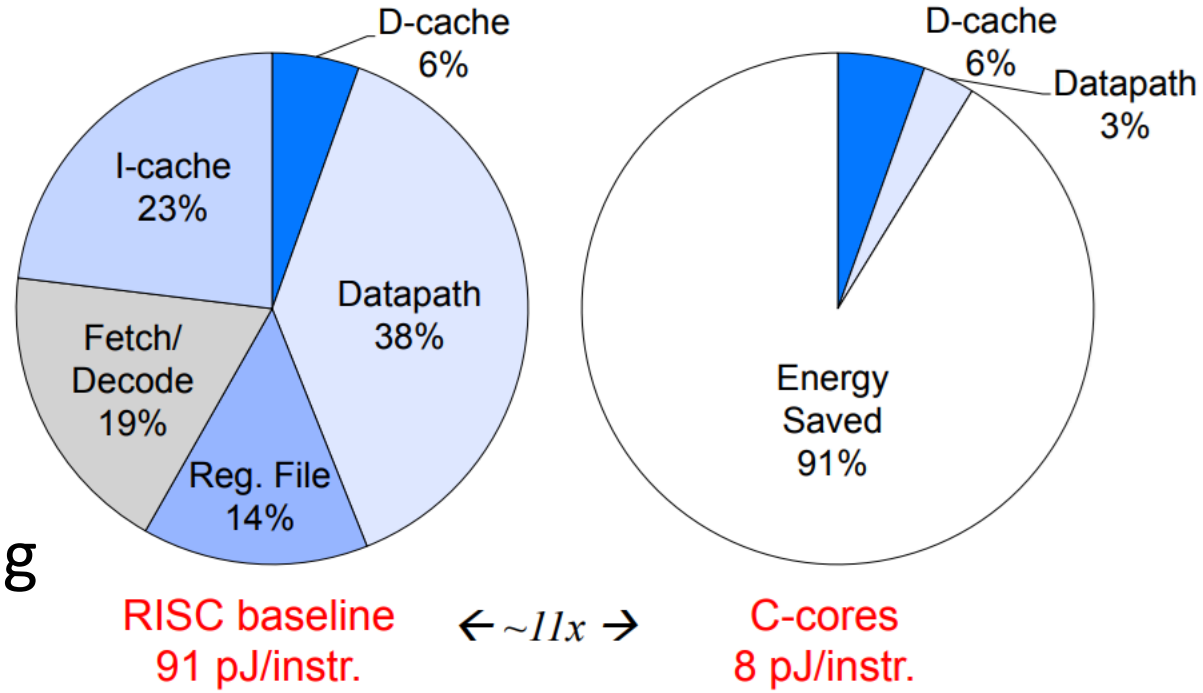


Accelerator without Instruction Arch

- Idea: Leverage dark silicon to “fight” the utilization wall

- Insights:
  - Power is now more expensive than area
  - Specialized logic can improve energy efficiency by 10-1000x

- Accelerator is the basic specializing method

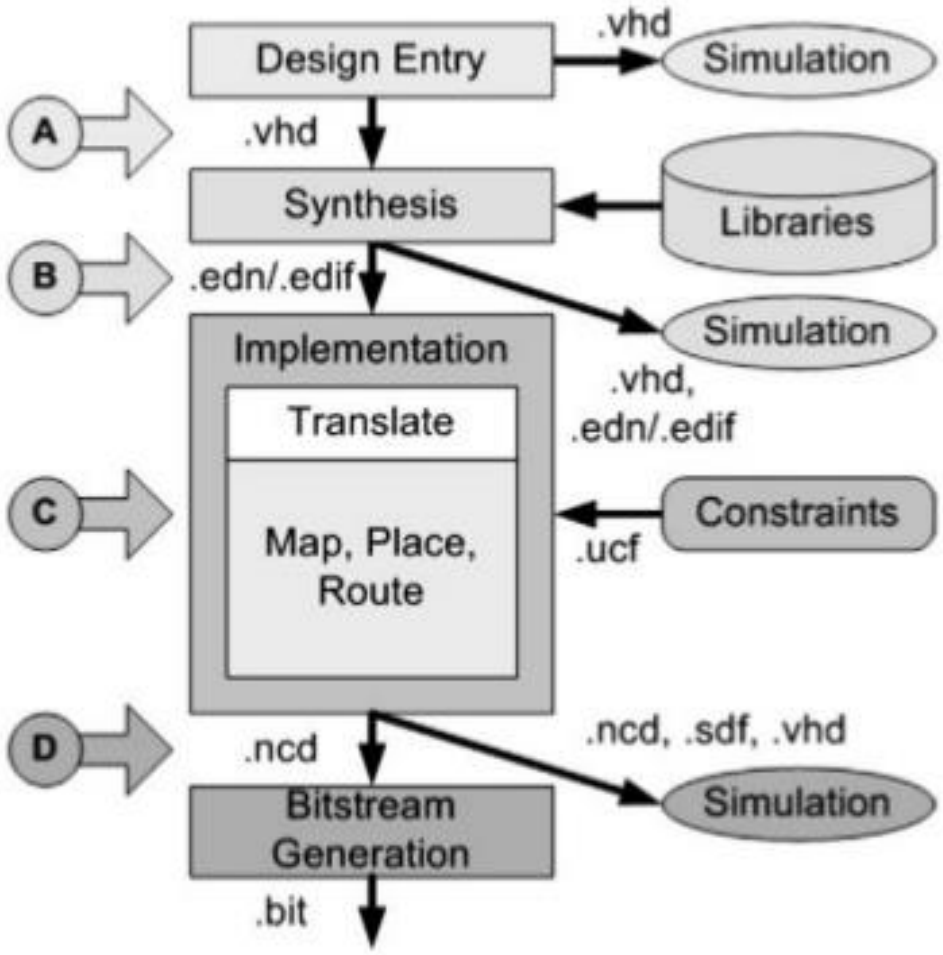


Reconfigurable Computing Devices

FPGA

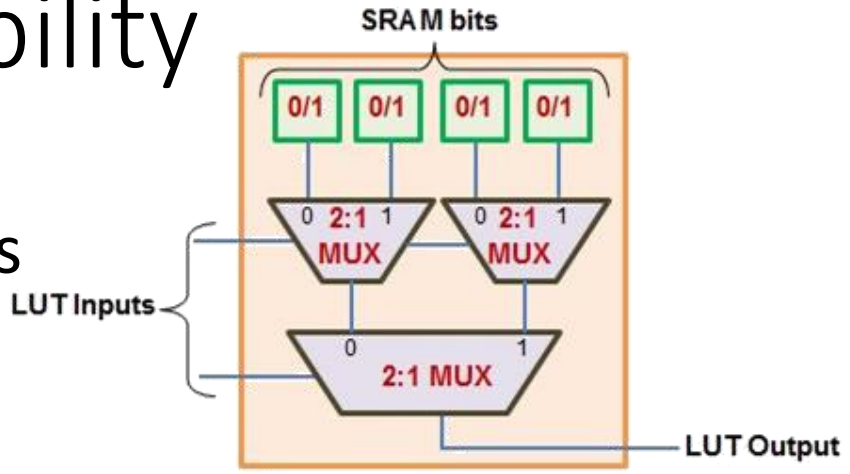
Reconfigurable Computing and Architecture

- From the traditional reconfigurable architecture and hardware
- best known reconfigurable architectures: **FPGA**
  - configuration block: bit level
  - data routing the major challenge
- We also called it fine-grained reconfigurable architectures.



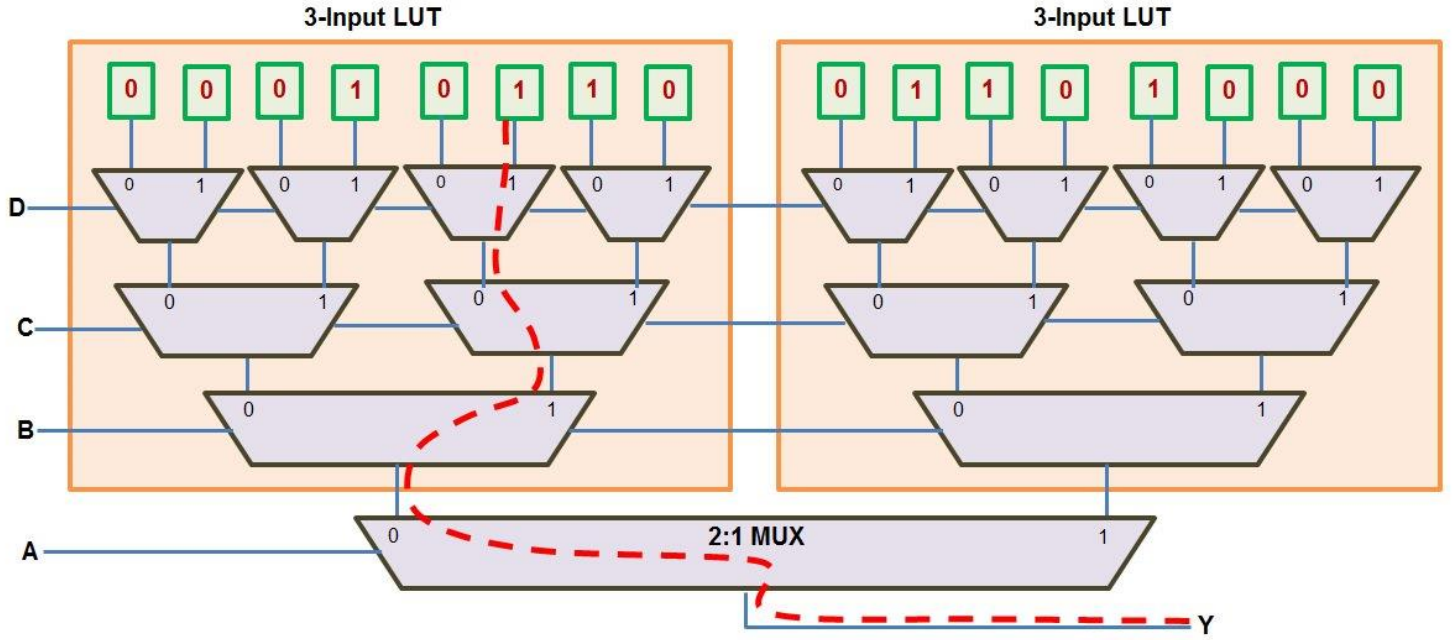
Look-up-table based Configurability

- Basic structure of LUT: SRAM + Switches/ MUX's
- Mapping of "any" logic – generate bit table



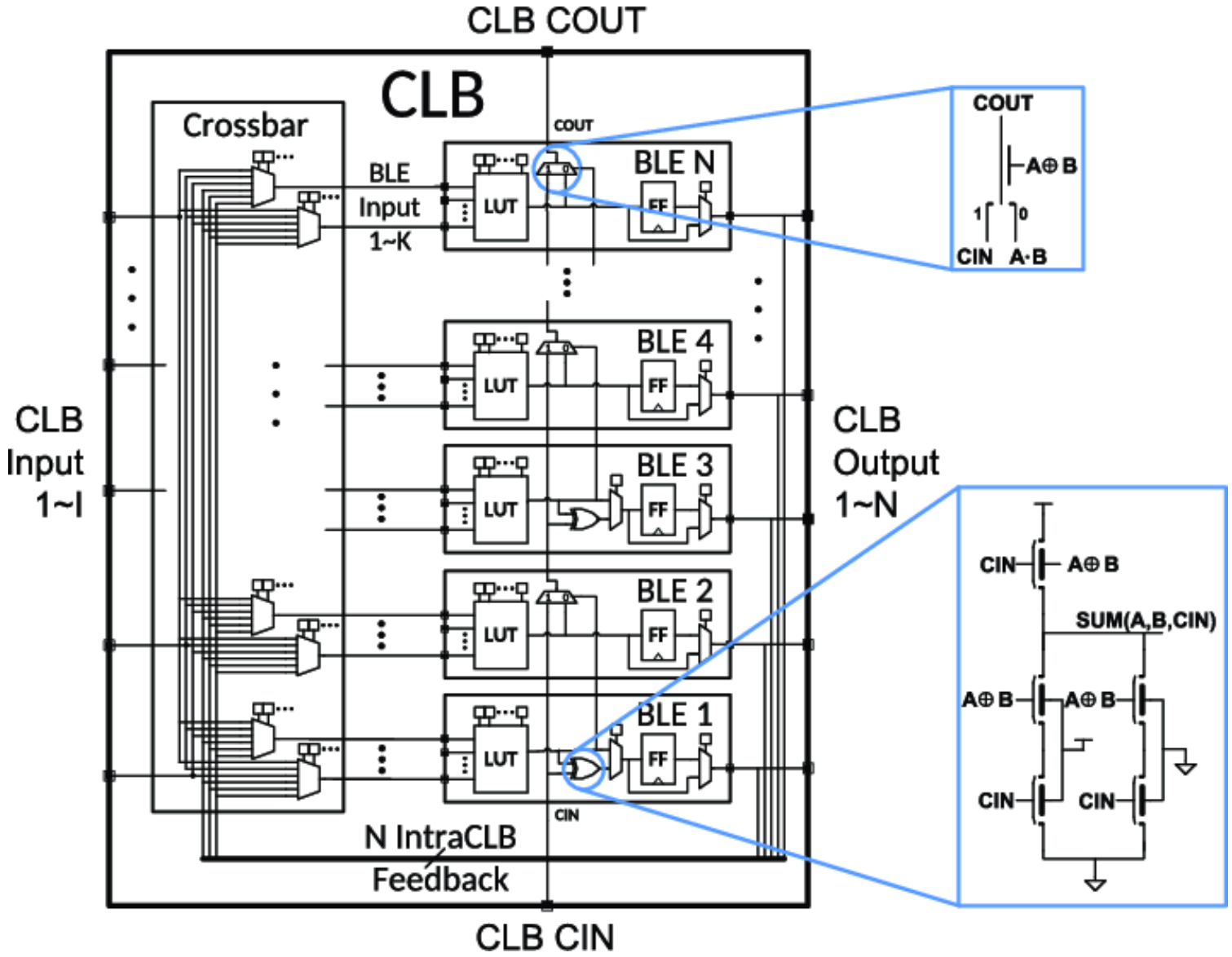
Truth Table

Inputs				Output
A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0



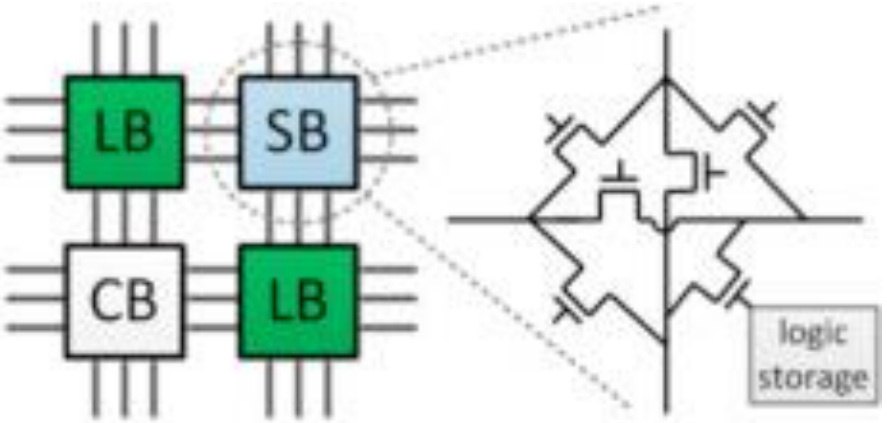
Cluster based element

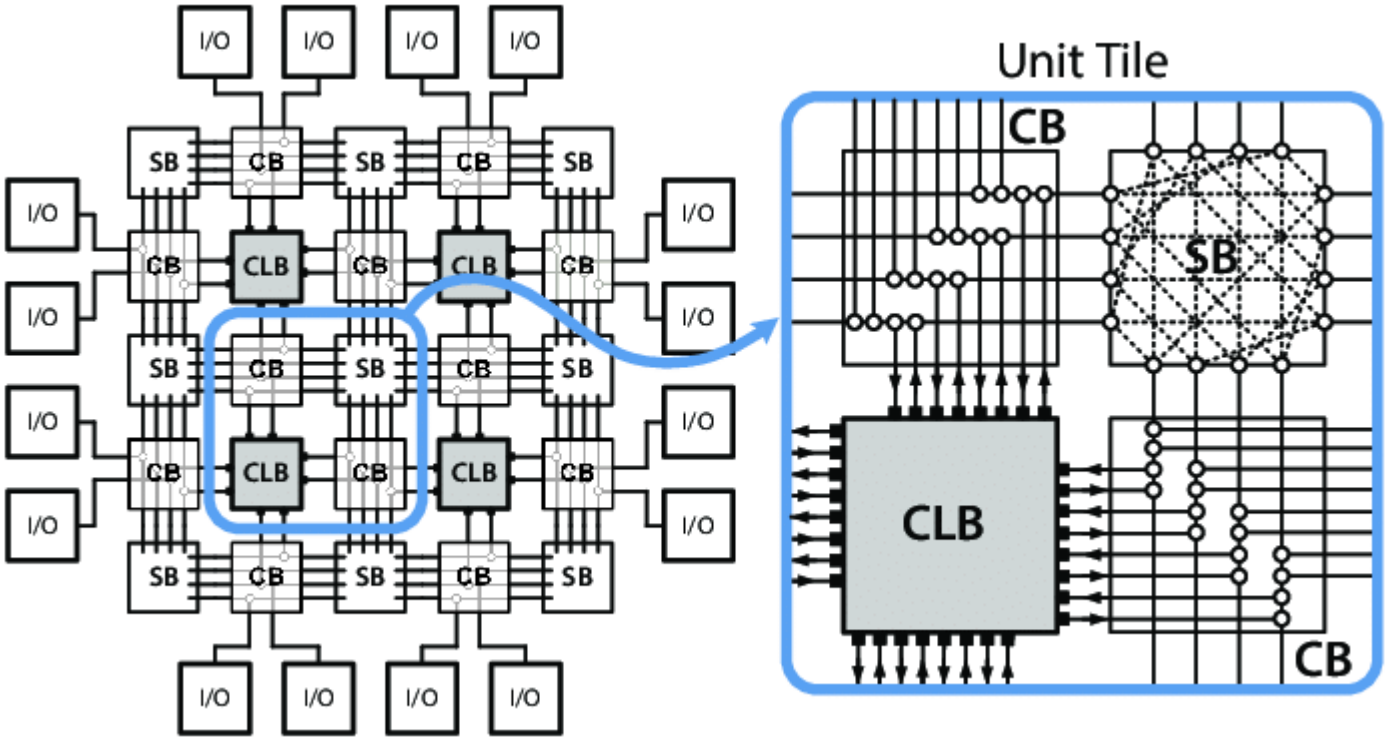
- CLB = Configurable logic block
- Locally, a CLB has crossbar, LUTs DFFs, Carry in/out logic



