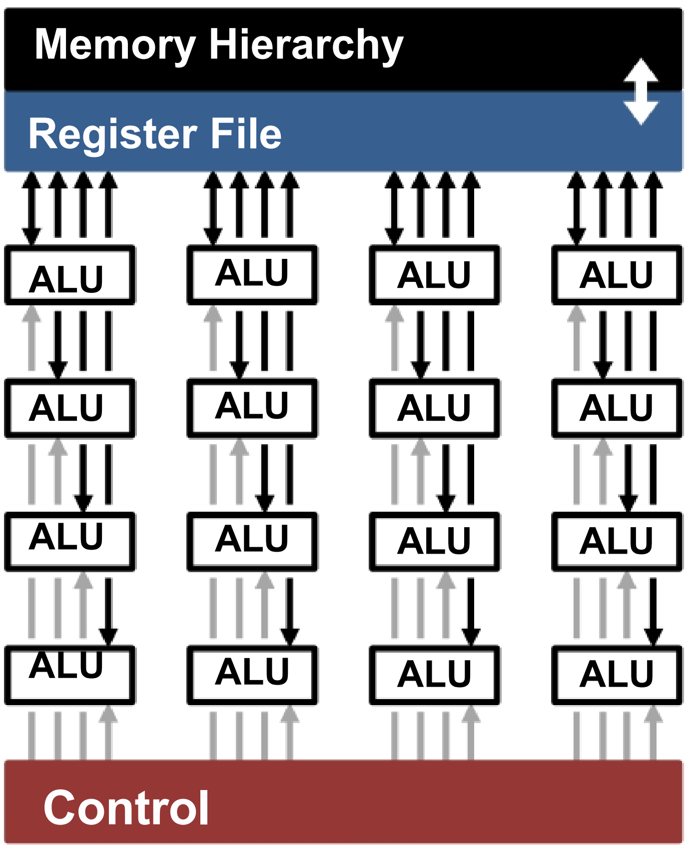
DNN Accelerator Architectures

Highly-Parallel Compute Paradigms

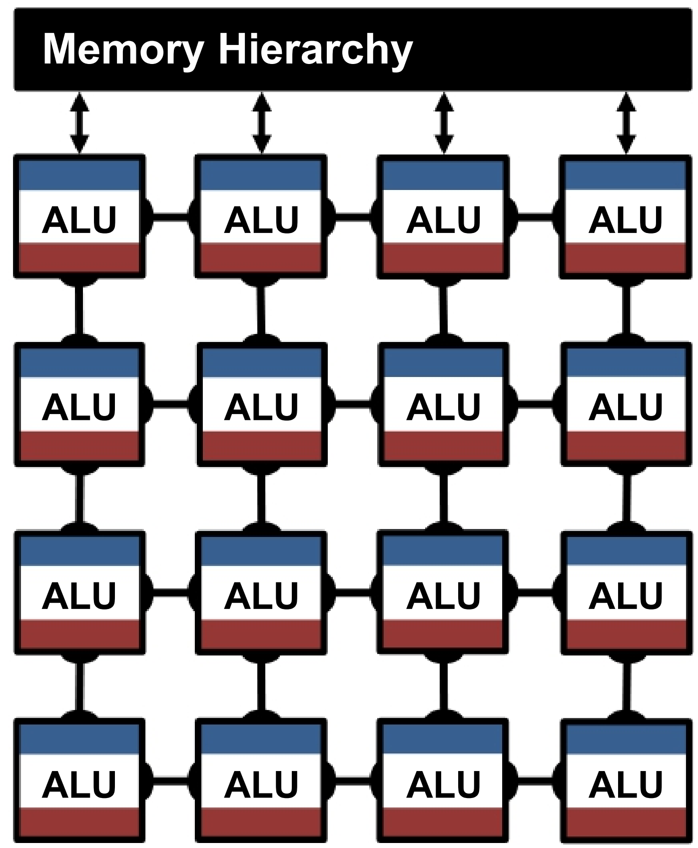
Temporal Architecture(SIMD/SIMT)



Mostly in CPU & GPU

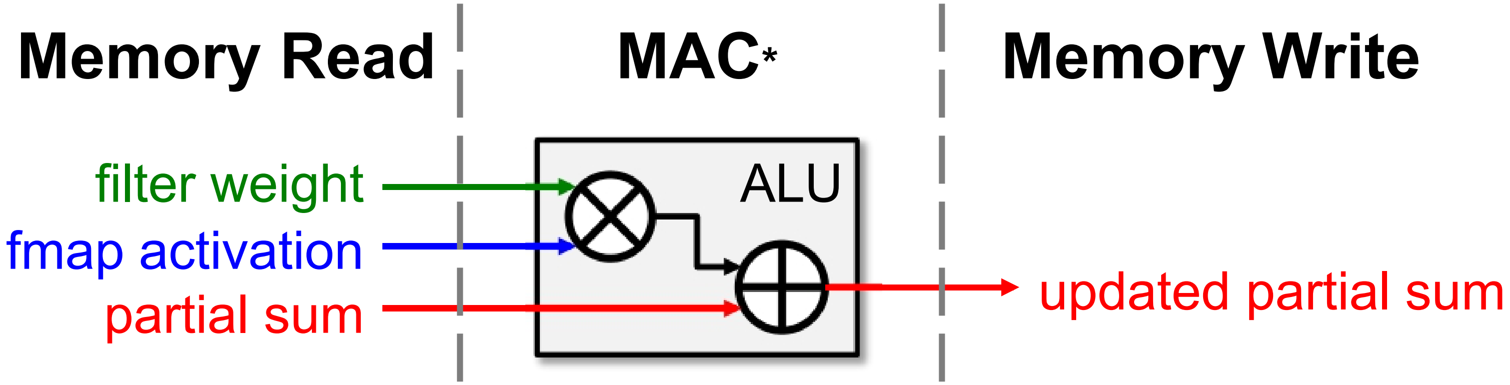
Centralized control for many ALUs (cannot communicate directly with each other directly

Spatial Architecture (Dataflow Processing)



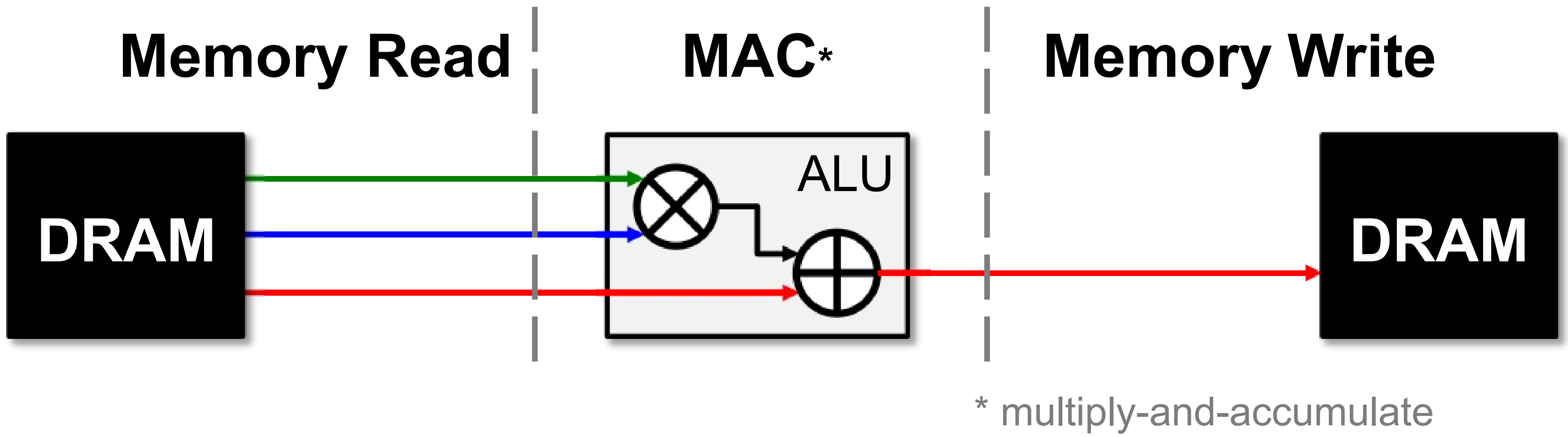
Data are passed from one ALU to another ALU
Mostly in ASIC or FPGA

Memory Access is the Bottleneck



* multiply-and-accumulate

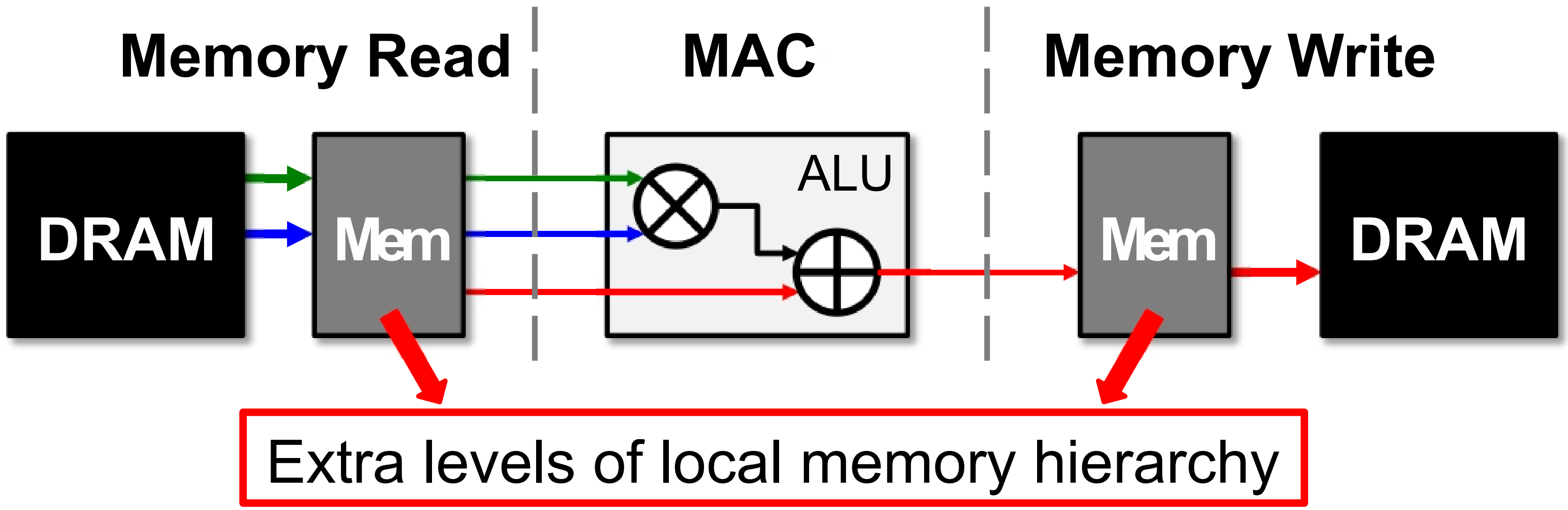
Memory Access is the Bottleneck



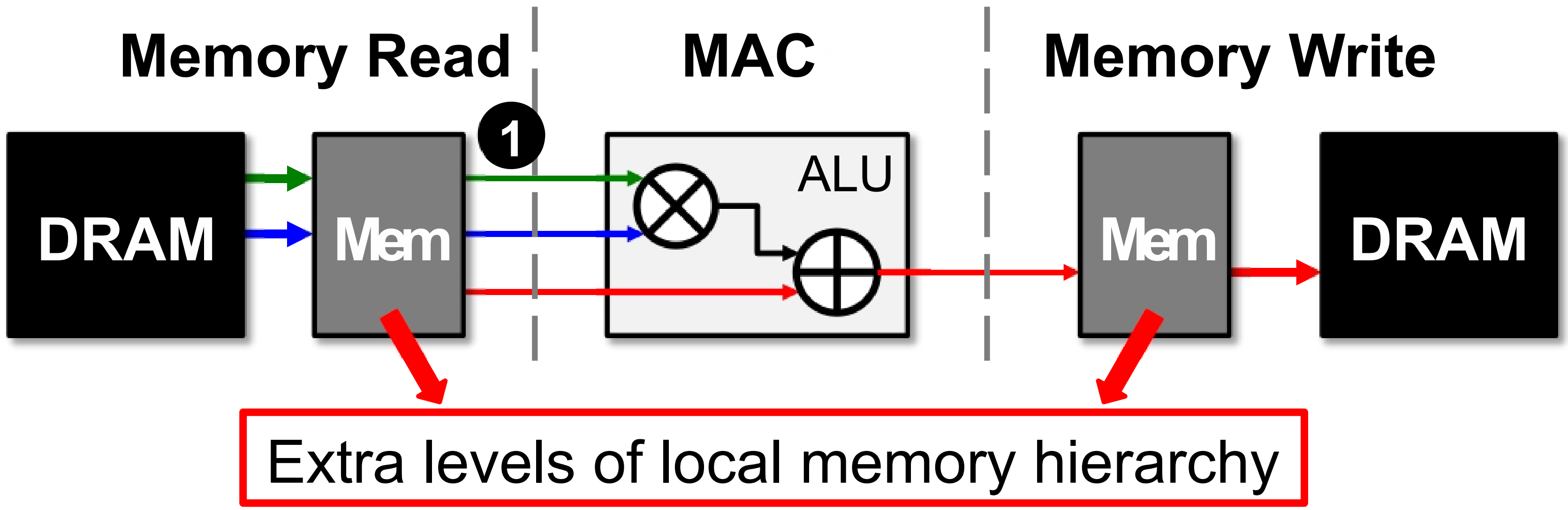
Worst Case: all memory R/W are **DRAM** accesses