Global Fabric

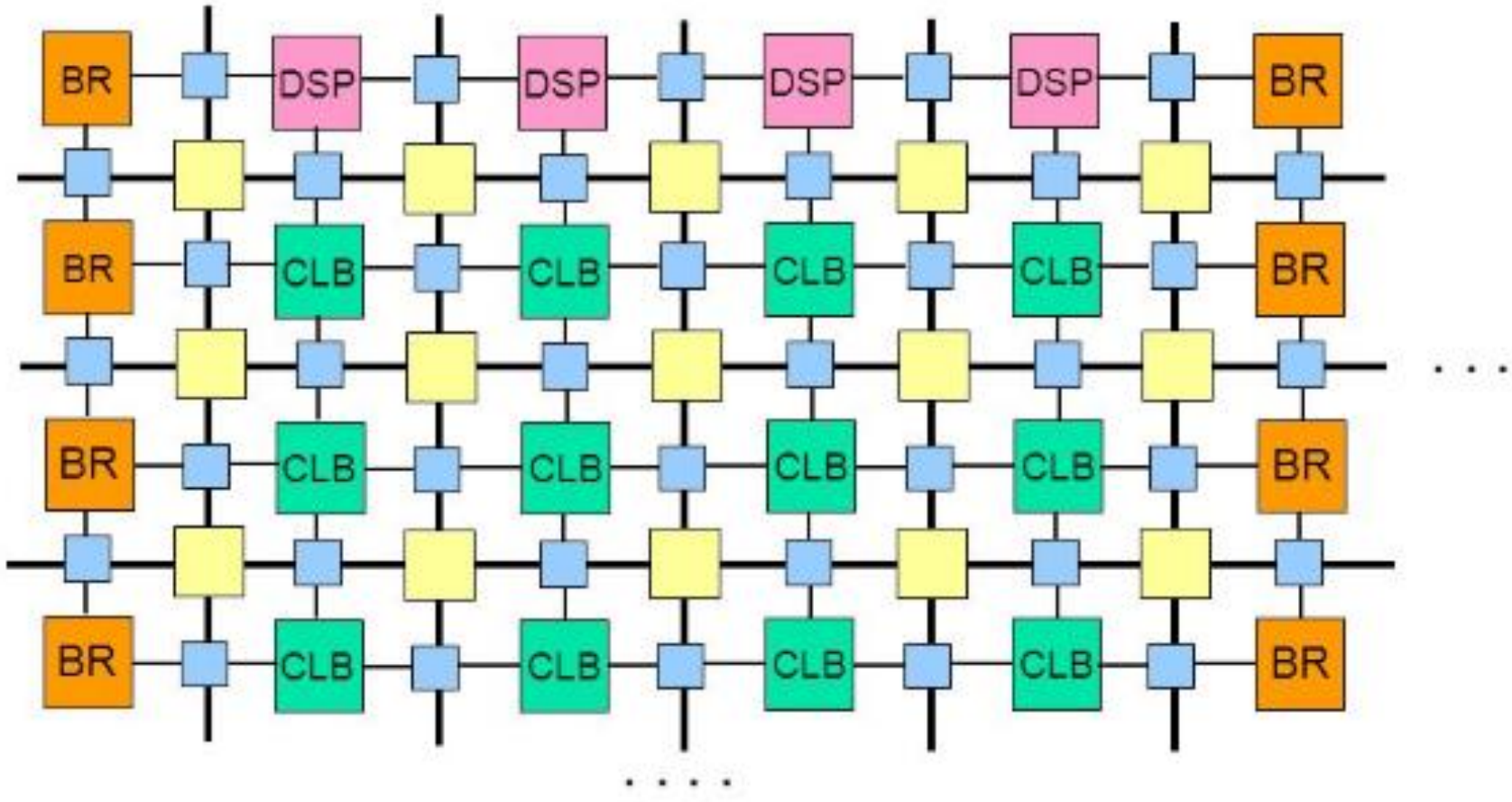
- Interconnection between CLBs:
- CB = connection boxes
- SB = switch boxes





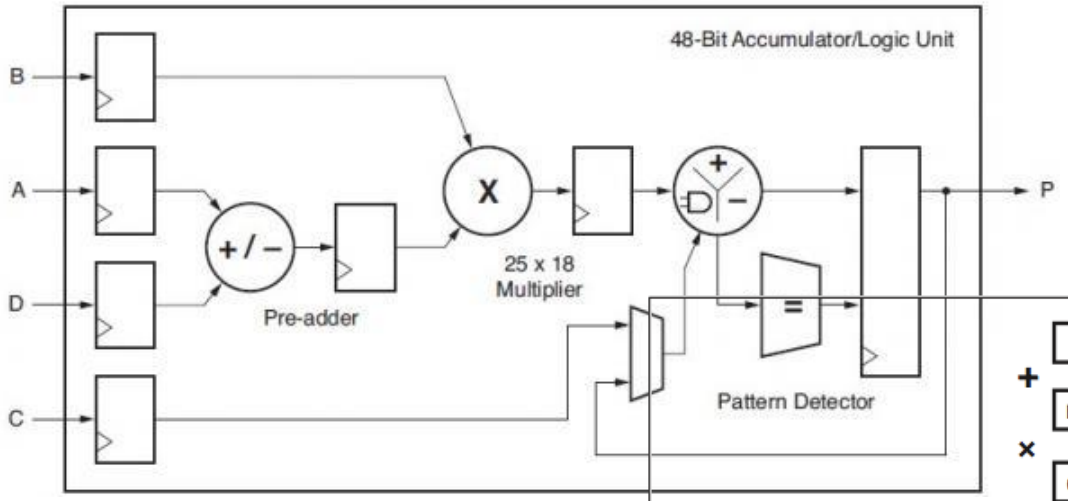
Modern FPGA has something more than CLB

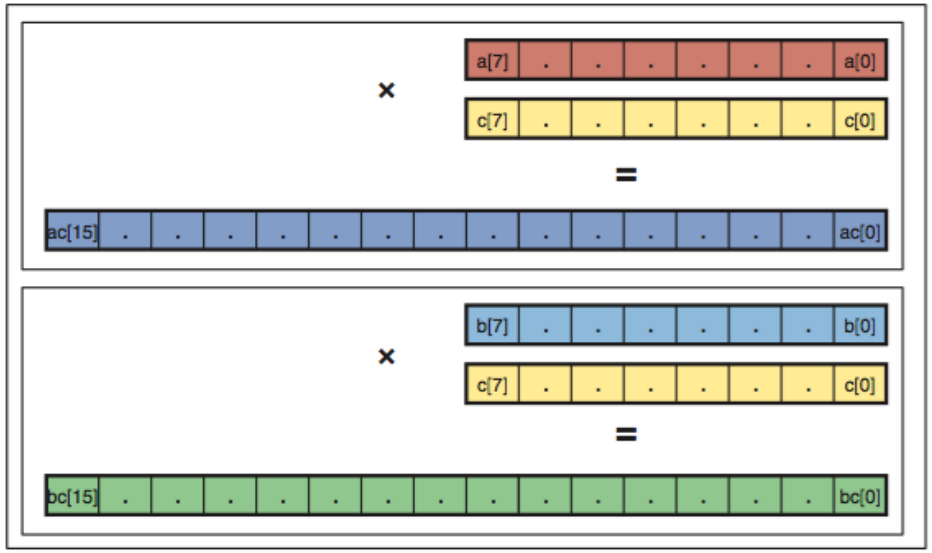
- CLB is not efficient for data storage, need something like SRAM (known as BRAM in FPGA)
- Multiplier and accumulator is also not efficient (DSP)



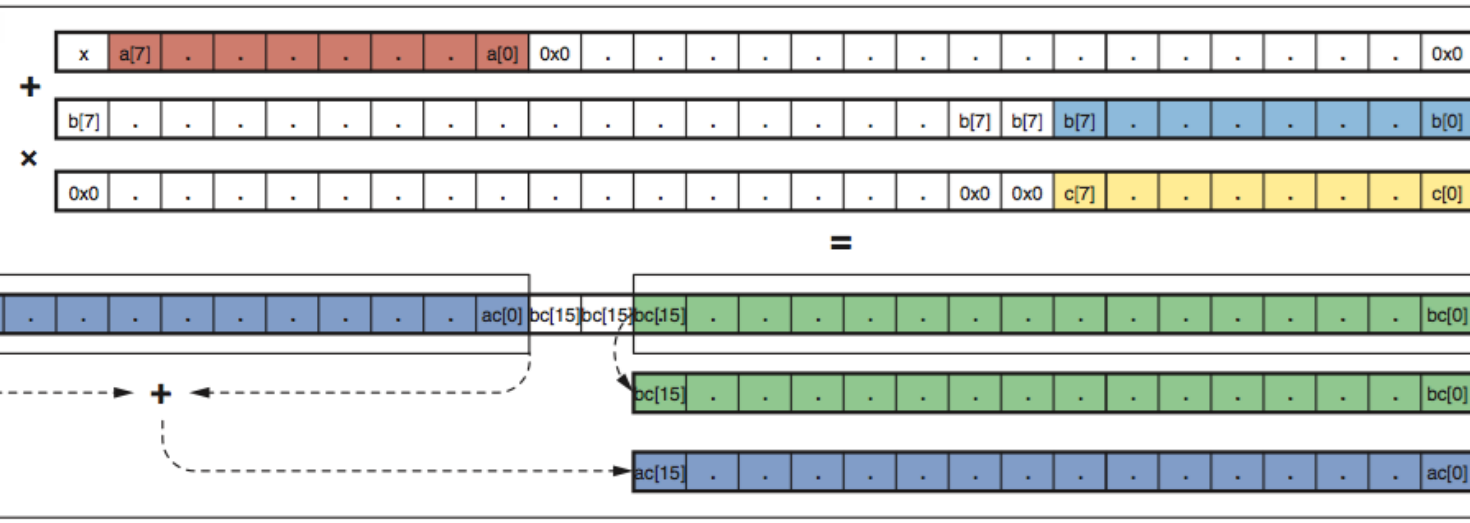
DSP block in FPGA

- DSP48 IP in Xilinx FPGA
- Splitting for Deep Learning





≡



A second thought

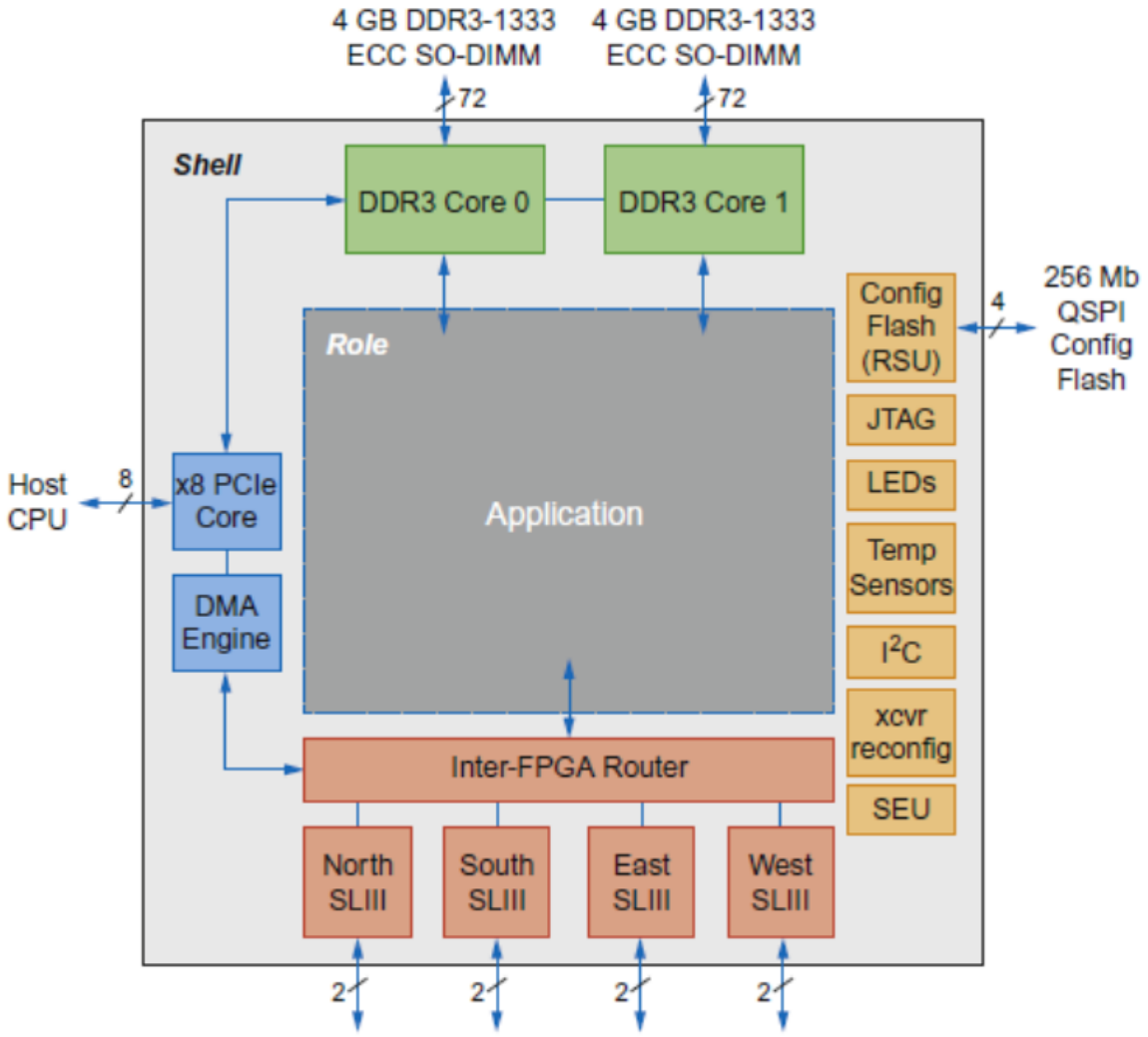
- FPGA using IPs for efficiency, so when we using FPGA, we first want to maximize the IP utilization
- Question: Is FPGA **a real fine-grained** reconfigurable hardware now?

	Kintex UltraScale	Kintex UltraScale+	Virtex UltraScale	Virtex UltraScale+	Zynq UltraScale+
MPSoC Processing System					✓
System Logic Cells (K)	318–1,451	356–1,143	783–5,541	862–3,780	103–1,143
Block Memory (Mb)	12.7–75.9	12.7–34.6	44.3–132.9	23.6–94.5	4.5–34.6
UltraRAM (Mb)		0–36		90–360	0–36
HBM DRAM (GB)				0–8	
DSP (Slices)	768–5,520	1,368–3,528	600–2,880	2,280–12,288	240–3,528

Case study – Microsoft Catapult

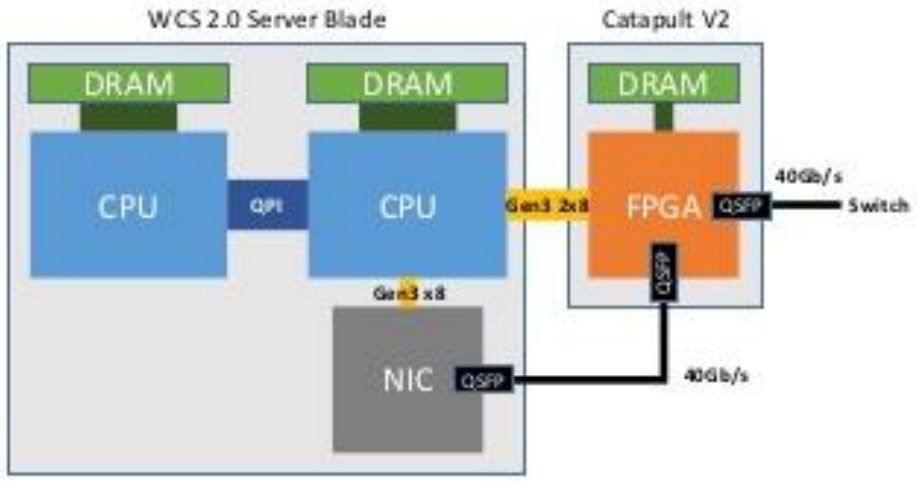
- Microsoft servers are no longer CPU/GPU only based, they are CPU/GPU/FPGA based

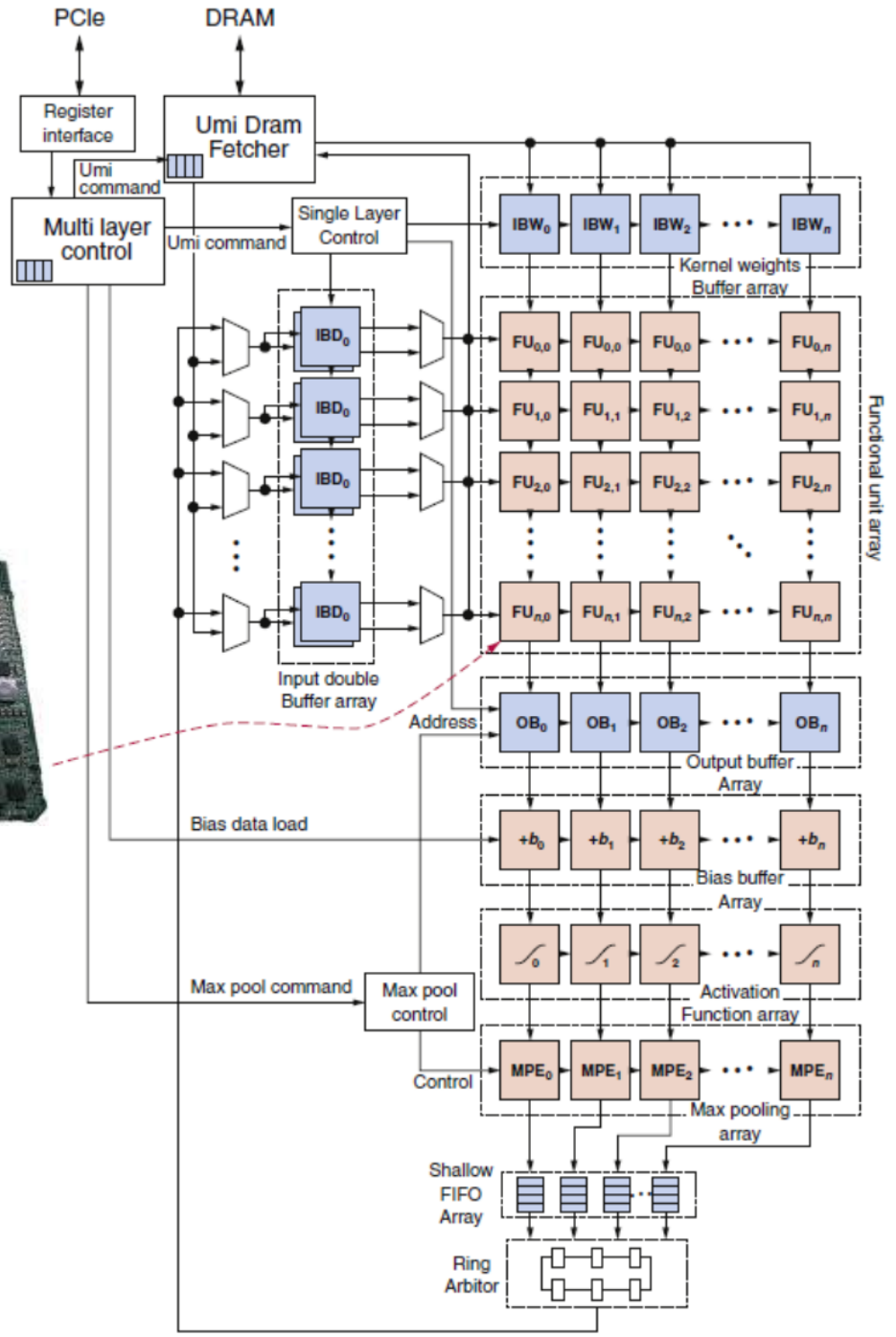
- FPGA is more efficient for domain specific tasks
  - FPGA (unconfigured) has 3962 18-bit ALUs and 5 MiB of on-chip memory
  - Programmed in Verilog RTL
  - Shell is 23% of the FPGA