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

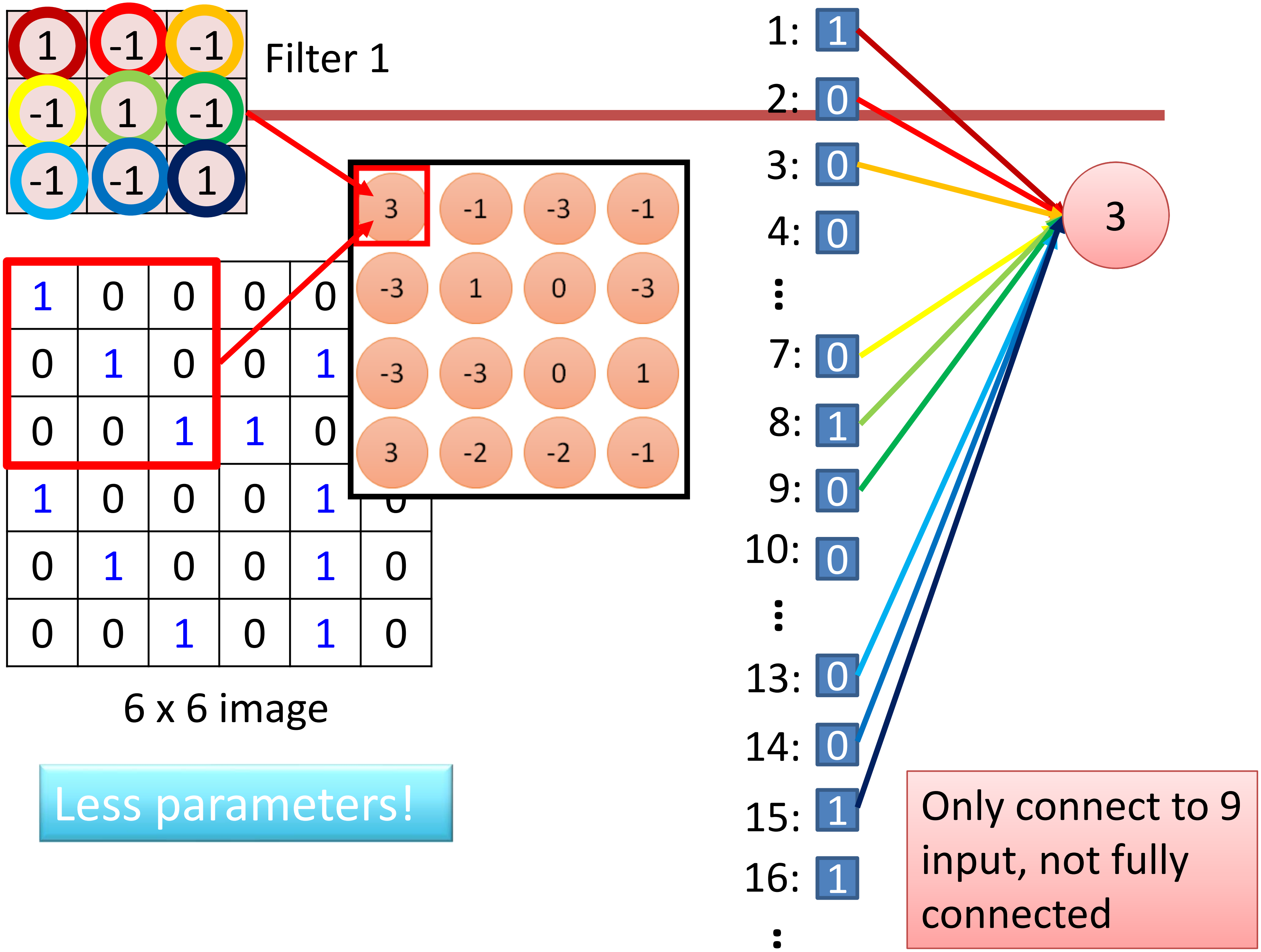
Memory Access is the Bottleneck

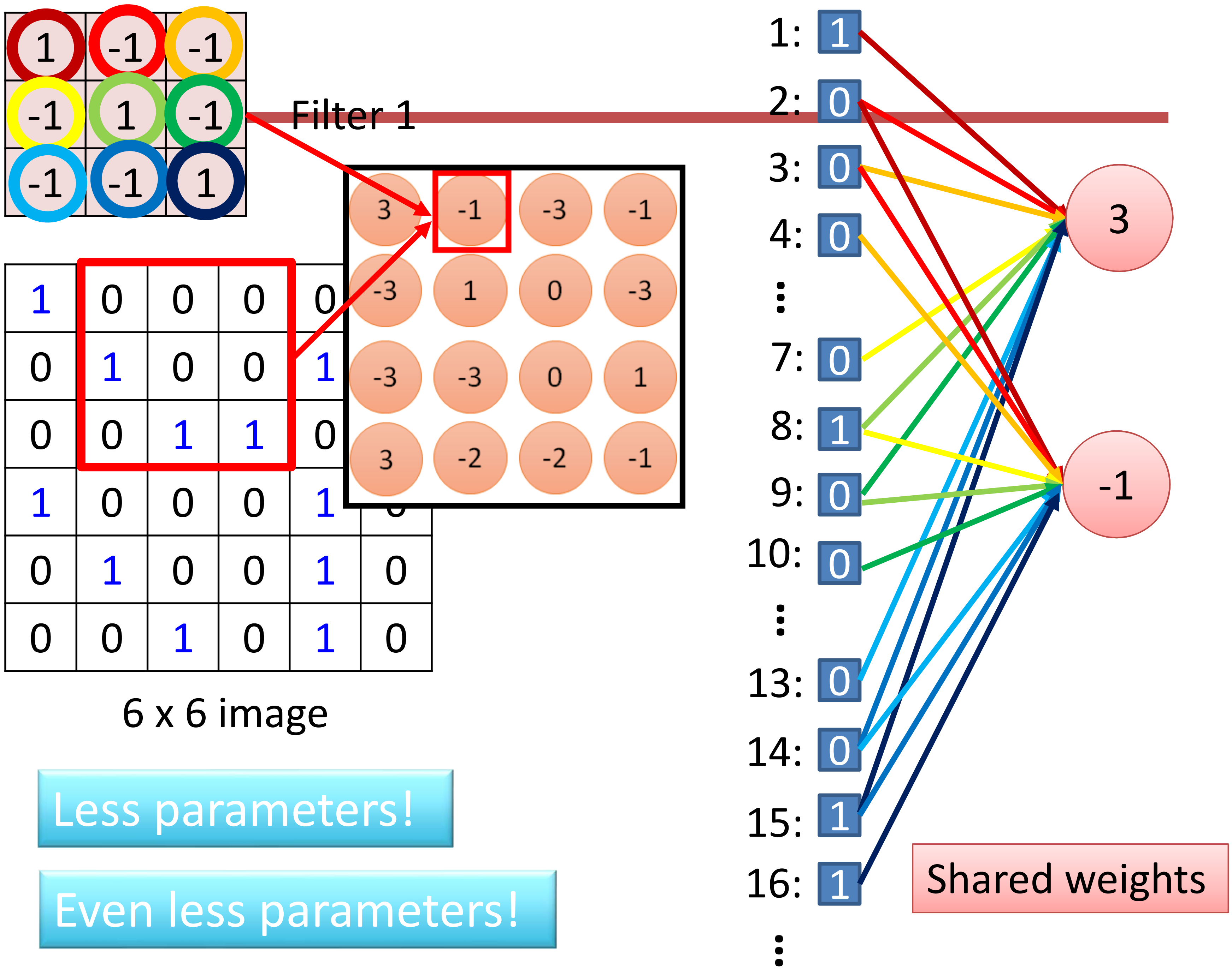


Memory Access is the Bottleneck



<u>Opportunities</u>: 1 **data reuse**


Filter 1
1 -1 -1
-1 1 -1
-1 -1 1
6 x 6 image
1 0 0 0 0 1
0 1 0 0 1 0
0 0 1 1 0 0
1 0 0 0 1 0
0 1 0 0 1 0
0 0 1 0 1 0
3 -1 -3 -1
-3 1 0 -3
-3 -3 0 1
3 -2 -2 -1
Less parameters!
1: 1
2: 0
3: 0
4: 0
7: 0
8: 1
9: 0
10: 0
13: 0
14: 0
15: 1
16: 1
3
Only connect to 9 input, not fully connected

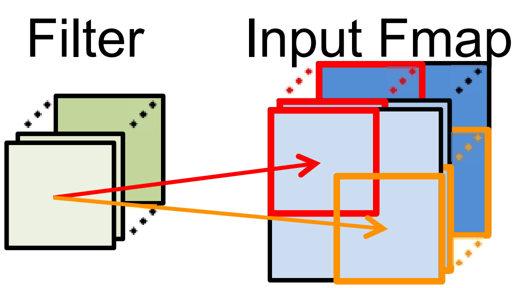


Filter 1
1 -1 -1
-1 1 -1
-1 -1 1
3 -1 -3 -1
-3 1 0 -3
-3 -3 0 1
3 -2 -2 -1
1 0 0 0 0
0 1 0 0 1
0 0 1 1 0
1 0 0 0 1 0
0 1 0 0 1 0
0 0 1 0 1 0
6 x 6 image
Less parameters!
Even less parameters!
1: 1
2: 0
3: 0
4: 0
7: 0
8: 1
9: 0
10: 0
13: 0
14: 0
15: 1
16: 1
3
-1
Shared weights