Catapult for Deep Learning

- CNN accelerators, can be matched to multiple FPGAs

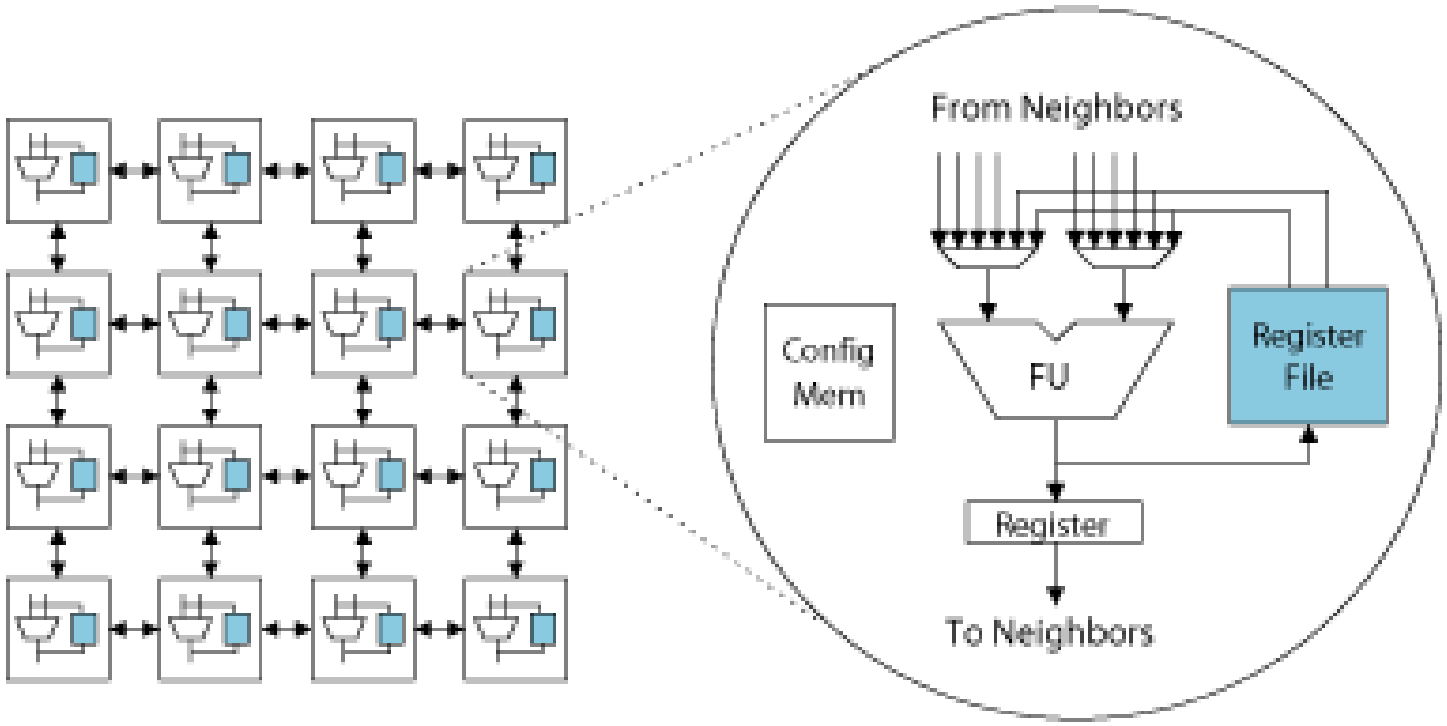




- The architecture justifies the economics
  1. Can act as a local compute accelerator
  2. Can act as a network/storage accelerator
  3. Can act as a remote compute accelerator

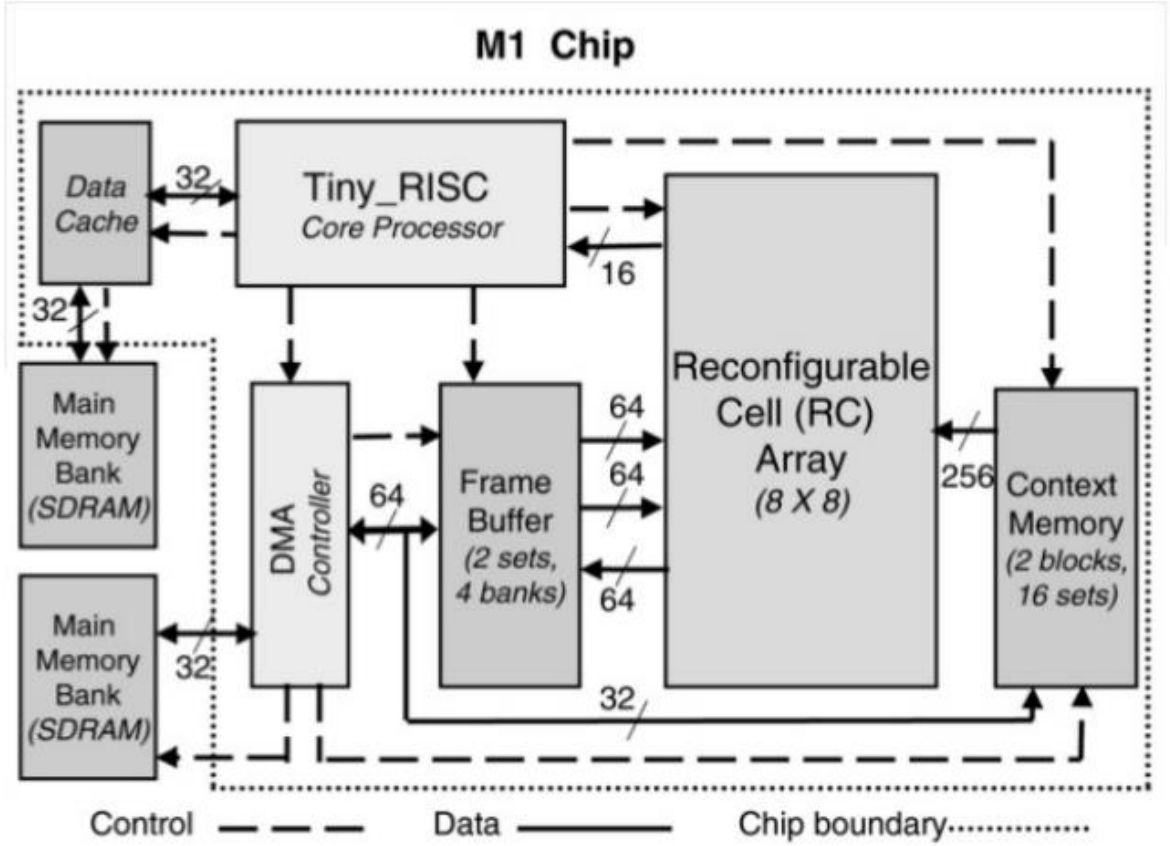
Coarse-Grained Reconfigurable Architecture

- Computing Blocks (PEs) and switched based routing
- Less flexible than fine-grained reconfigurable architectures
- But easier to program
- Fast reconfiguration

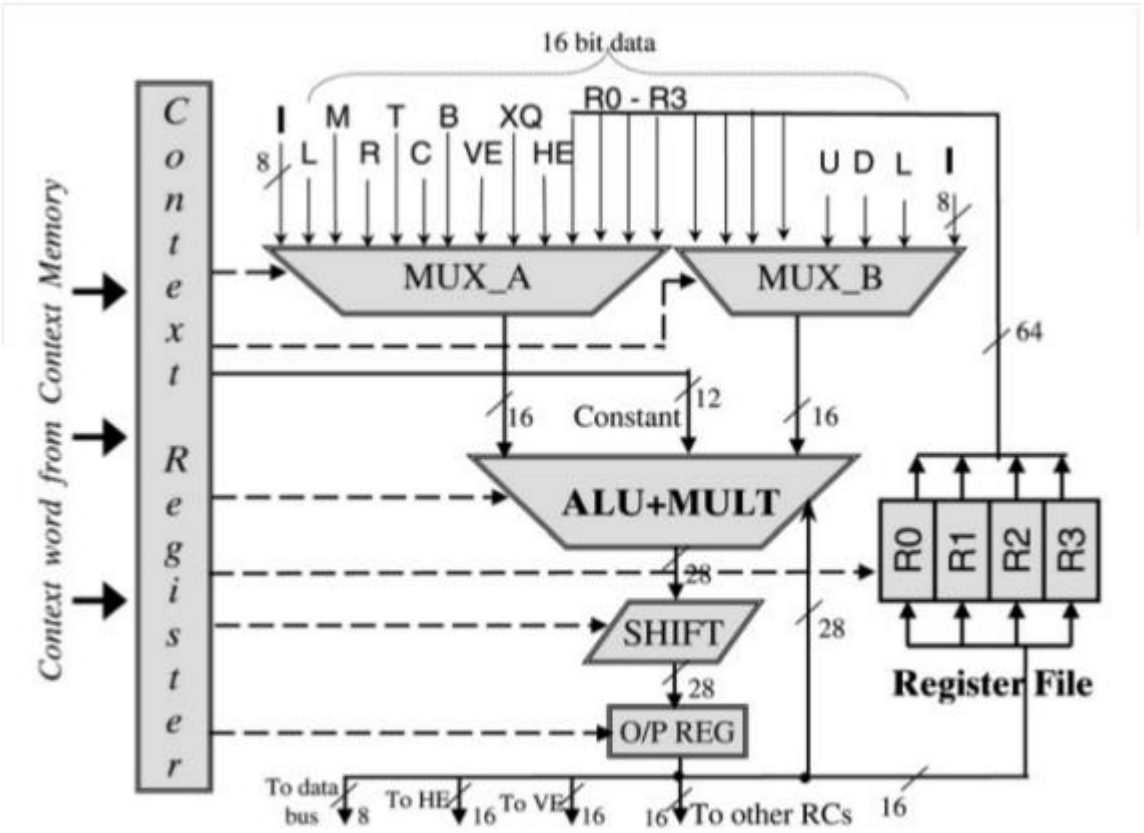


MorphoSys (Singh et al 2000)

- Overall Architecture

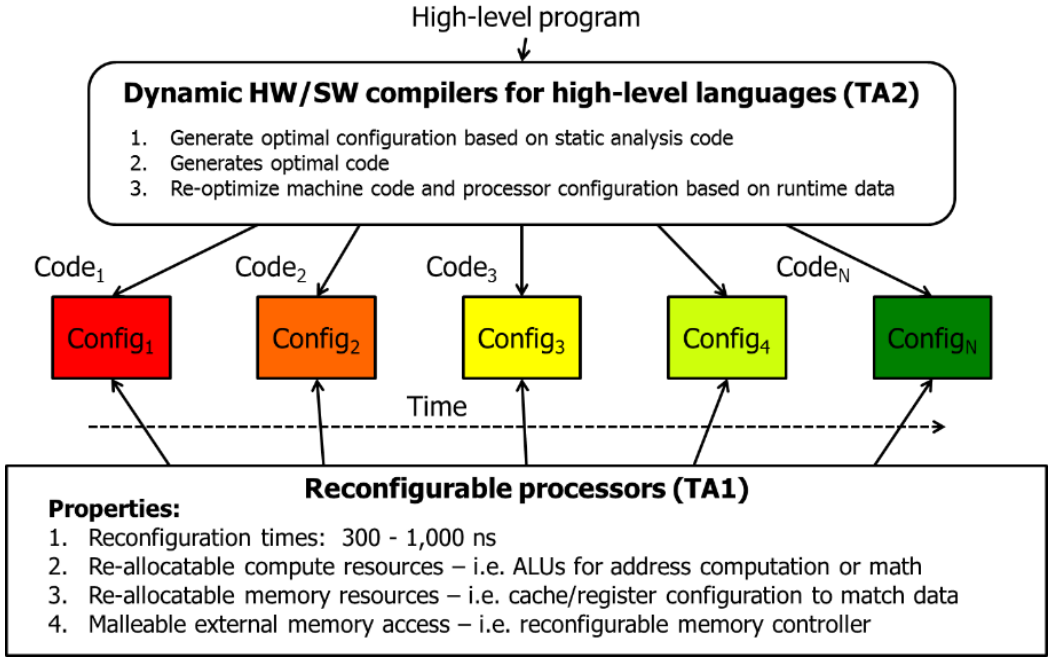


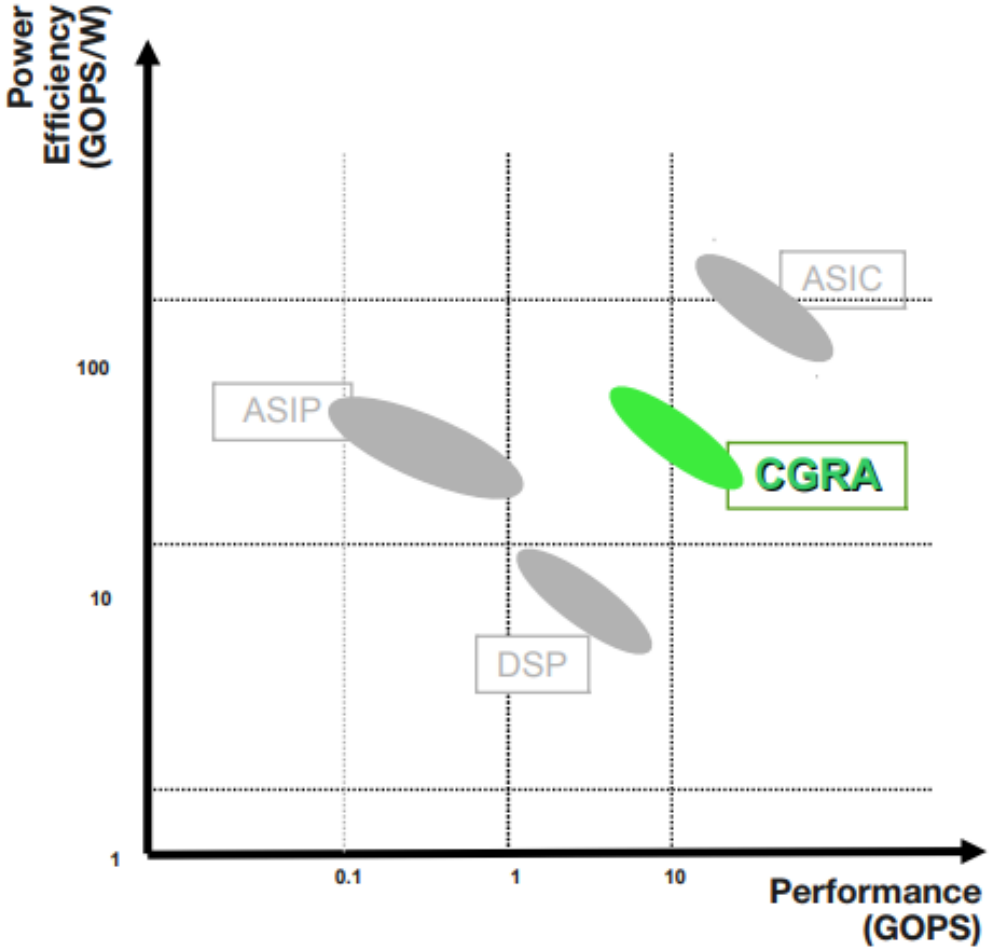
- Inside each Reconfigurable cell



In history

- CGRA was once a specific category for high performance design, but it never succeed
- But CGRA people never give up.



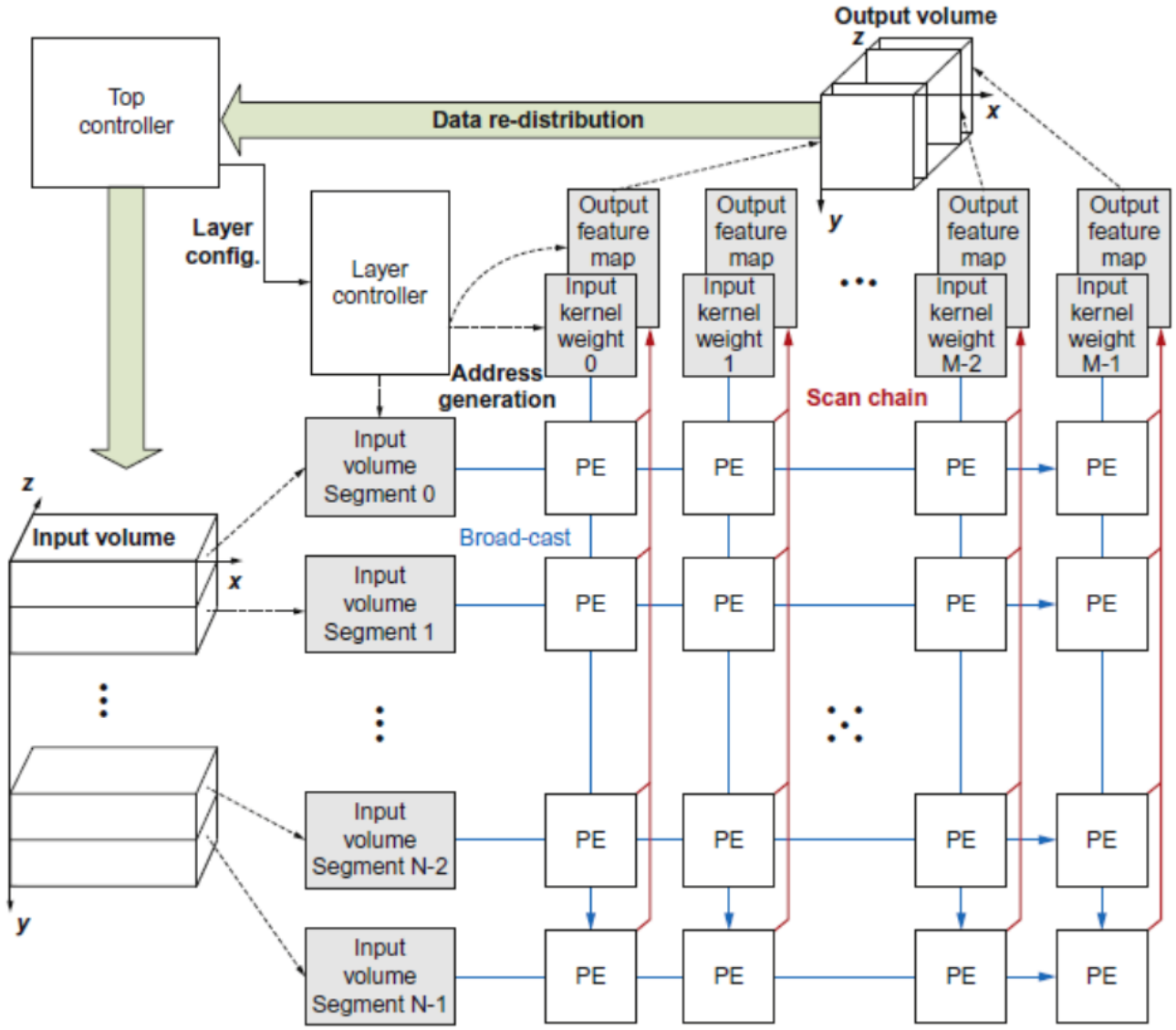


DARPA Software defined hardware Project in 2017

CGRA / FPGA Today

- CGRA as a specific chip category no longer exists
- BUT MOST FPGA implementation, even ASIC, design using CGRA ideas (Thinker from 清华)

Catapult view with PEs →



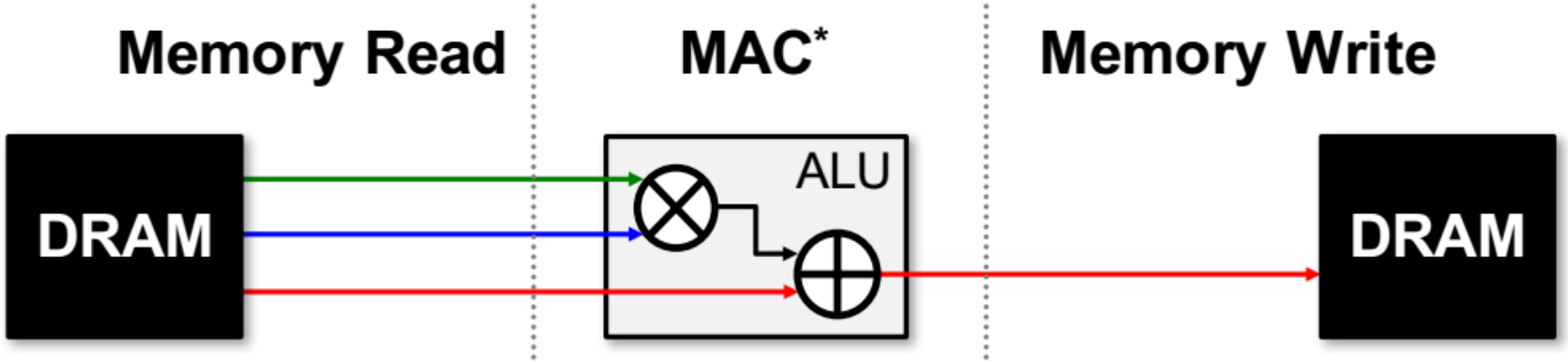
Data Flows

Routing among PE arrays

Memory Access is the Bottleneck

Assembly code for MAC

```
ldh r3, [r0], #2   ; load input
ldh r4, [r1], #2   ; load input
mul r5, r4, r3     ; multiply
add r2, r2, r5     ; accumulate
```

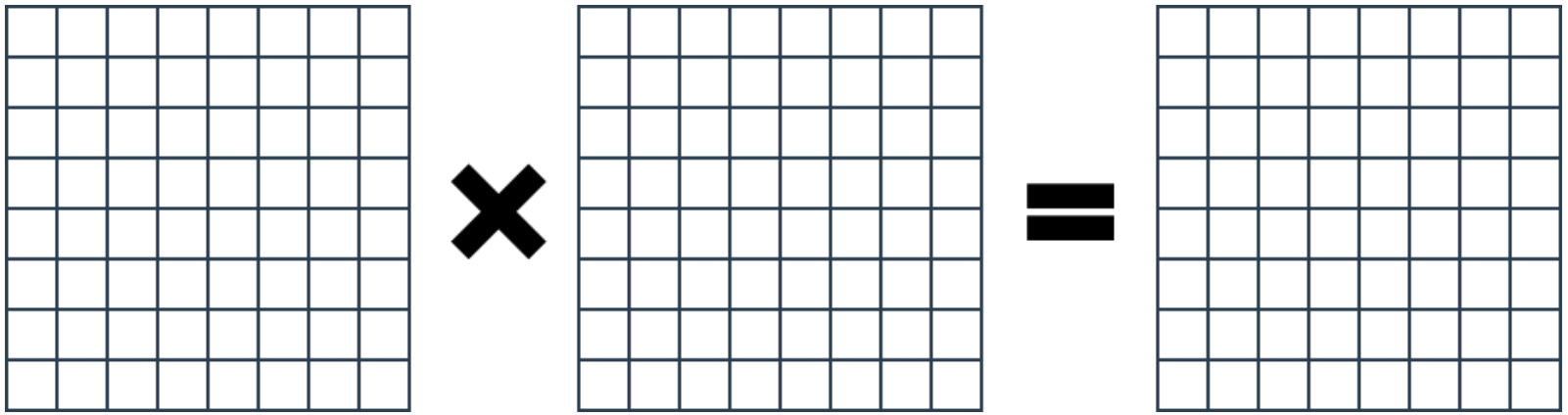


* multiply-and-accumulate

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

- Example: AlexNet [NIPS 2012] has **724M** MACs
  → **2896M** DRAM accesses required

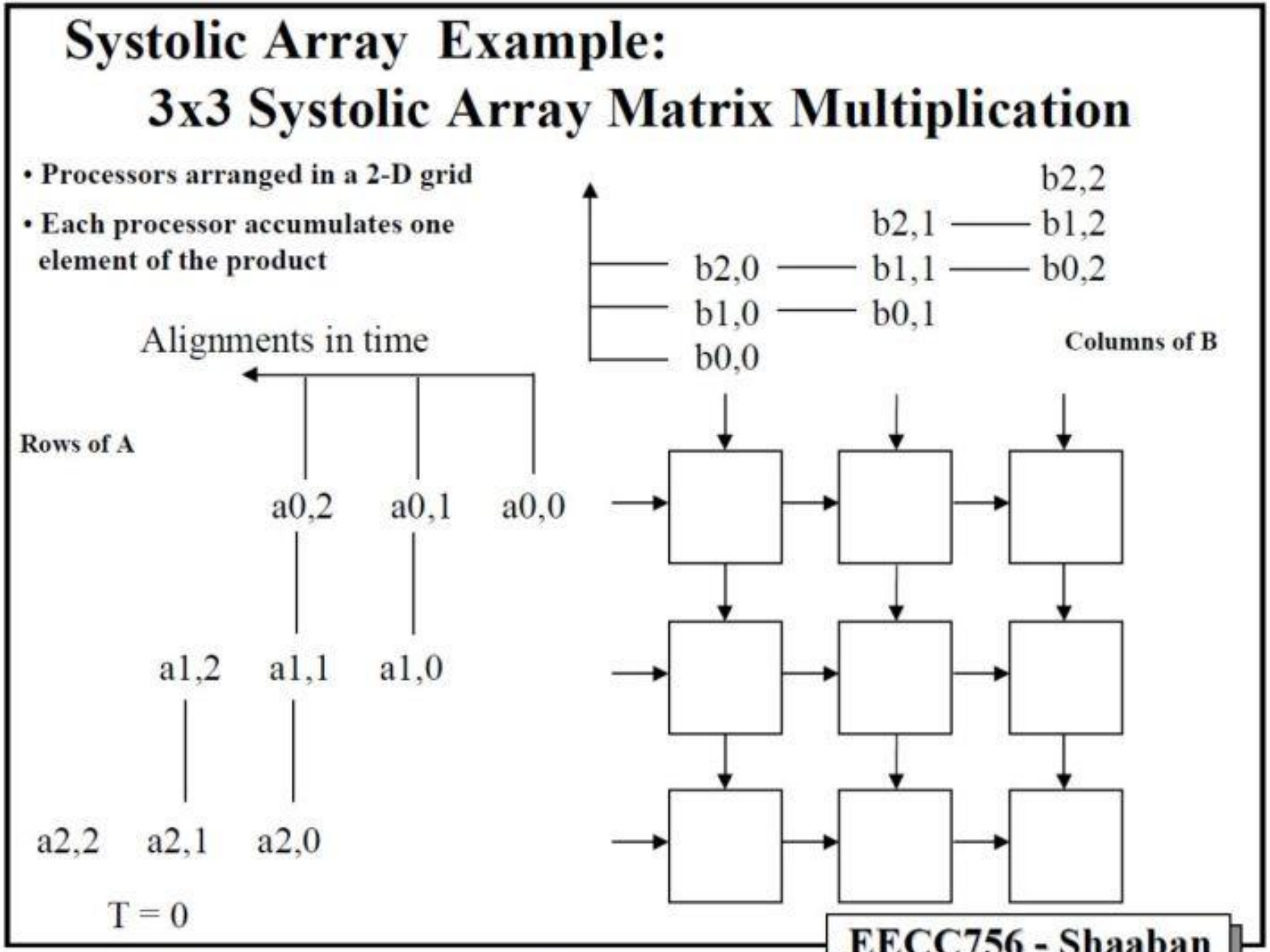
Do we really need so many memory access?



- Assume we have two M x M matrix
- Questions: How many scalar multiplication we need?
  How many data we really need to read/write from/to memory?

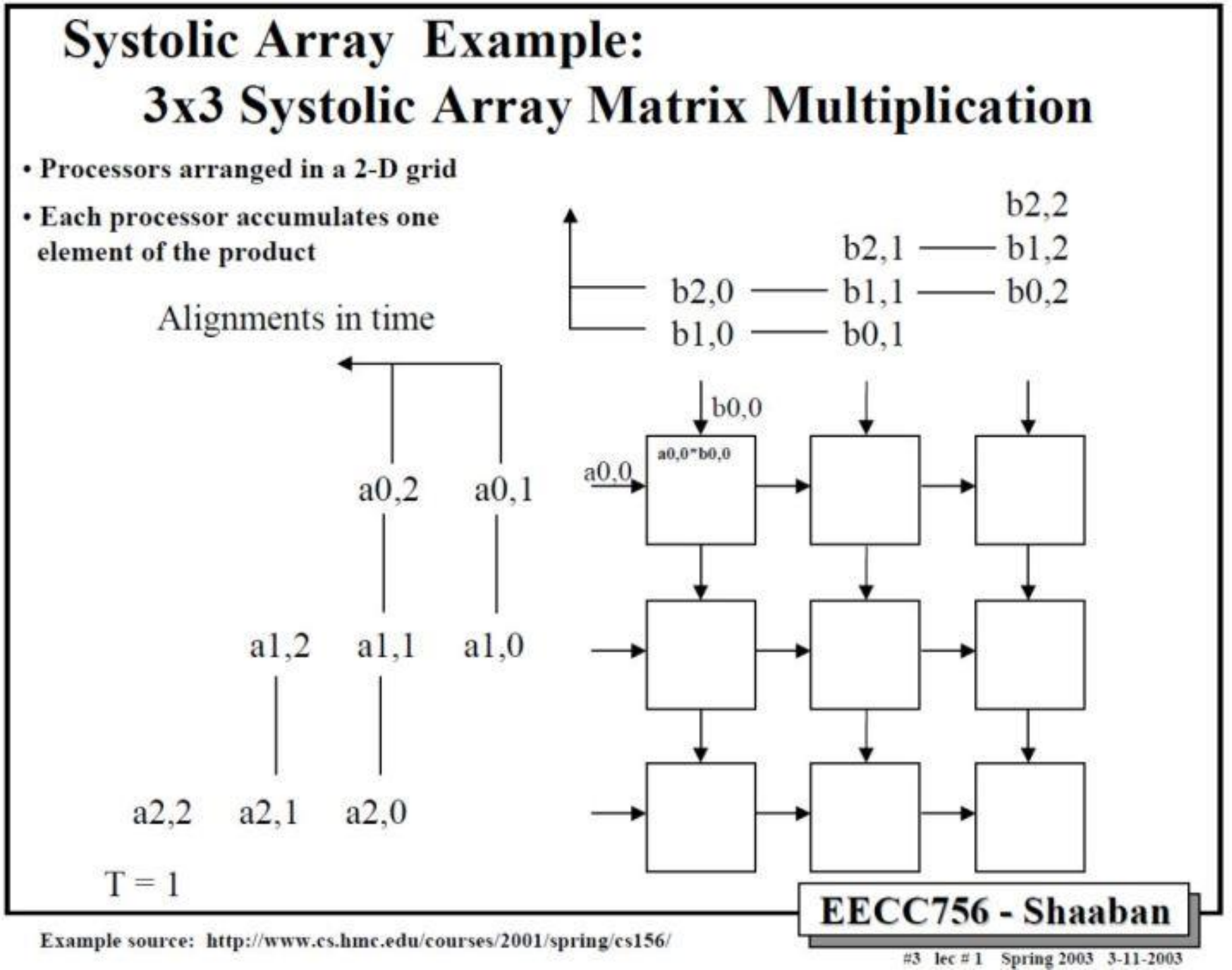
Data flow Example for Matrix Multiplication With multiple PEs (CGRA)

Systolic Array



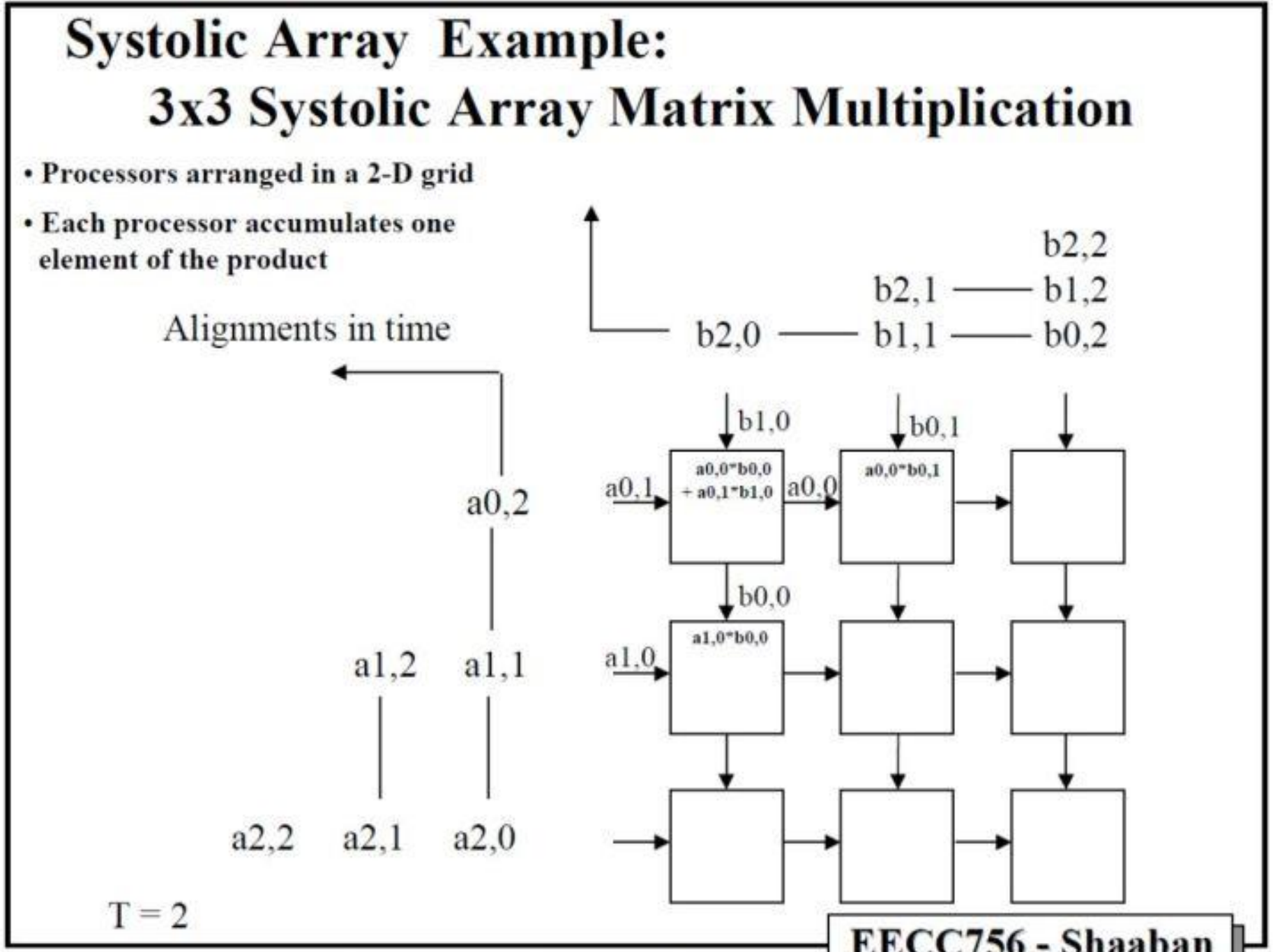
Data flow Example for Matrix Multiplication With multiple PEs (CGRA)

Systolic Array



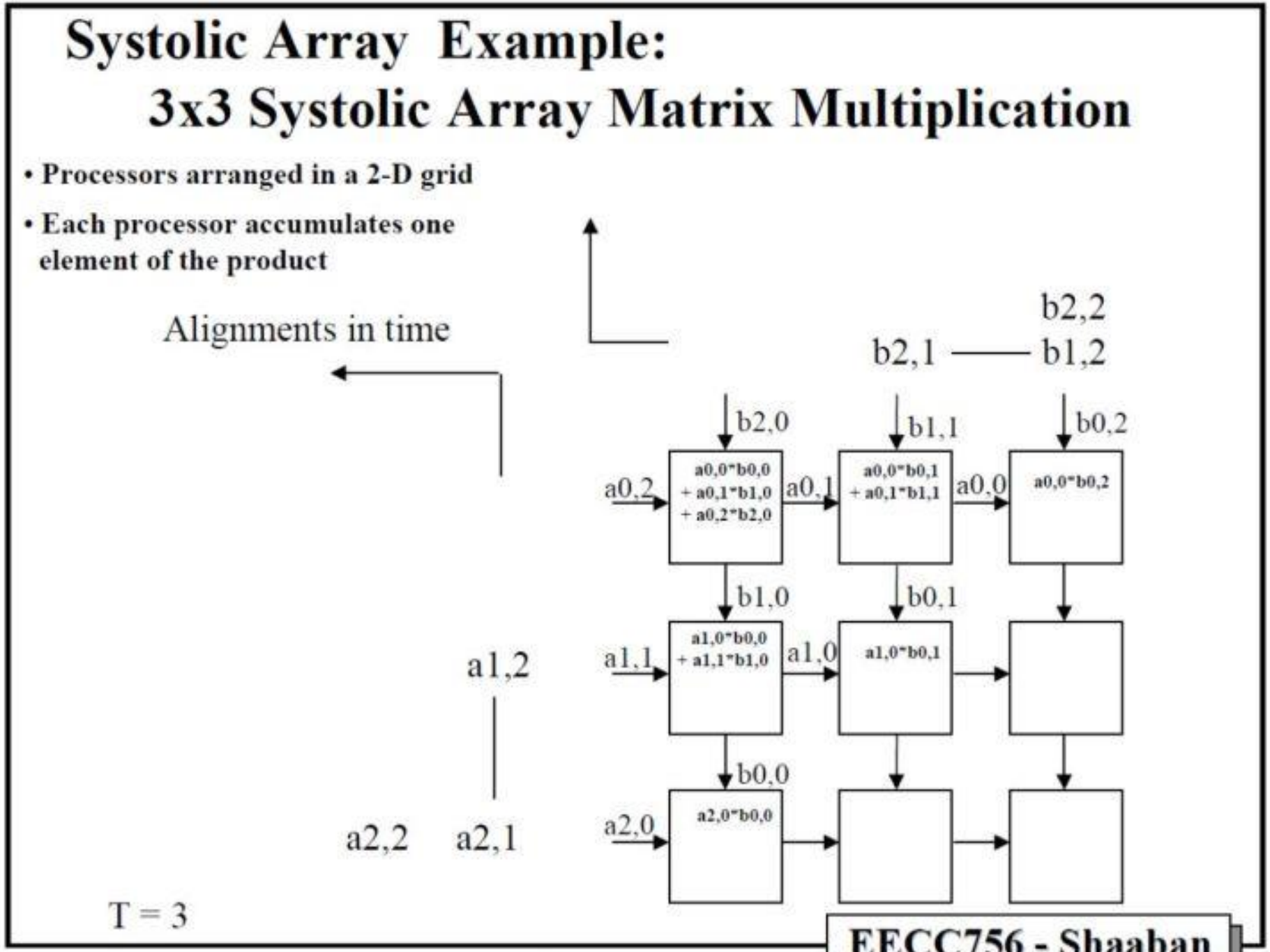
Data flow Example for Matrix Multiplication With multiple PEs (CGRA)

Systolic Array



Data flow Example for Matrix Multiplication With multiple PEs (CGRA)