Types of Data Reuse in DNN

Convolutional Reuse

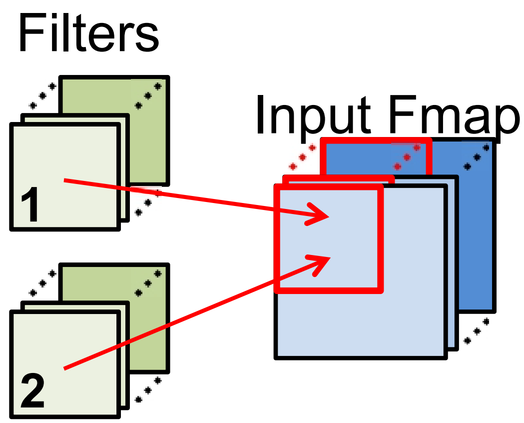
In CONV layers only
(sliding window)



Reuse Both Activations and Filter weights

Fmap Reuse

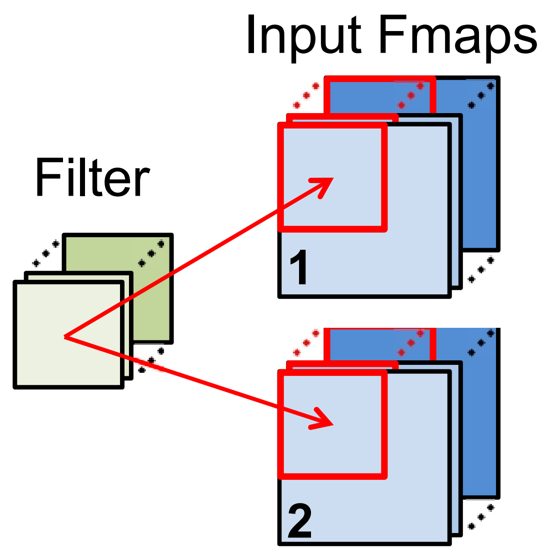
In both CONV and FC layers



Reuse Activations

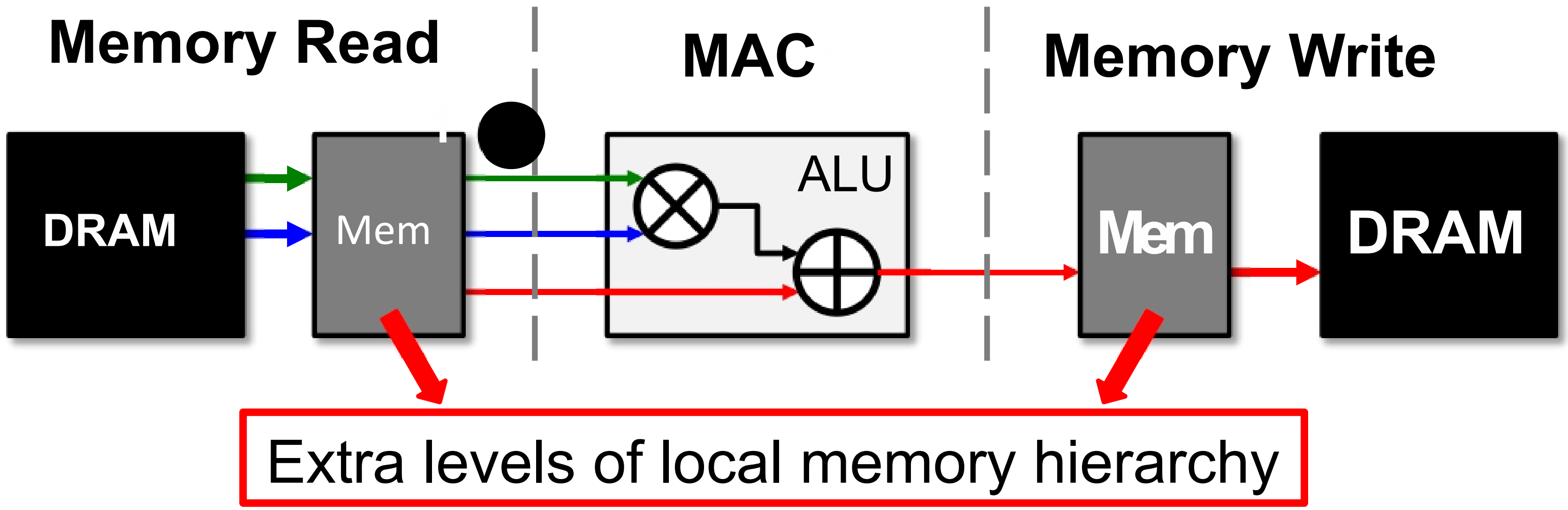
Filter Reuse

In both CONV and FC layers (batch size > 1)