Systolic Array

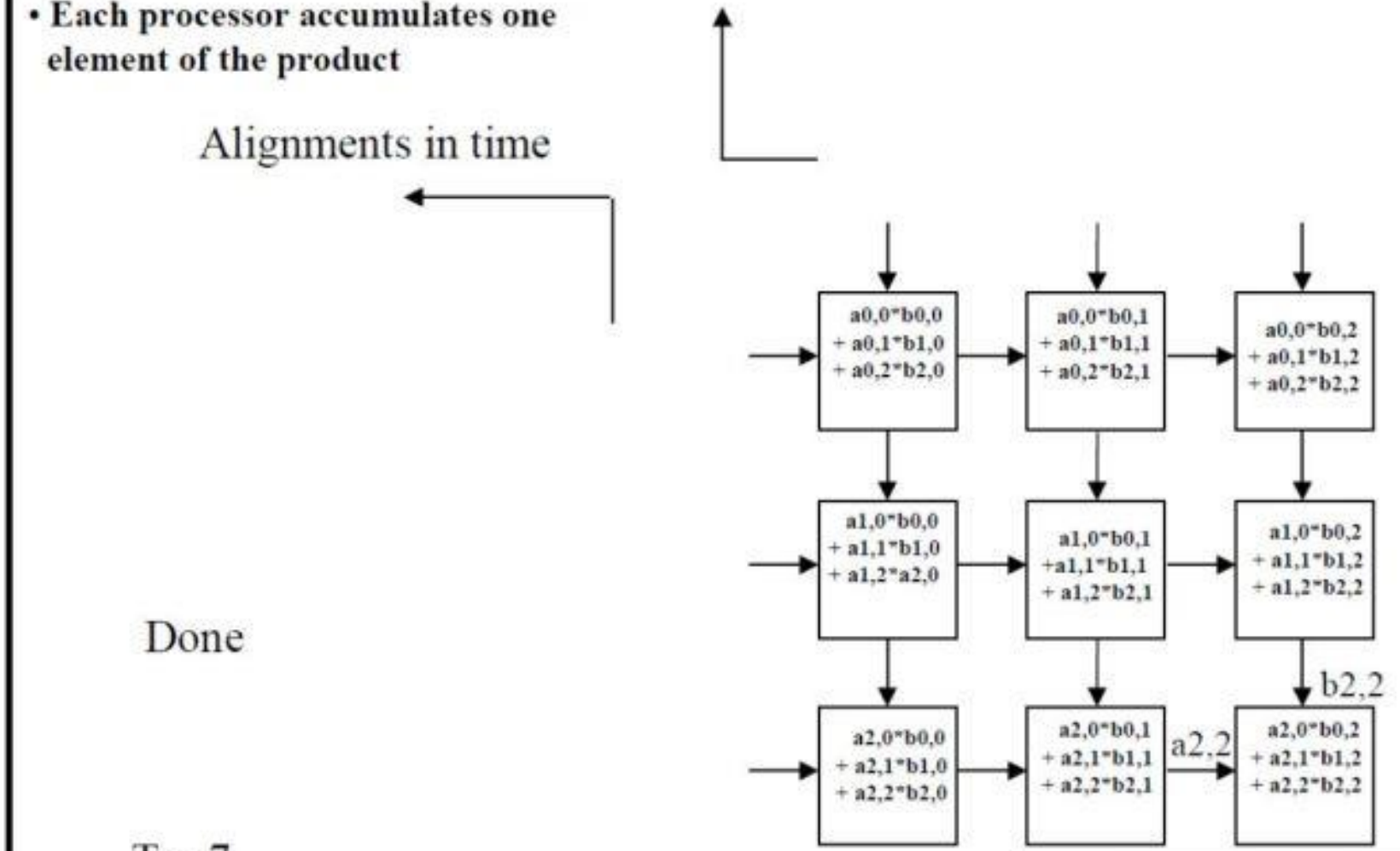


Data flow Example for Matrix Multiplication With multiple PEs (CGRA)

Systolic Array

Systolic Array Example:
3x3 Systolic Array Matrix Multiplication

- Processors arranged in a 2-D grid
- Each processor accumulates one element of the product

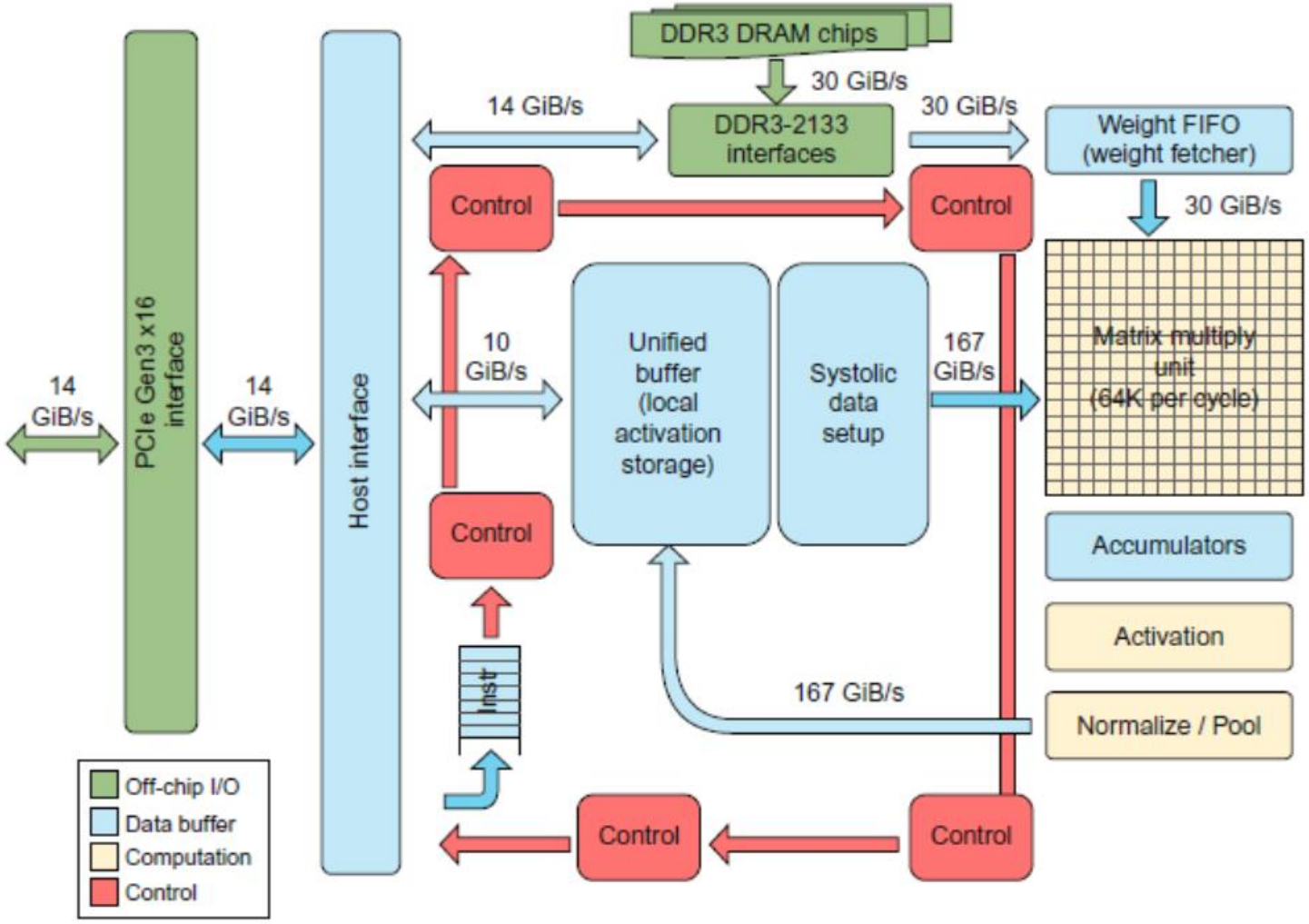


EECC756 - Shaaban

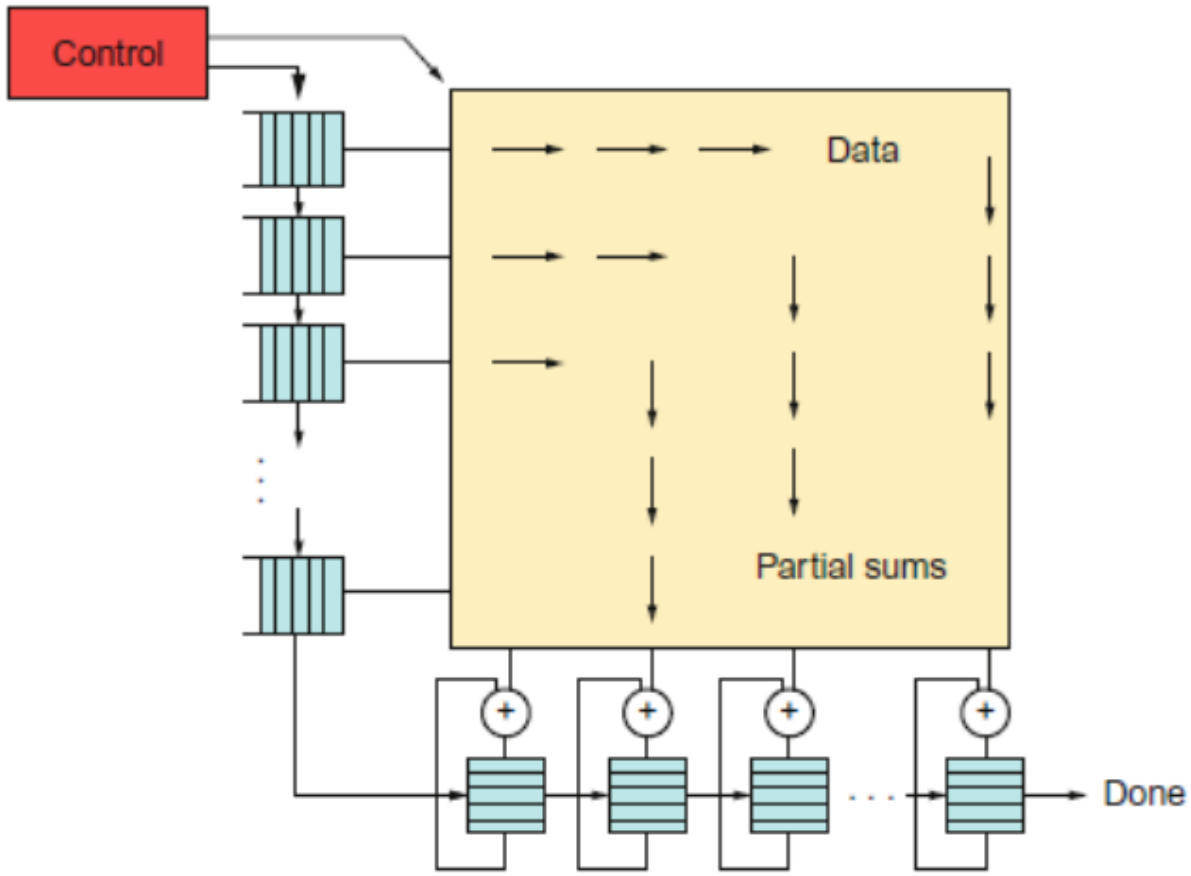
Example source: http://www.cs.hmc.edu/courses/2001/spring/cs156/

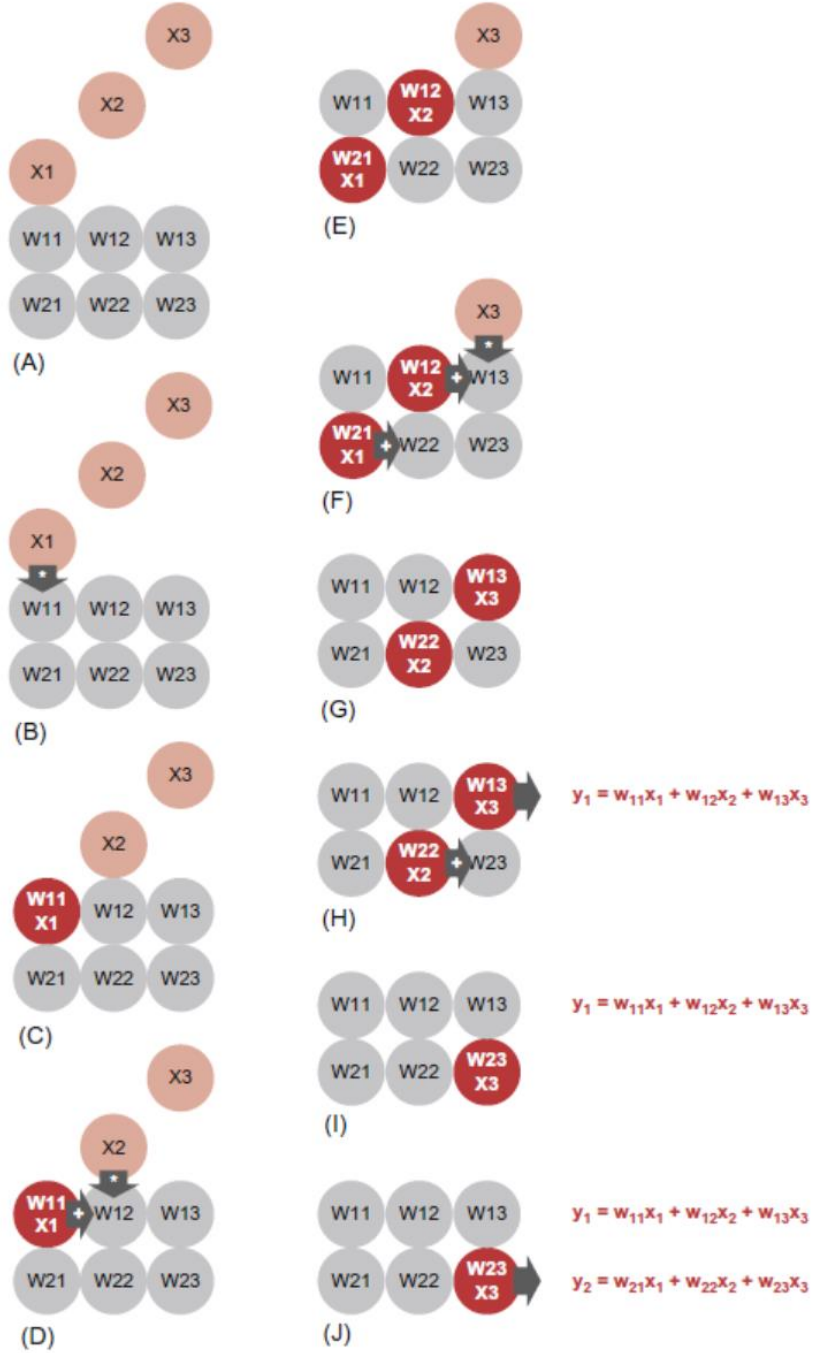
Case Study: TPU using systolic array

- Google's DNN ASIC
- 256 x 256 8-bit matrix-multiply unit
- Large software-managed scratchpad
- Coprocessor on the PCIe bus