Reuse Filter weights

Memory Access is the Bottleneck

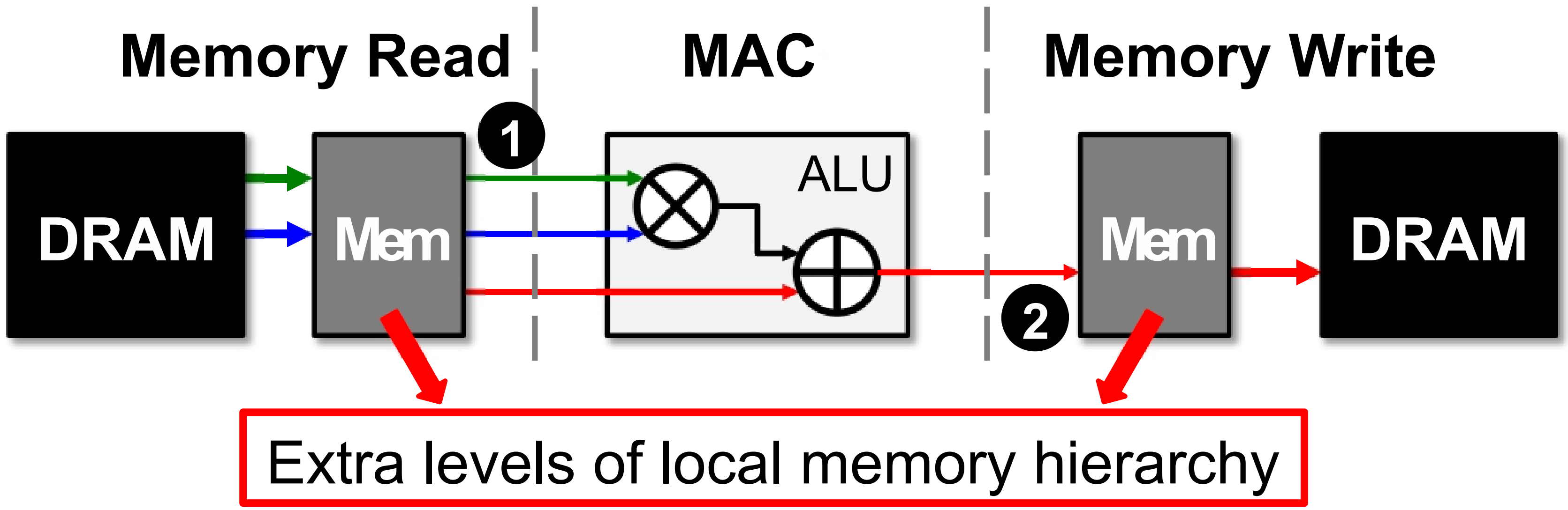


Opportunities: 1 **data reuse**

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

** AlexNet CONV layers

Memory Access is the Bottleneck

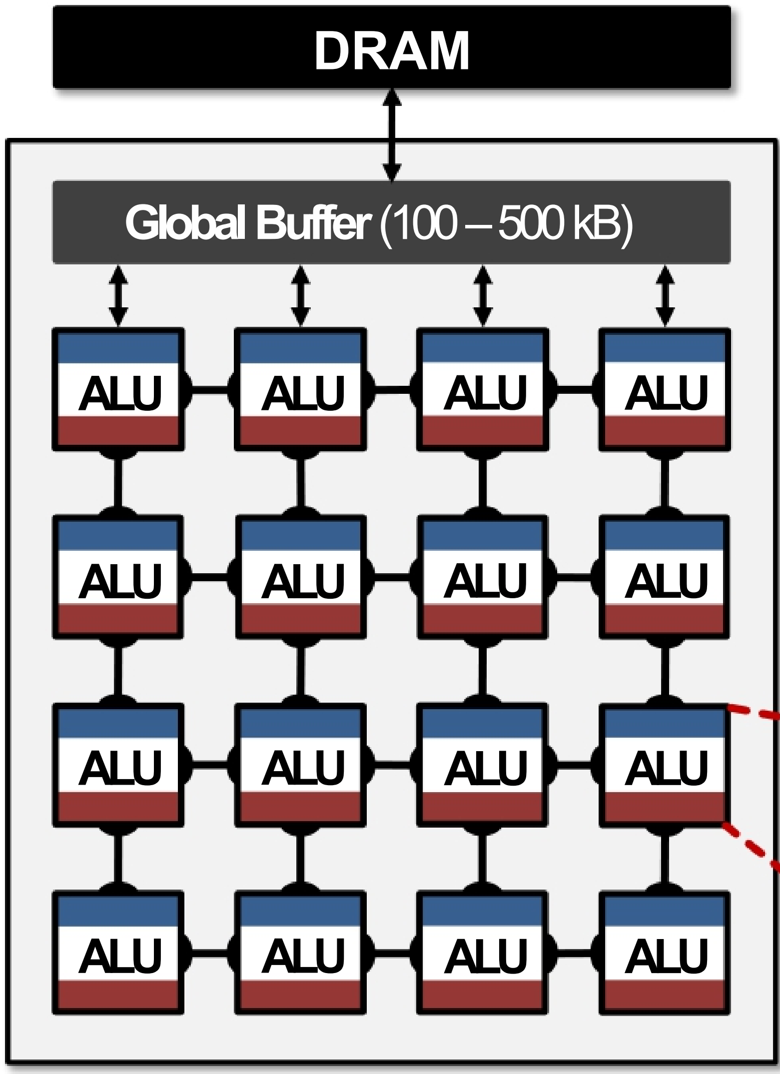


Opportunities: 1 **data reuse** 2 **local accumulation**

1) Can reduce DRAM reads of filter/fmap by up to **500×**
2) Partial sum accumulation does **NOT** have to access DRAM

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

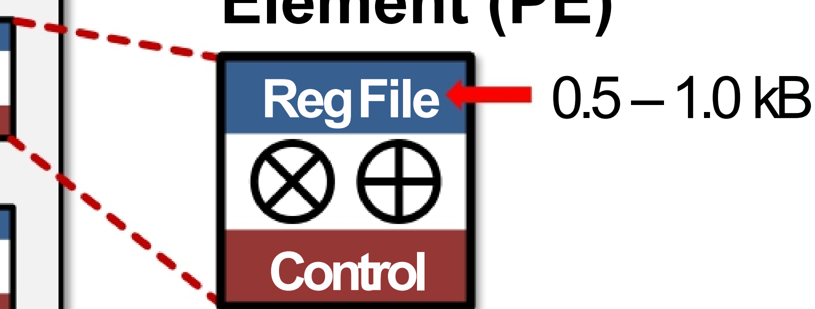
Spatial Architecture for DNN



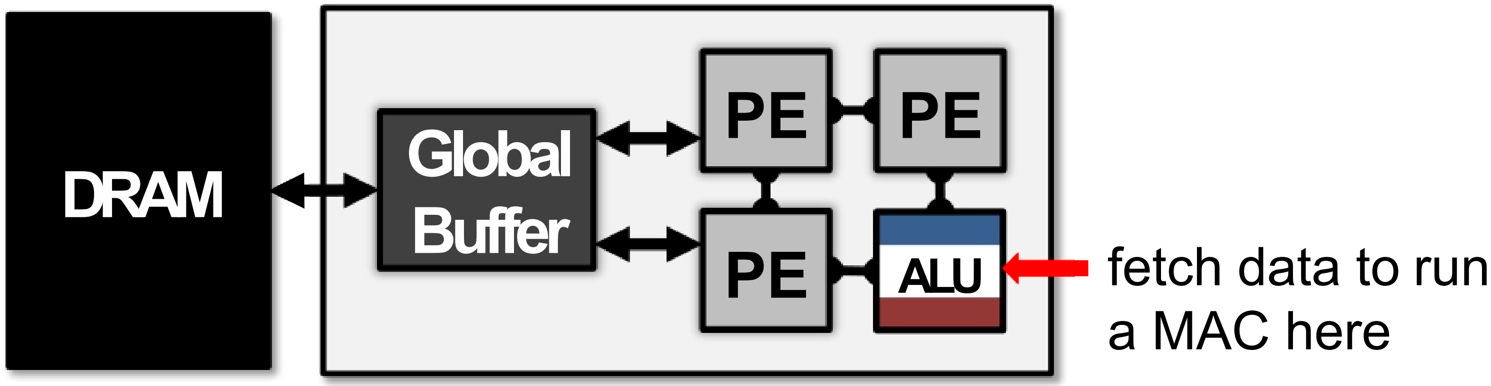
Local Memory Hierarchy

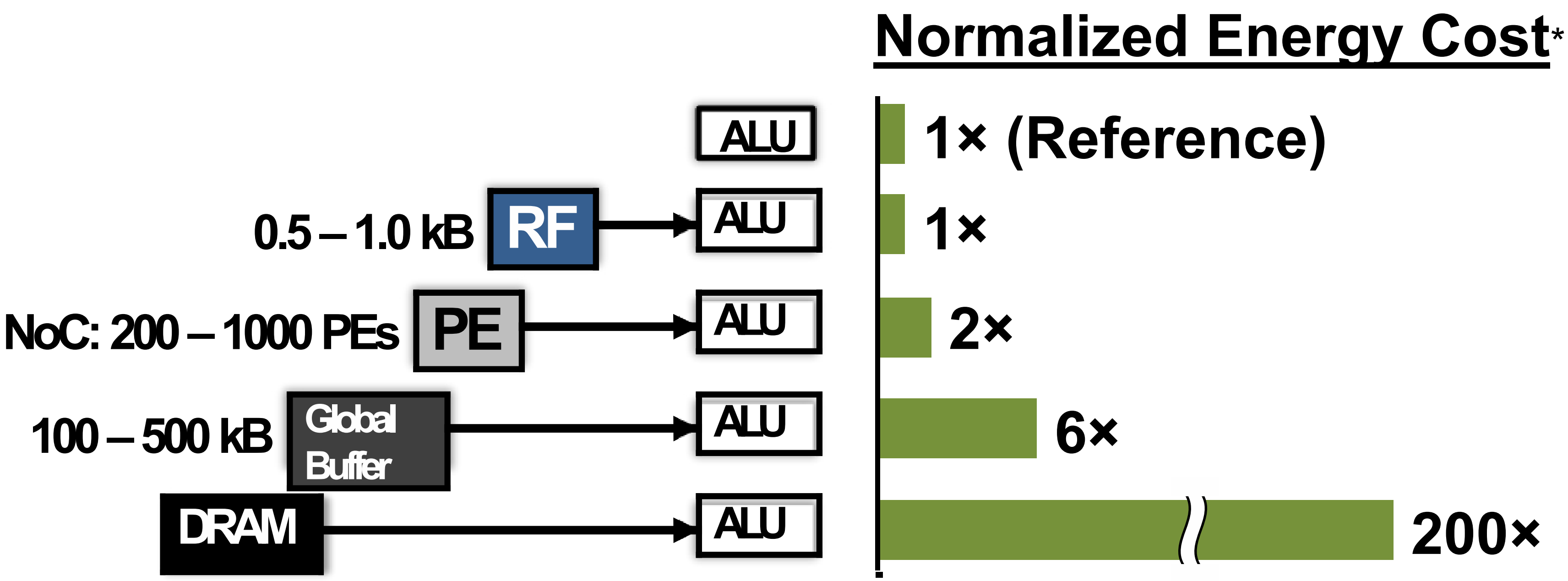
- Global Buffer
- Direct inter-PE network
- PE-local memory (RF)