TPU Operation





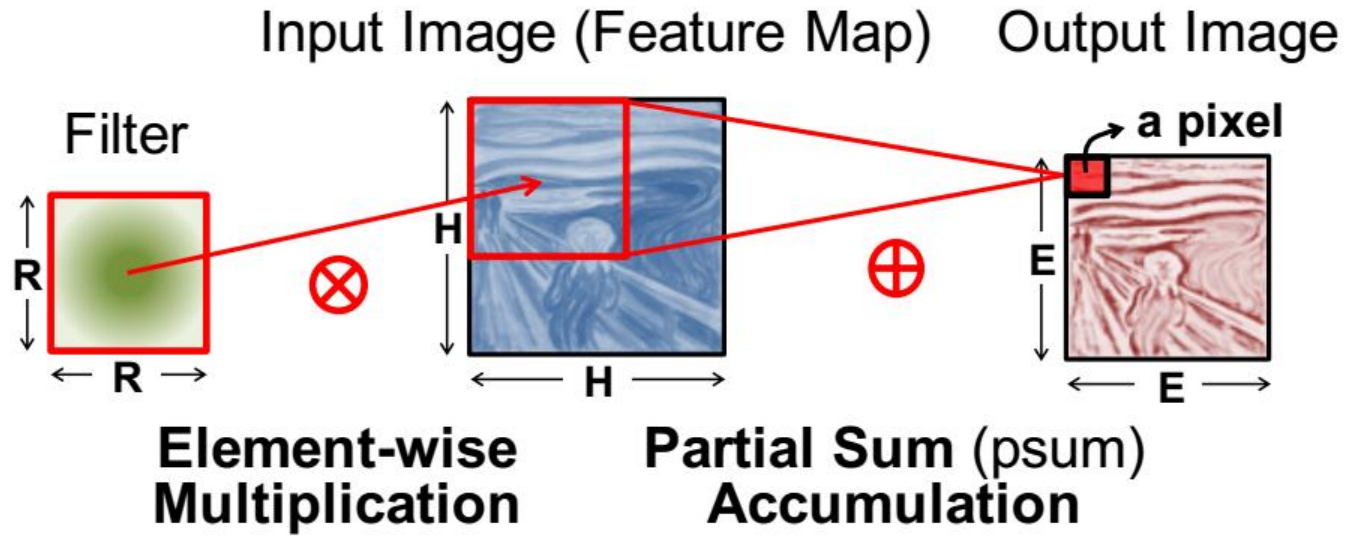
Ref: Google TPU form ISCA 2017

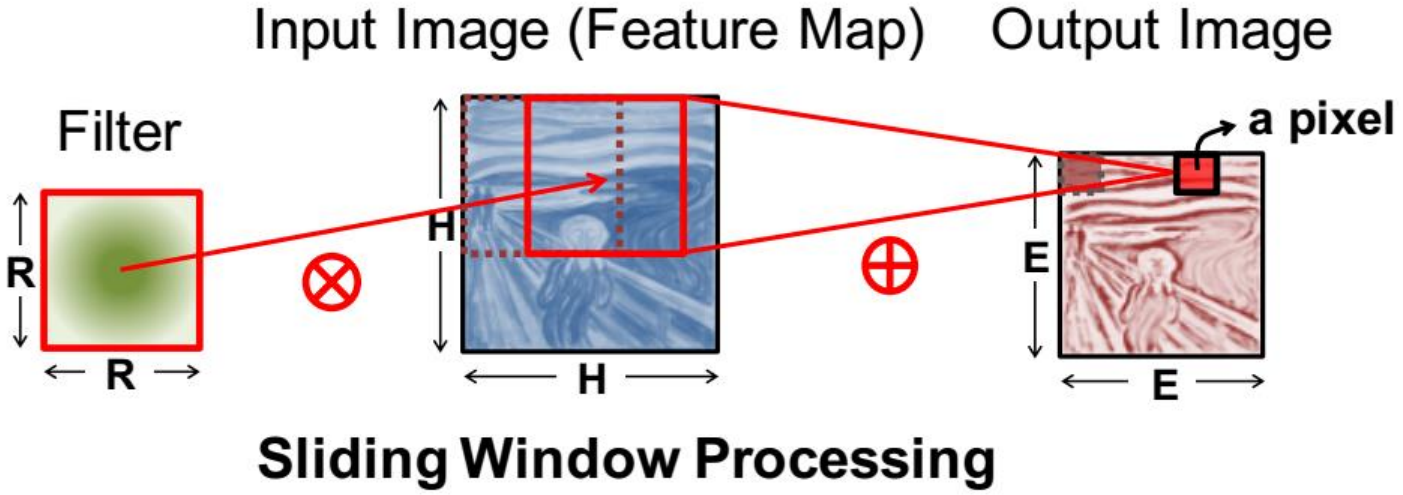
CNN

More chance of Bandwidth saving

1) Matrix Multiplication

2) Sliding window



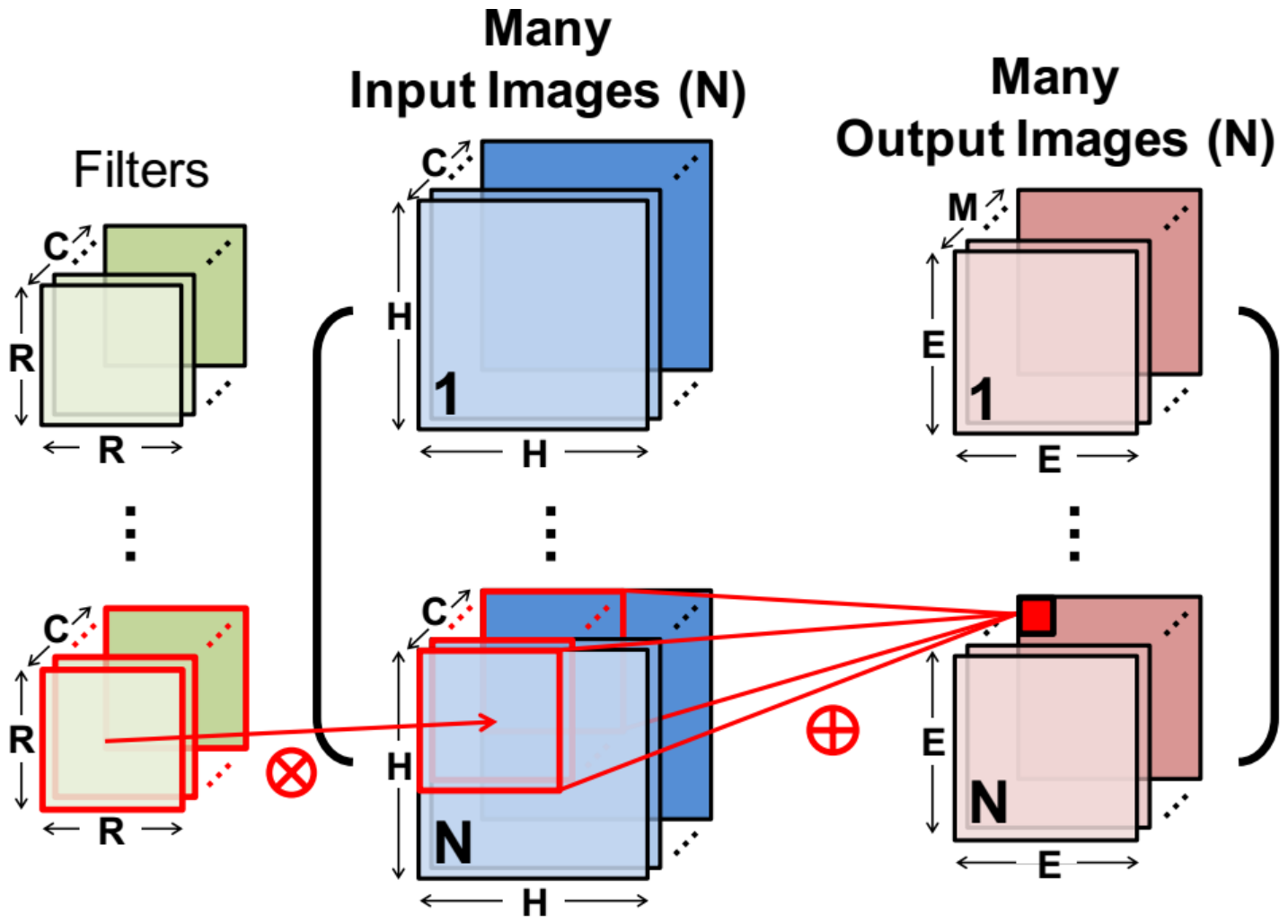


High Dimensional CNN

More chance of Bandwidth saving

3) Filter sharing

4) Feature sharing

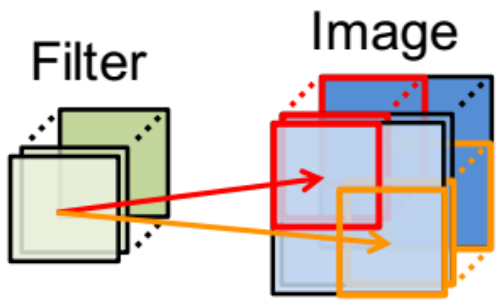


Types of Data Reuse in CNN

Data Reuse relaxes the bandwidth requirement

Convolutional Reuse

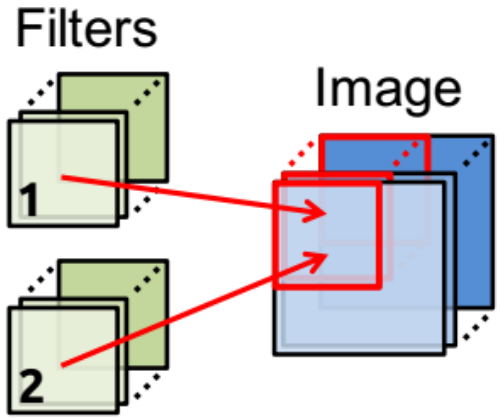
CONV layers only
(sliding window)



Reuse: Image pixels
Filter weights

Image Reuse

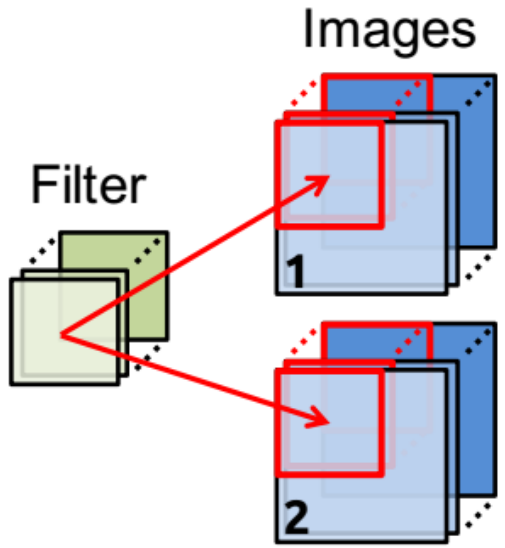
CONV and FC layers



Reuse: Image pixels

Filter Reuse

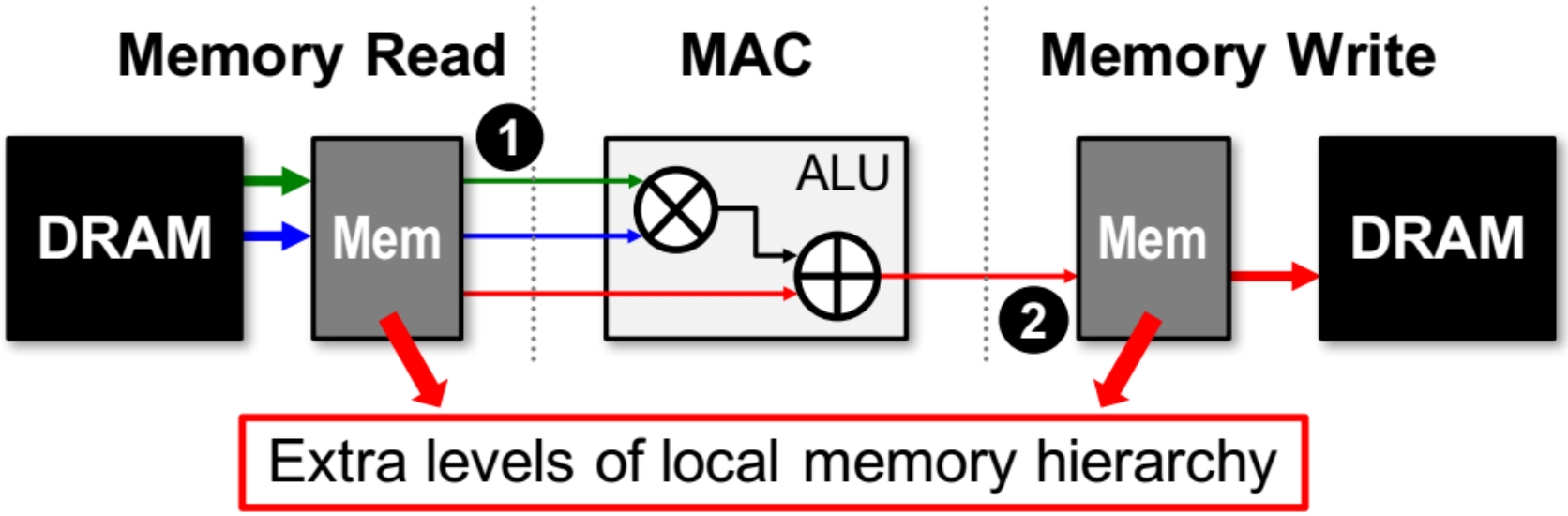
CONV and FC layers
(batch size > 1)



Reuse: Filter weights

Data Reuse relaxes the bandwidth requirement

Memory Access is the Bottleneck



Opportunities: ❶ **data reuse** ❷ **local accumulation**

❶ Can reduce DRAM reads of filter/image by up to **500×**

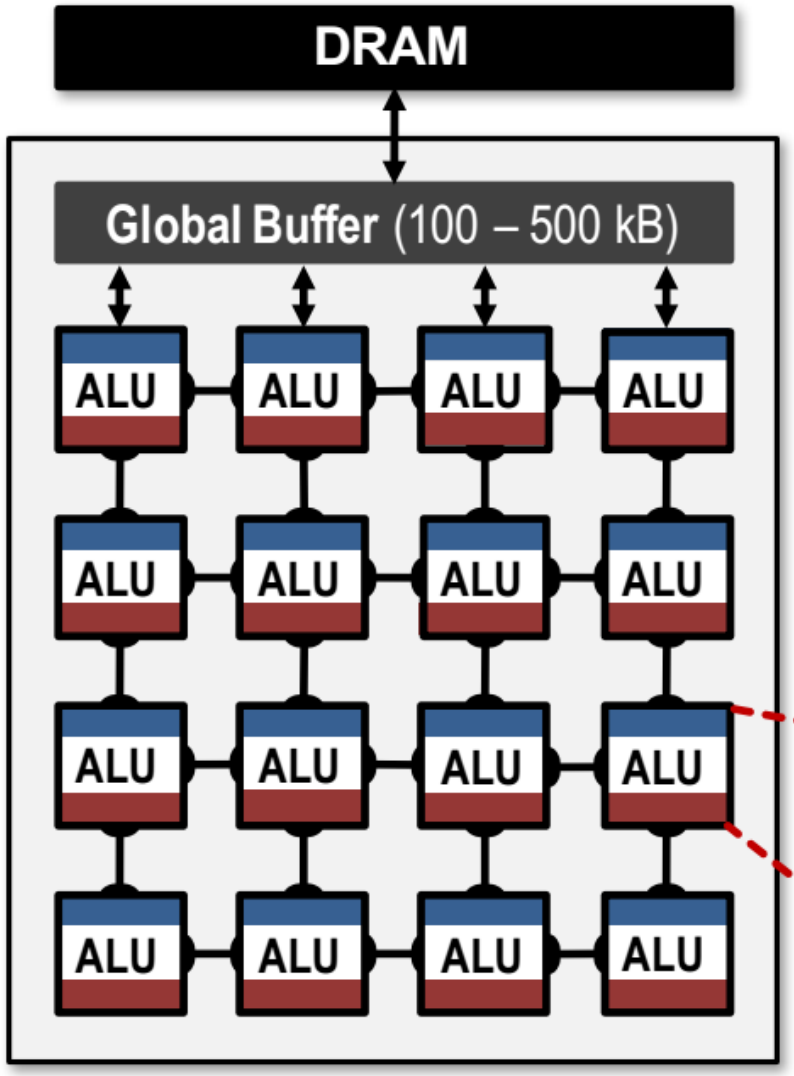
❷ Partial sum accumulation does **NOT** have to access DRAM

- Example: DRAM access in AlexNet can be reduced from **2896M** to **61M** (best case)

Spatial Architecture for CNN

Using CGRA idea, but routing is application specific

Data flow



Local Memory Hierarchy

- Global Buffer
- Direct inter-PE network
- PE-local memory (RF)

Processing Element (PE)

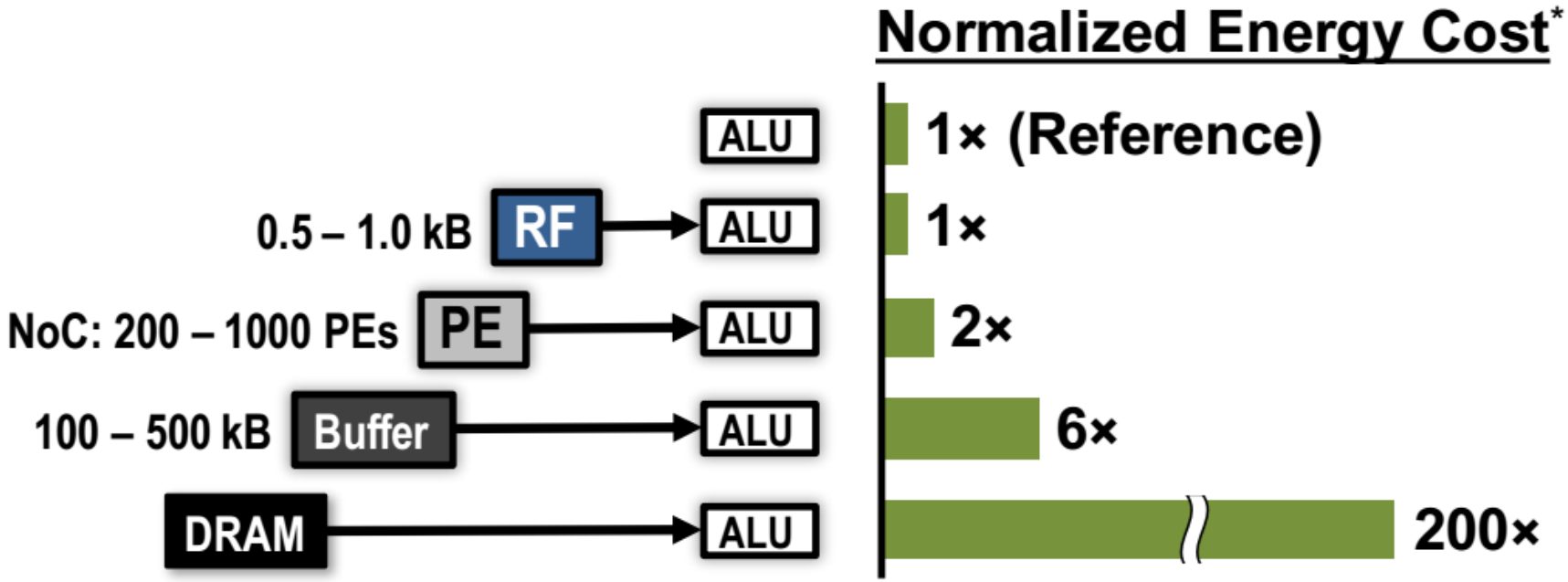
0.5 – 1.0 kB

The design methodology using the memory hierarchy idea but no need for coherency

Low-Cost Local Data Access

How to exploit ❶ **data reuse** and ❷ **local accumulation** with *limited* low-cost local storage?

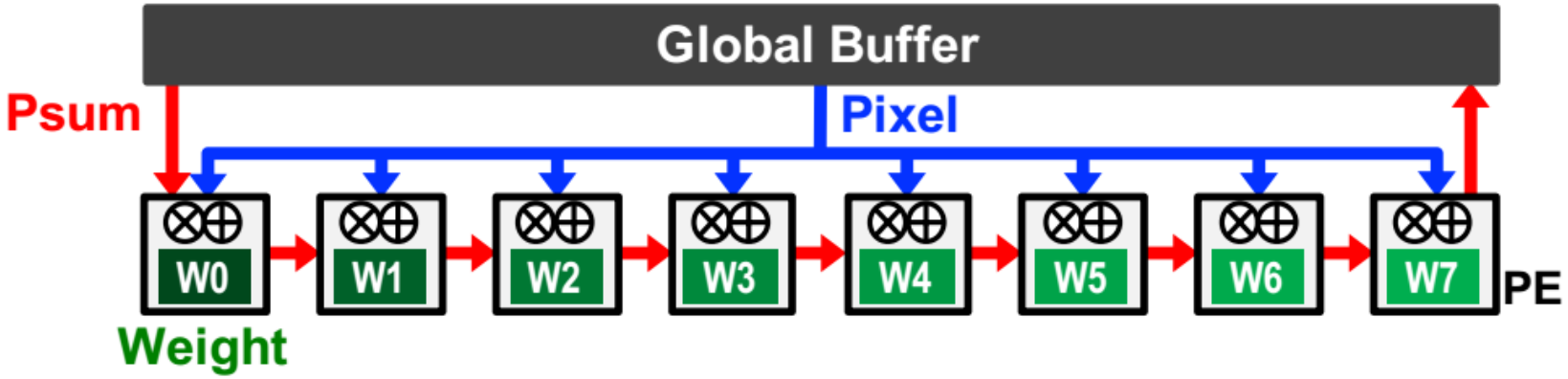
specialized **processing dataflow** required!