Processing Element (PE)



Low-Cost Local Data Access



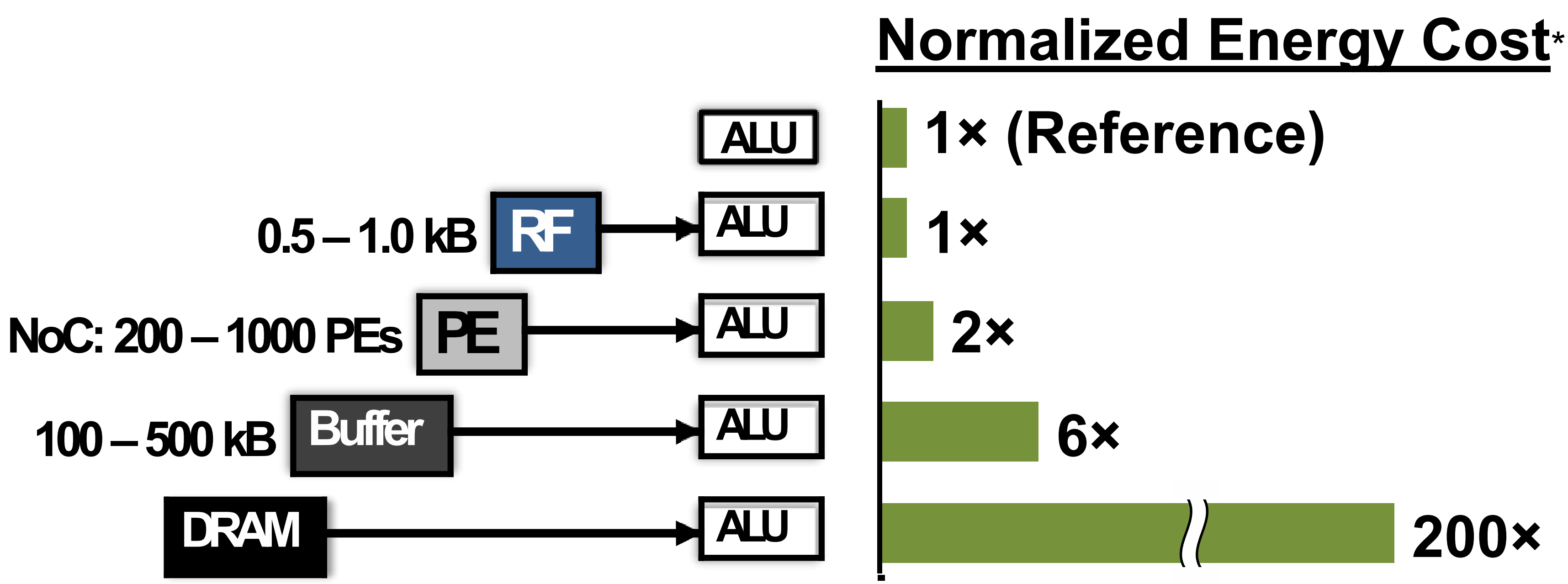


* measured from a commercial 65nm process

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!



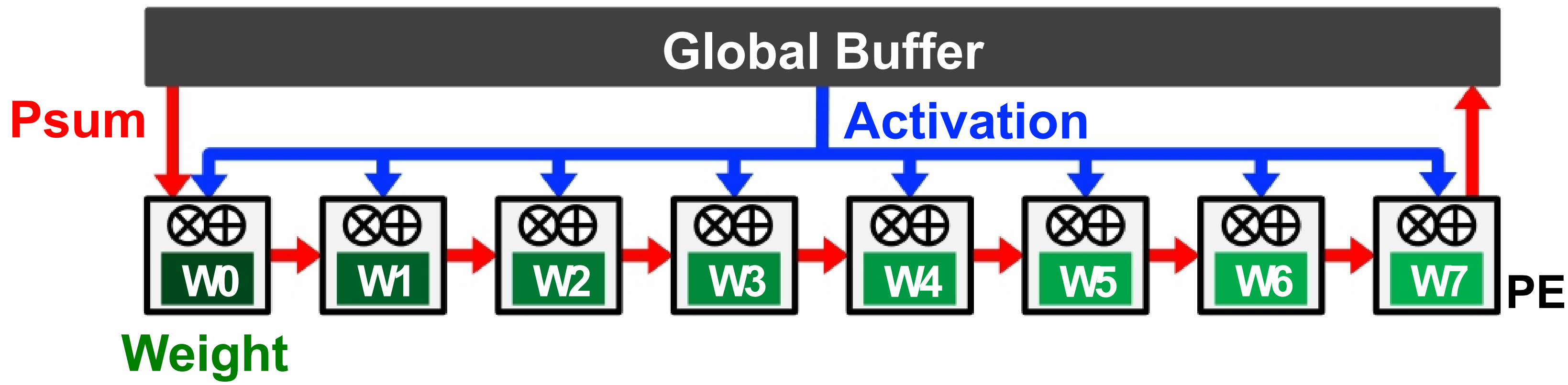
* measured from a commercial 65nm process

Dataflow Taxonomy

- Weight Stationary (WS)
- Output Stationary (OS)
- No Local Reuse (NLR)
- Row Stationary

[Chen et al., ISCA 2016]