* measured from a commercial 65nm process

Common data flows for multi PE designs

Weight Stationary (WS)

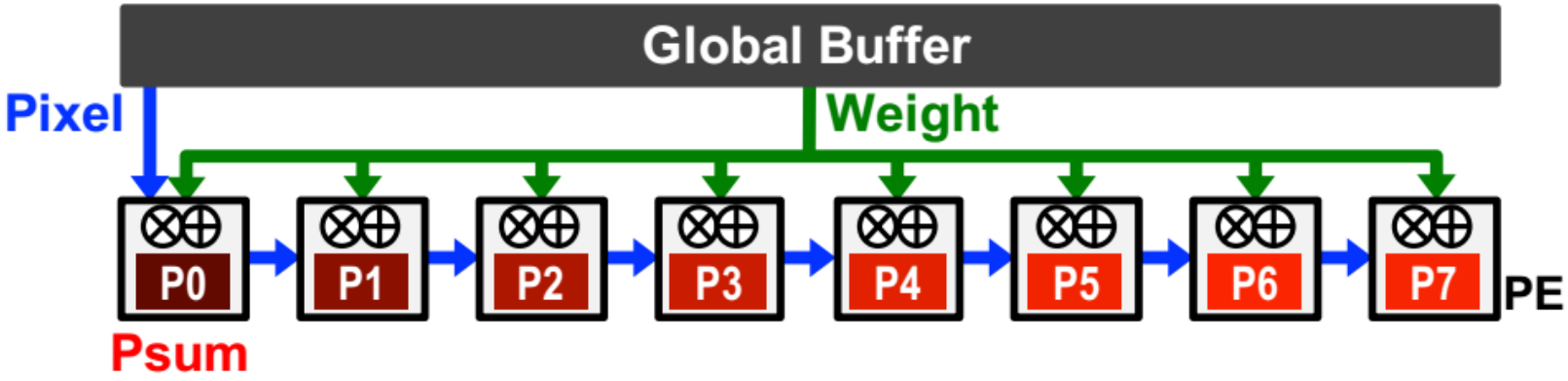


- **Minimize weight** read energy consumption
  - maximize convolutional and filter reuse of weights
- **Examples:**

[**Chakradhar**, *ISCA* 2010] [**nn-X (NeuFlow)**, *CVPRW* 2014]
[**Park**, *ISSCC* 2015] [**Origami**, *GLSVLSI* 2015]

Common data flows for multi PE designs

Output Stationary (OS)



- **Minimize partial sum** R/W energy consumption
  - maximize local accumulation
- **Examples:**

  [**Gupta**, *ICML* 2015]

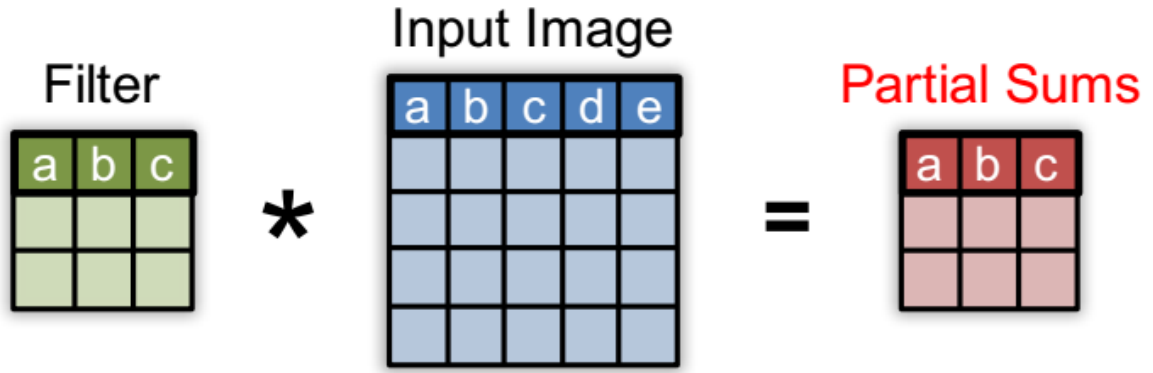
  [**Peemen**, *ICCD* 2013]

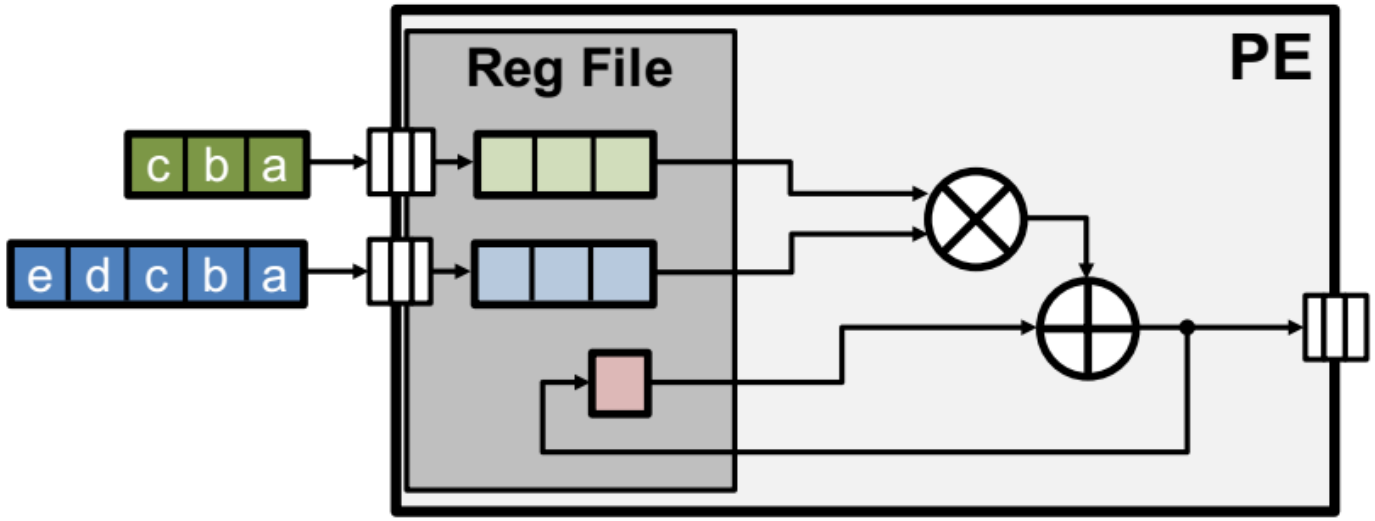
  [**ShiDianNao**, *ISCA* 2015]

1D Row Convolution in PE

Row Stationary

A specific Data flow for CNNs stride = 1

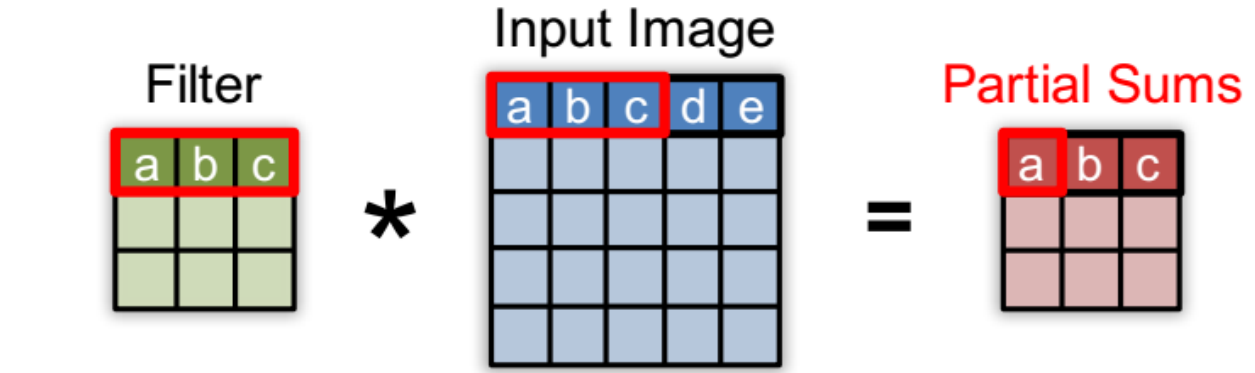


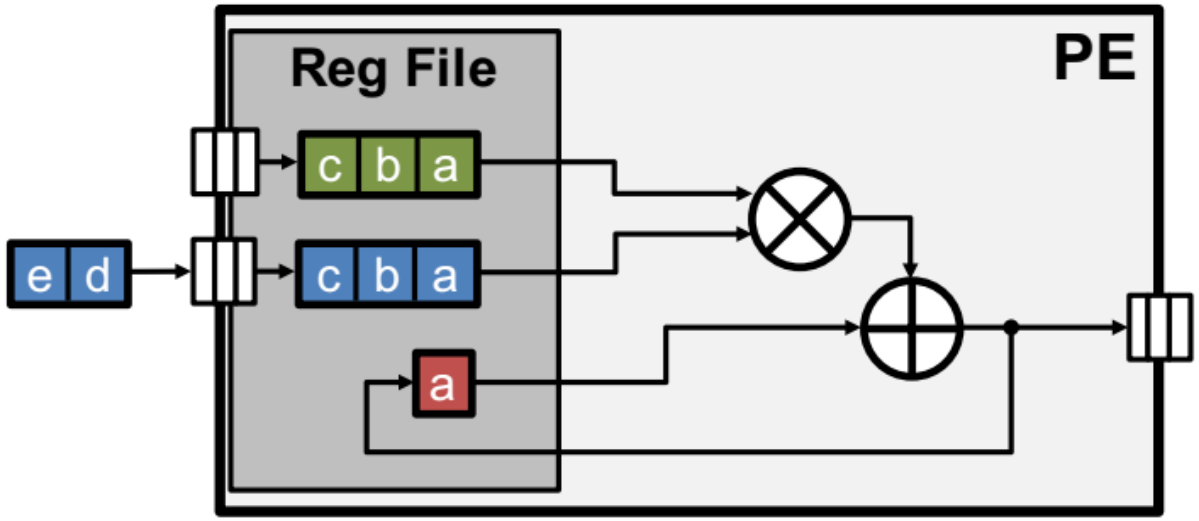


1D Row Convolution in PE

Row Stationary

A specific Data flow for CNNs stride = 1

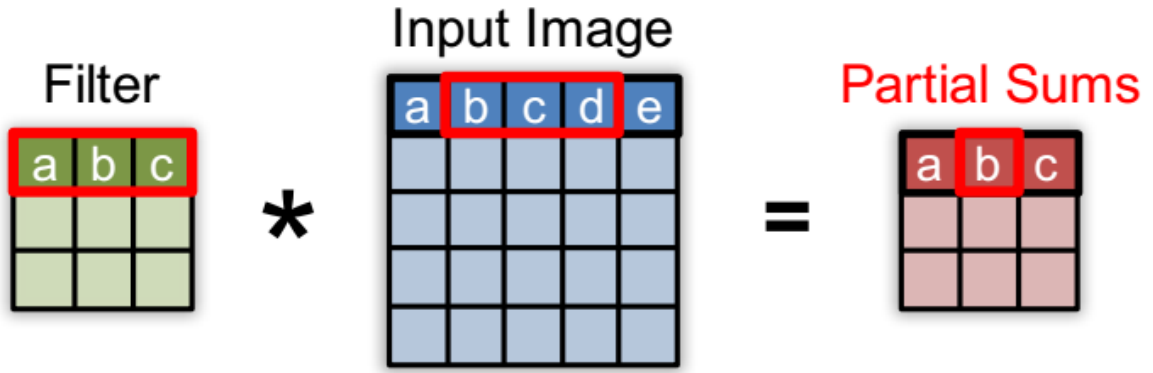


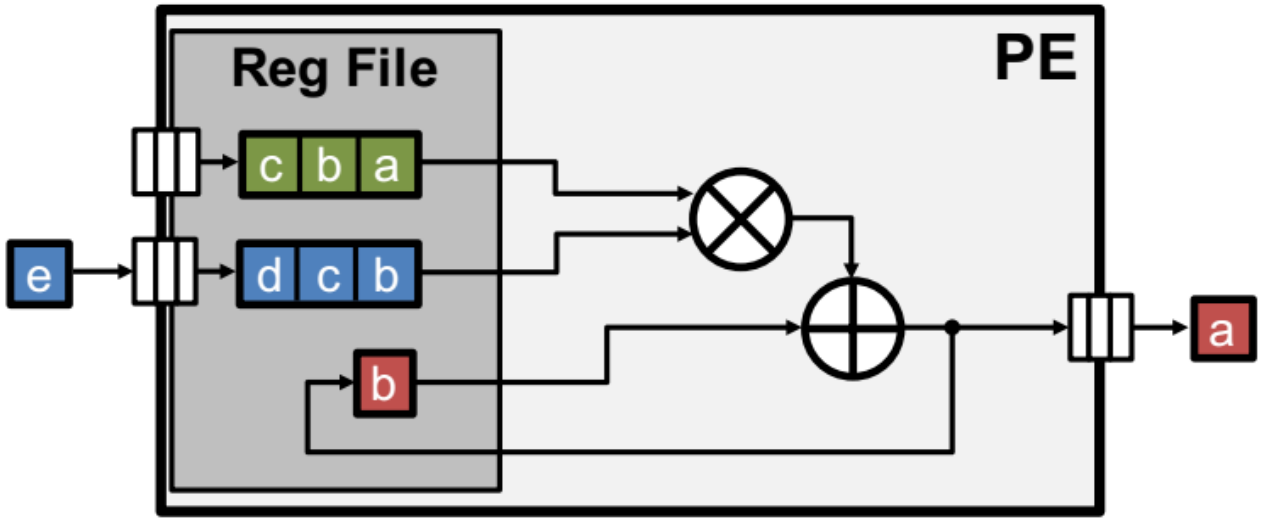


1D Row Convolution in PE

Row Stationary

A specific Data flow for CNNs stride = 1

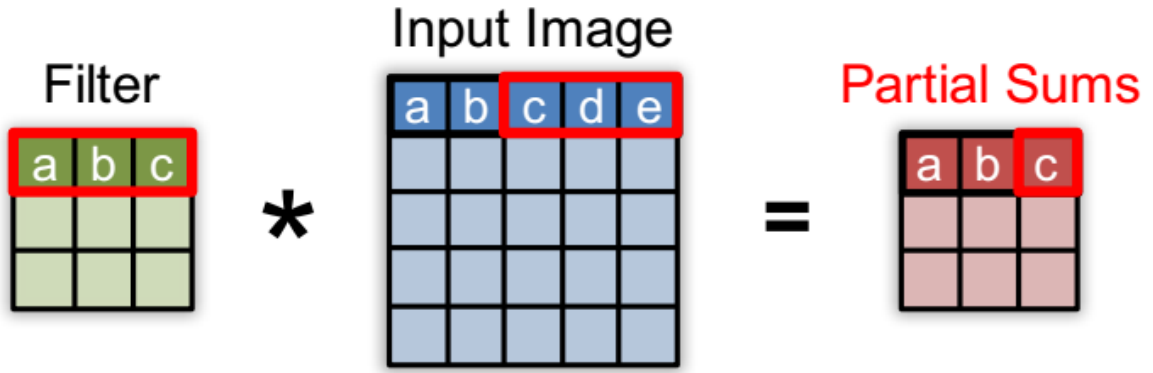


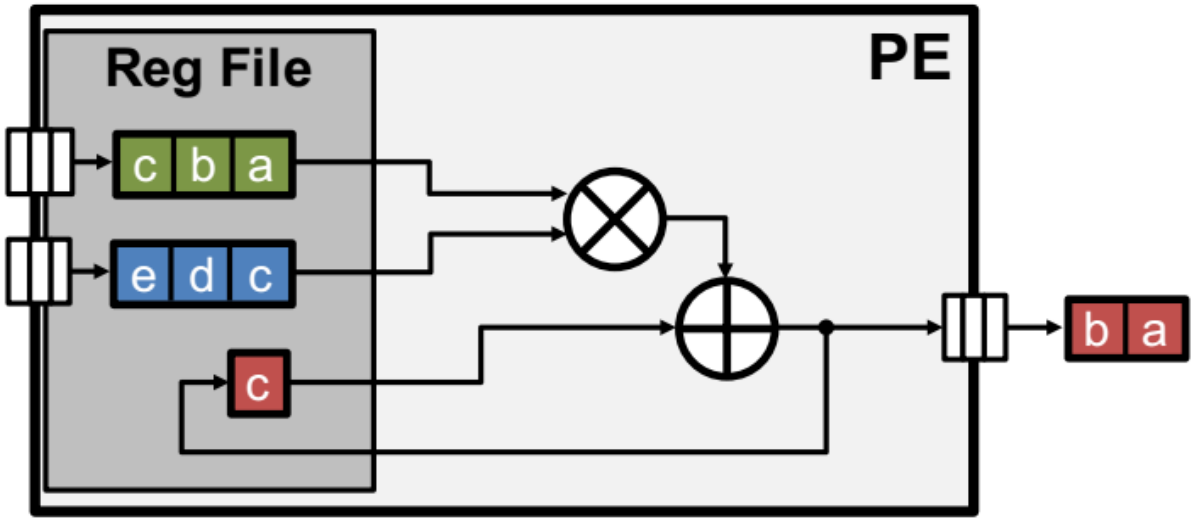


1D Row Convolution in PE

Row Stationary

A specific Data flow for CNNs stride = 1

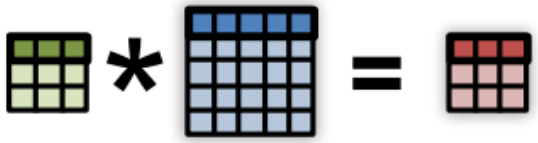




2D Convolution in PE Array

Row Stationary

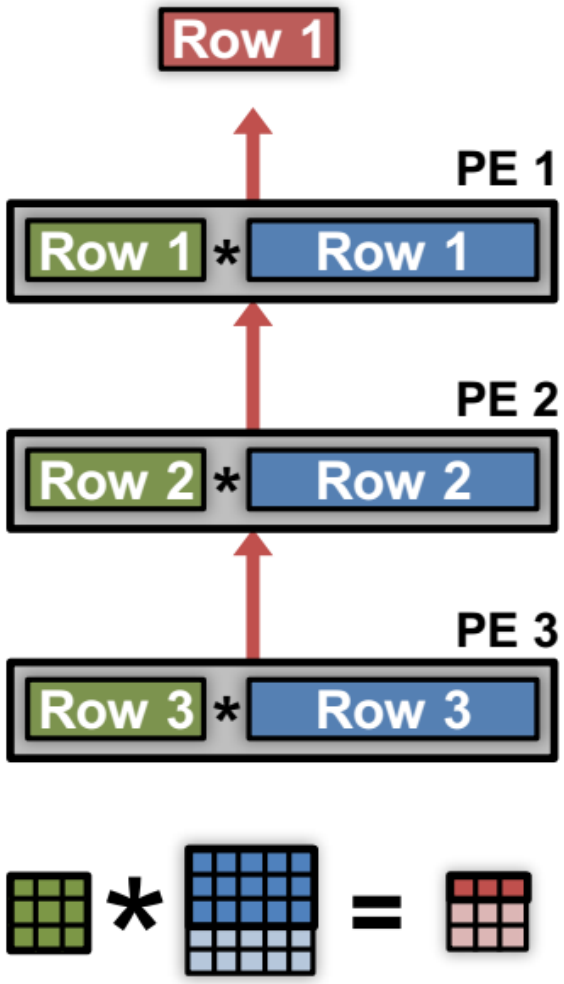
A specific Data flow for CNNs stride = 1



2D Convolution in PE Array

Row Stationary

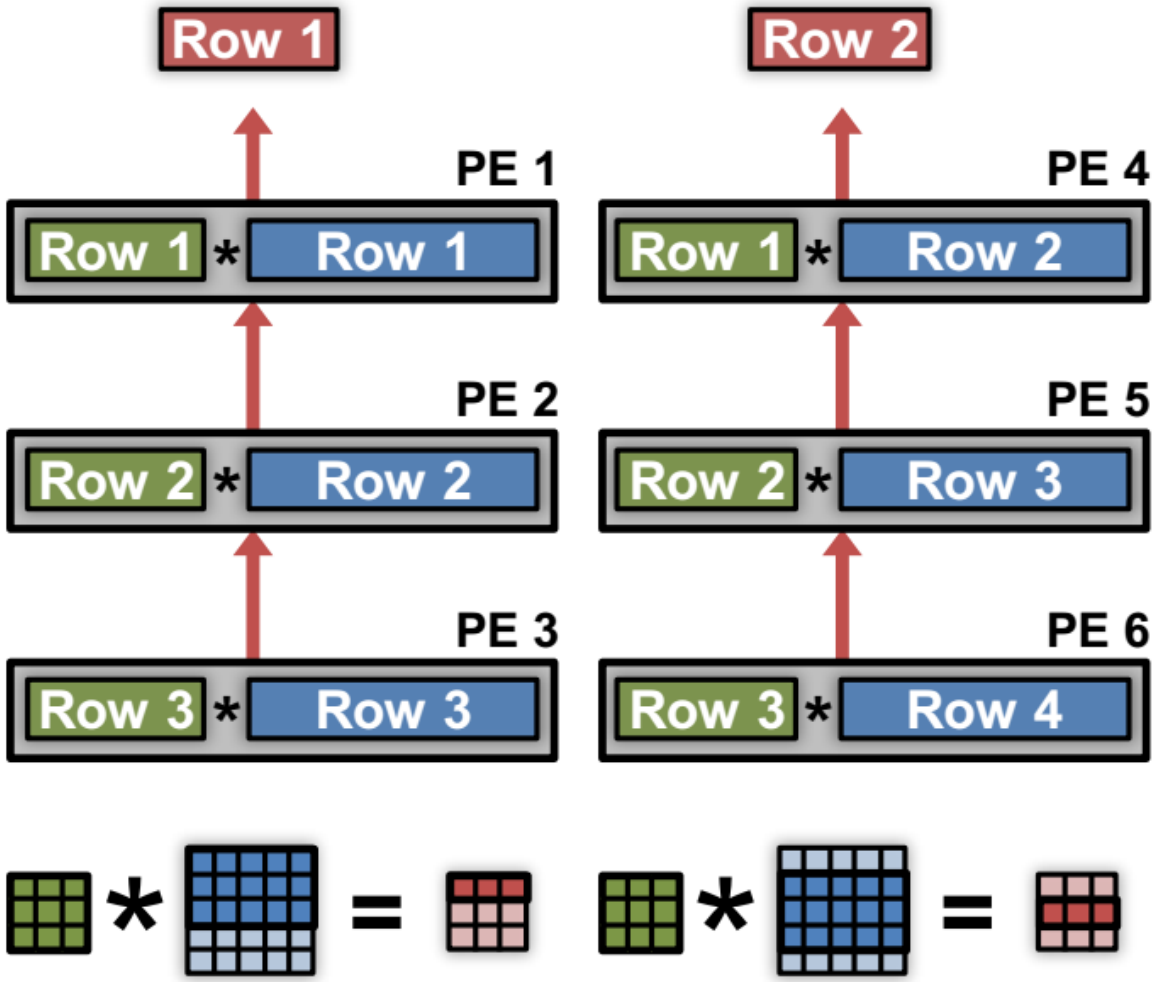
A specific Data flow for CNNs stride = 1