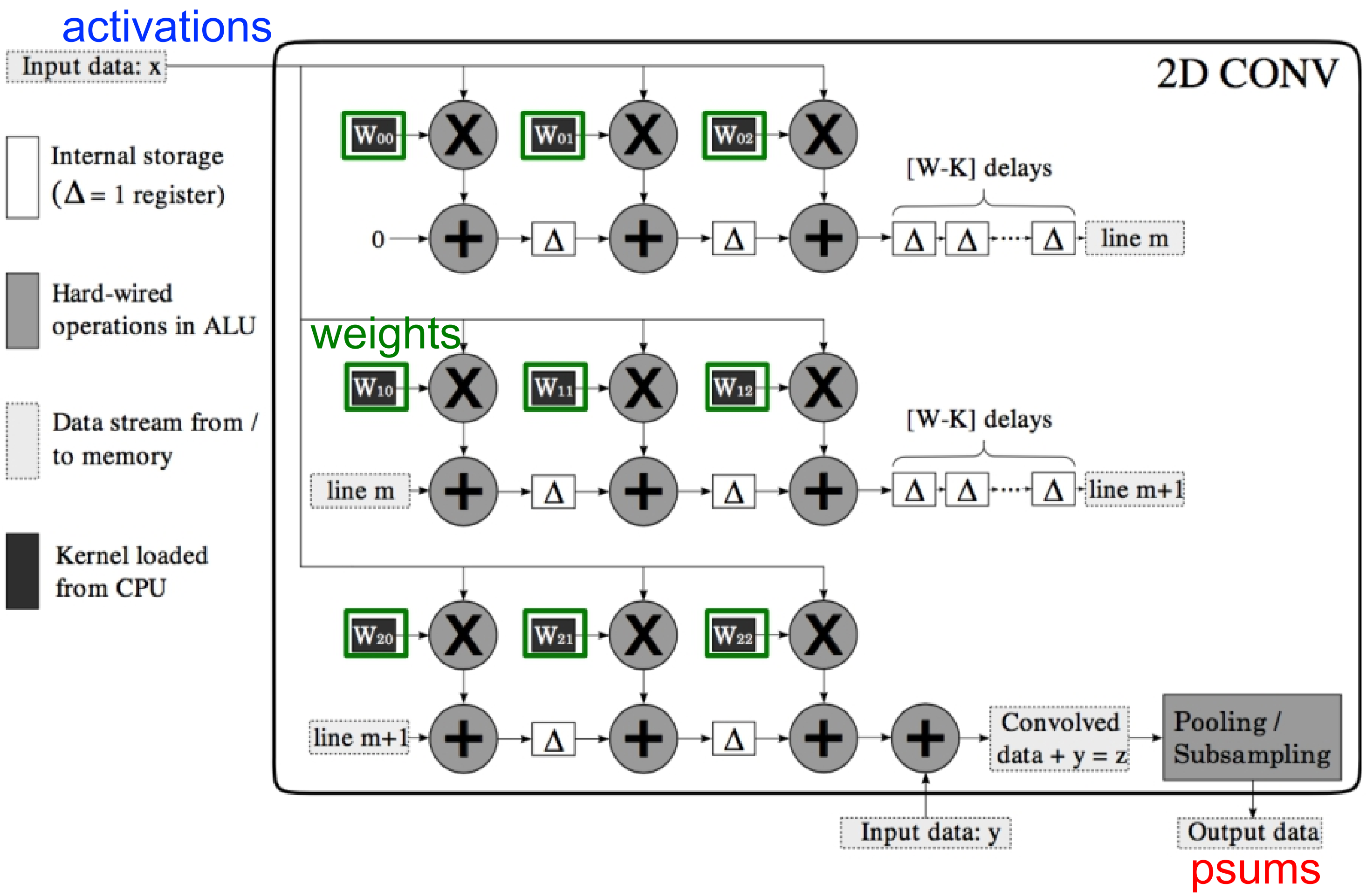
Weight Stationary (WS)



- **Minimize** **weight** read energy consumption
  - maximize convolutional and filter reuse of weights
- **Broadcast** **activations** and **accumulate** **psums** **spatially** across the PE array.

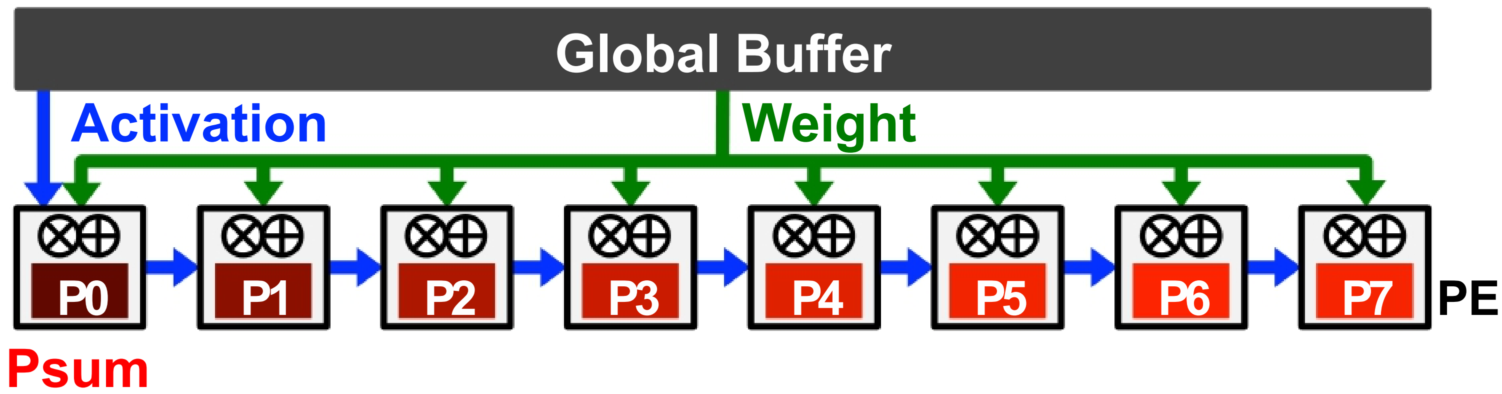
WS Example: nn-X (NeuFlow)

A 3×3 2D Convolution Engine



[Farabet et al., ICCV 2009]

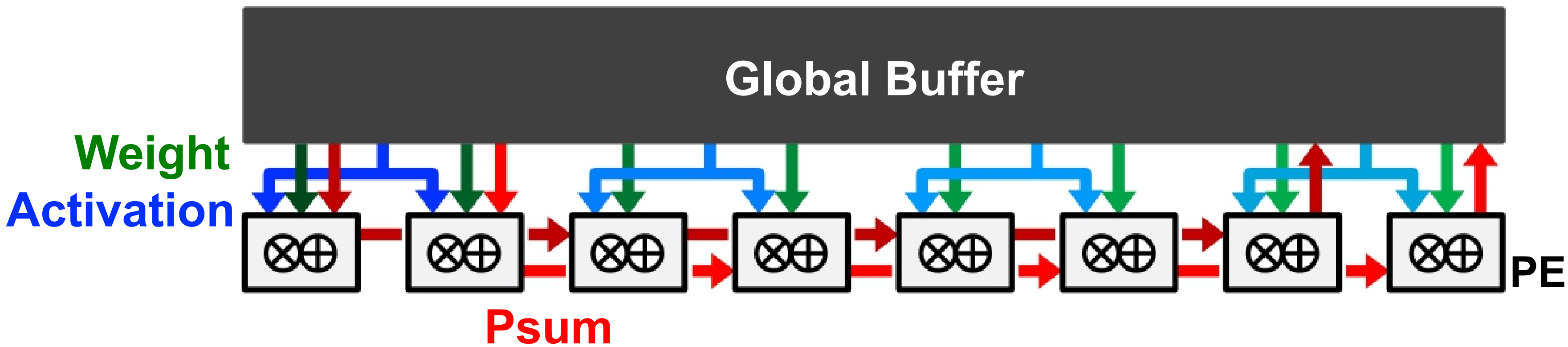
Output Stationary (OS)



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

- **Broadcast/Multicast** **filter weights** and **reuse** **activations** **spatially** across the PE array