2D Convolution in PE Array

Row Stationary

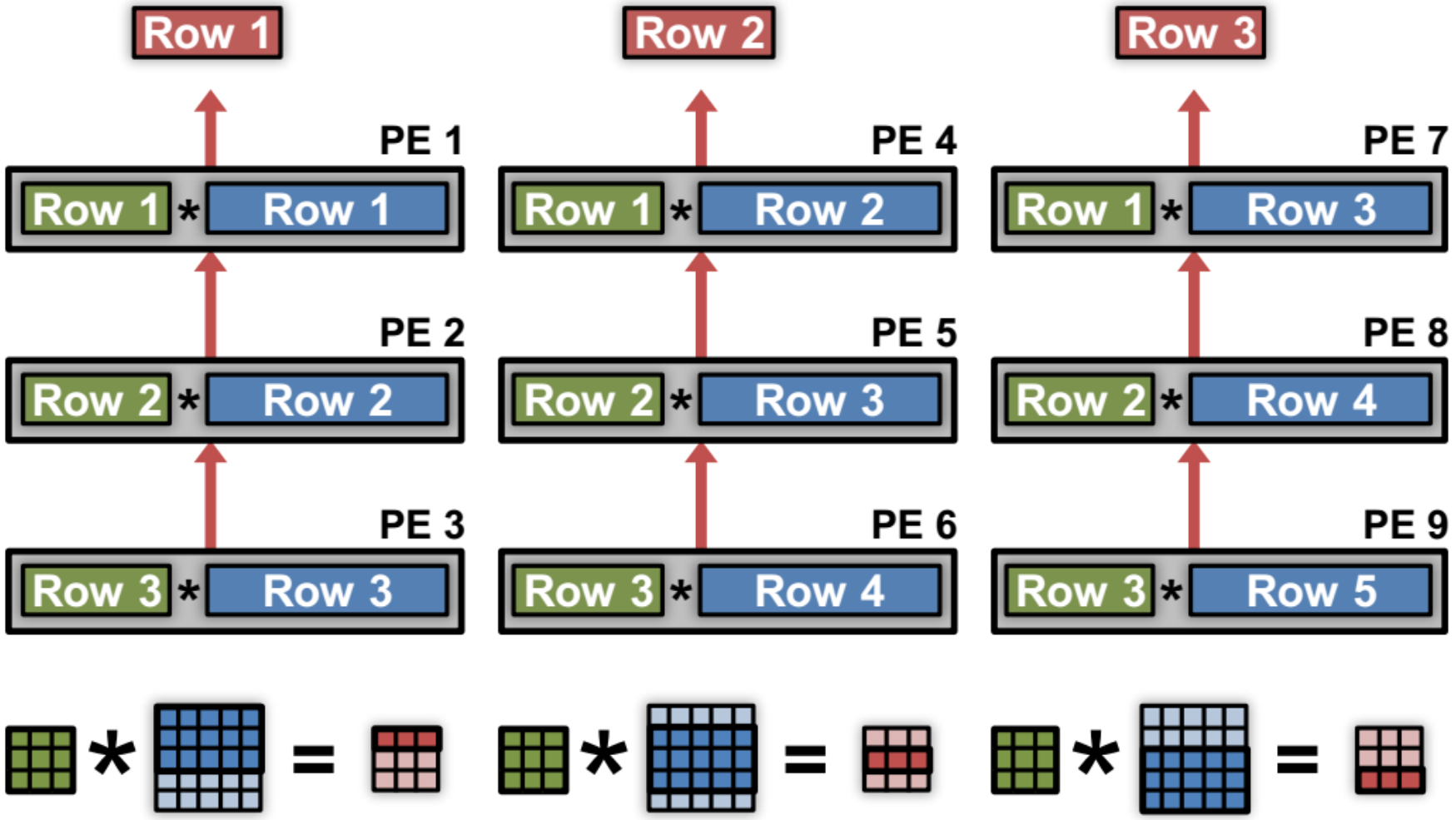
A specific Data flow for CNNs stride = 1



2D Convolution in PE Array

Row Stationary

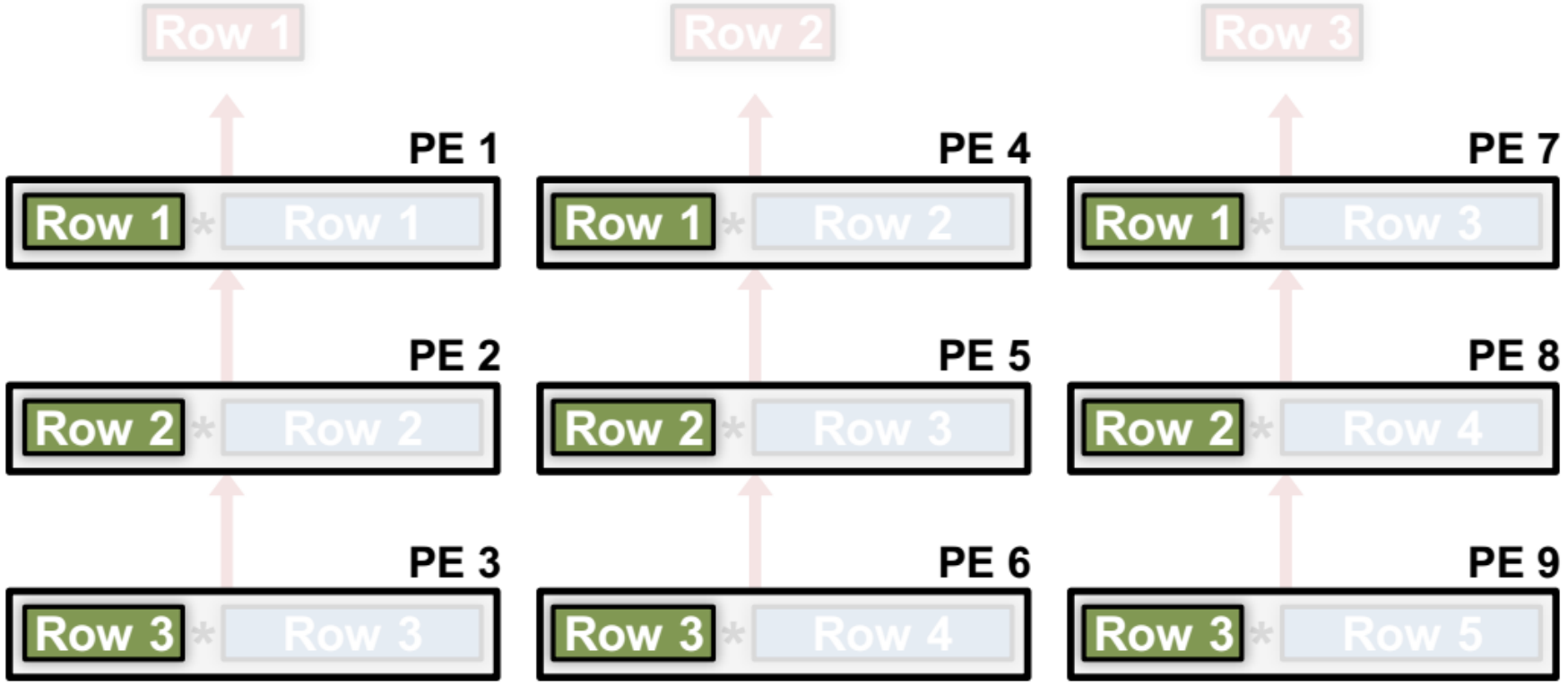
A specific Data flow for CNNs stride = 1



Convolutional Reuse Maximized

Row Stationary

A specific Data flow for CNNs stride = 1



Filter rows are reused across PEs **horizontally**

Convolutional Reuse Maximized

Row Stationary

A specific Data flow for CNNs stride = 1

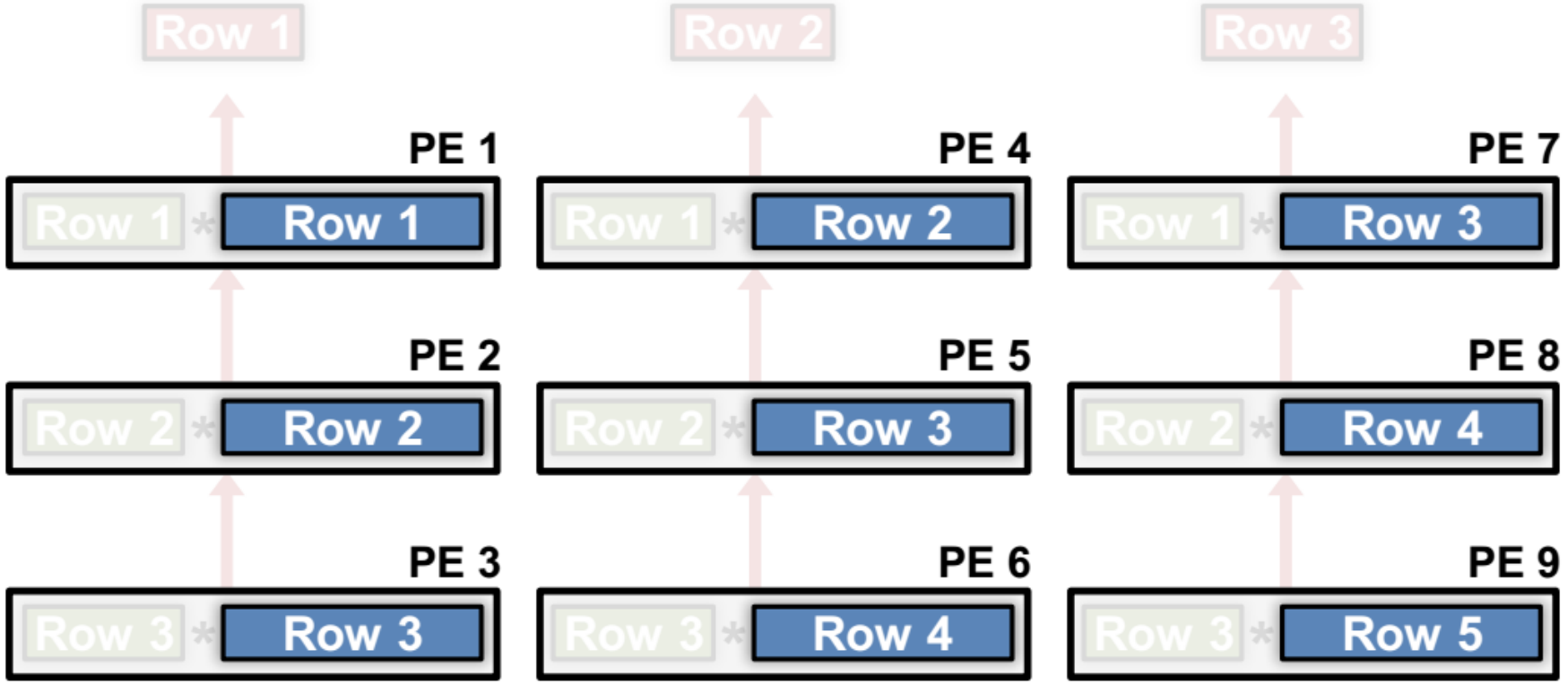
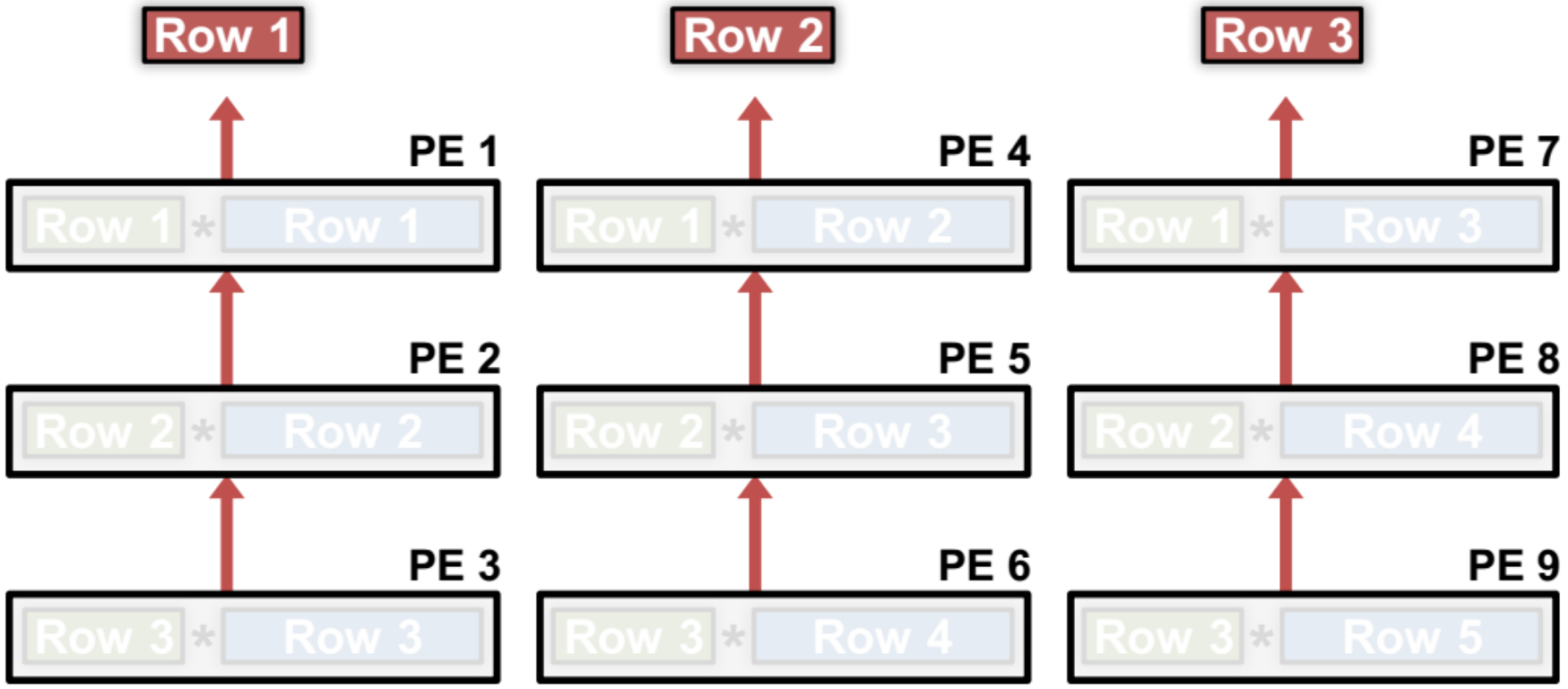


Image rows are reused across PEs **diagonally**

Maximize 2D Accumulation in PE Array

Row Stationary

A specific Data flow for CNNs stride = 1

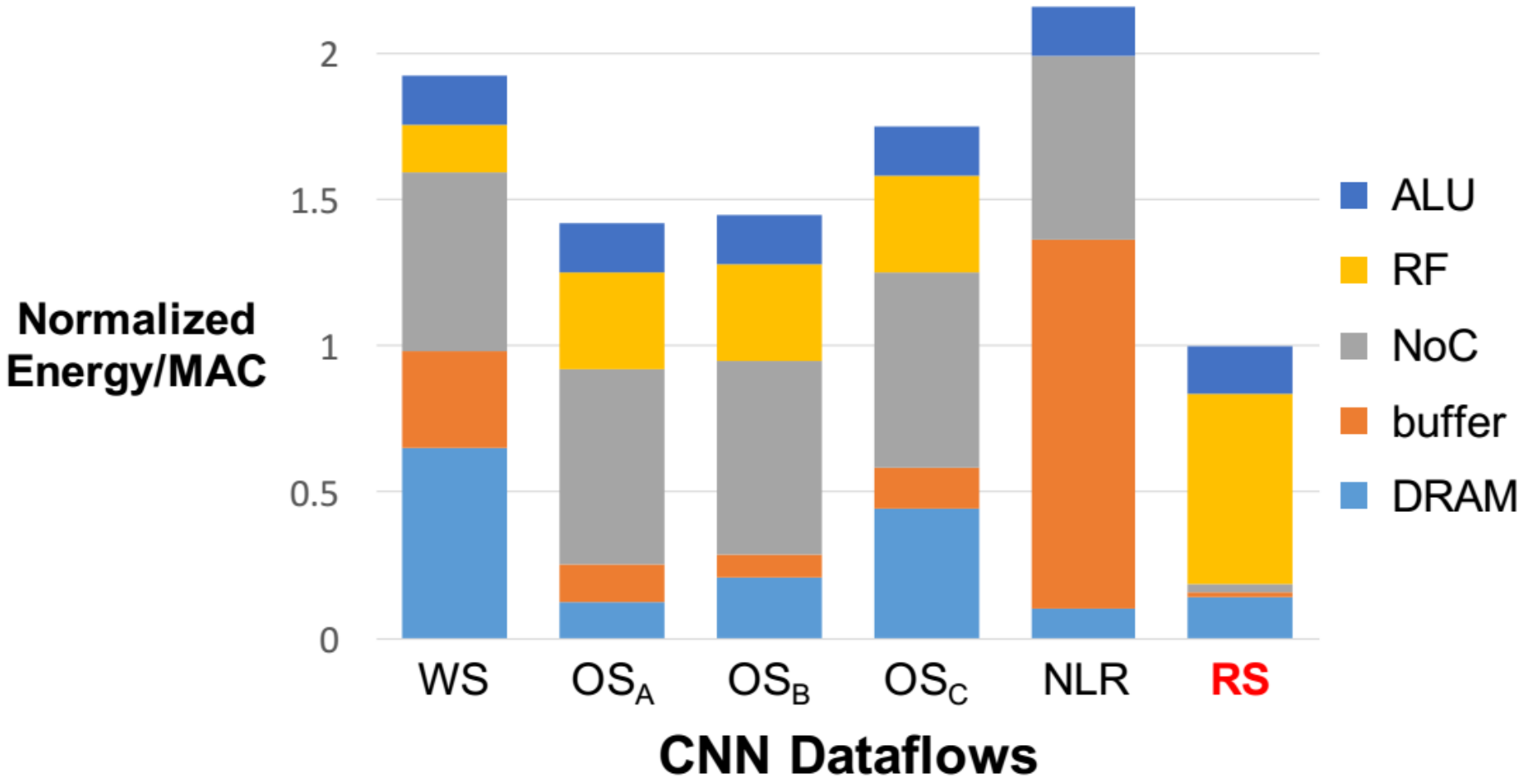


Partial sums accumulate across PEs **vertically**

Dataflow Comparison: CONV Layers

Row Stationary Performance

Case study: Eyeriss from MIT, ISSCC/ISCA 2016



RS uses **1.4× – 2.5× lower** energy than other dataflows

Conclusions

FPGA ,CGRA, Data flow (systolic Array)