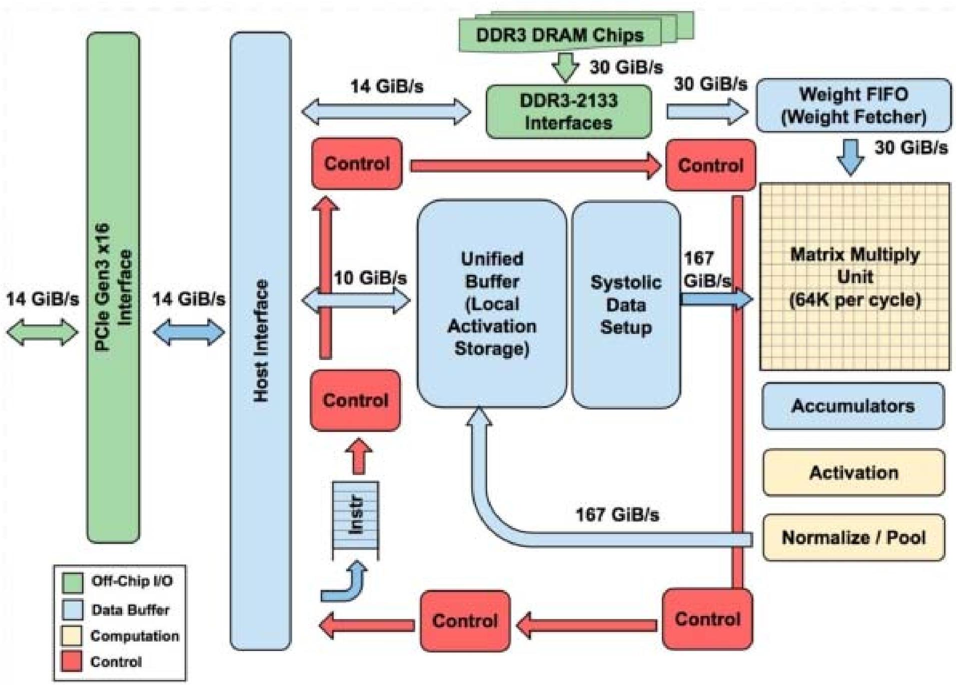
No Local Reuse (NLR)



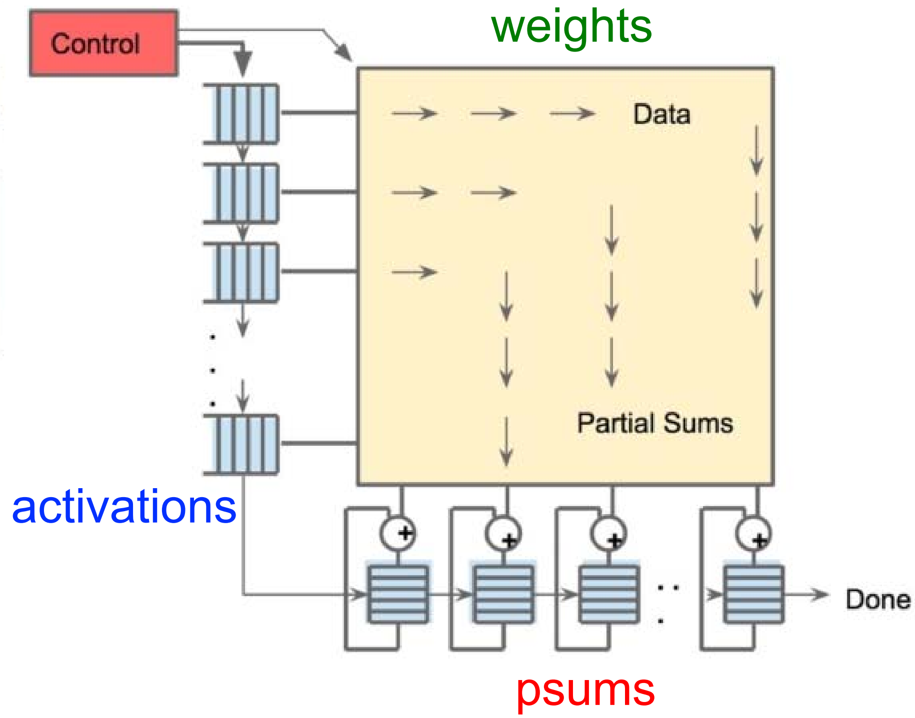
- Use a **large global buffer** as shared storage
  - Reduce **DRAM** access energy consumption
- **Multicast activations, single-cast weights, and accumulate psums spatially** across the PE array

NLR Example: TPU

Top-Level Architecture



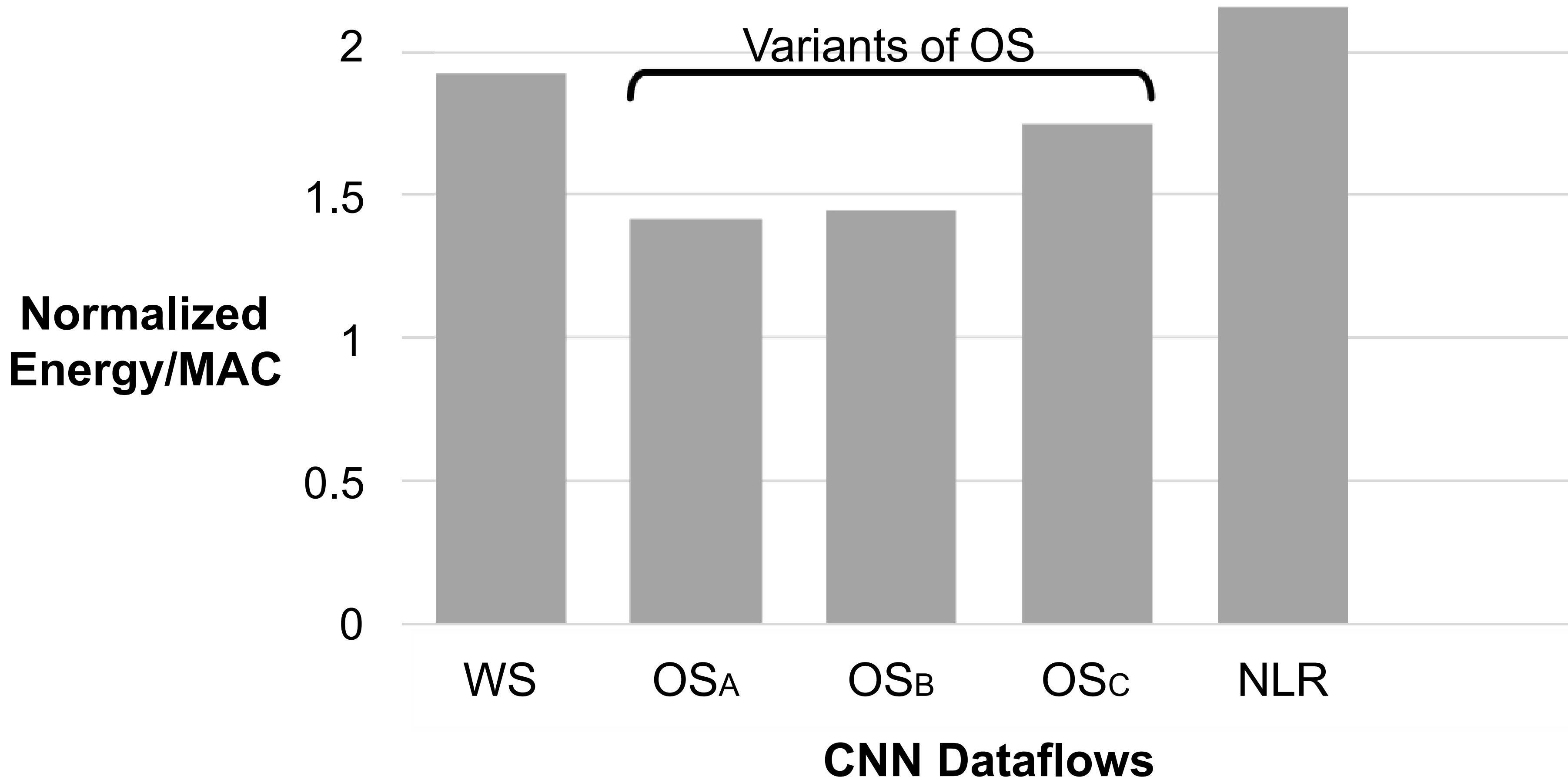
Matrix Multiply Unit



[Jouppi et al., ISCA 2017]

Energy Efficiency Comparison

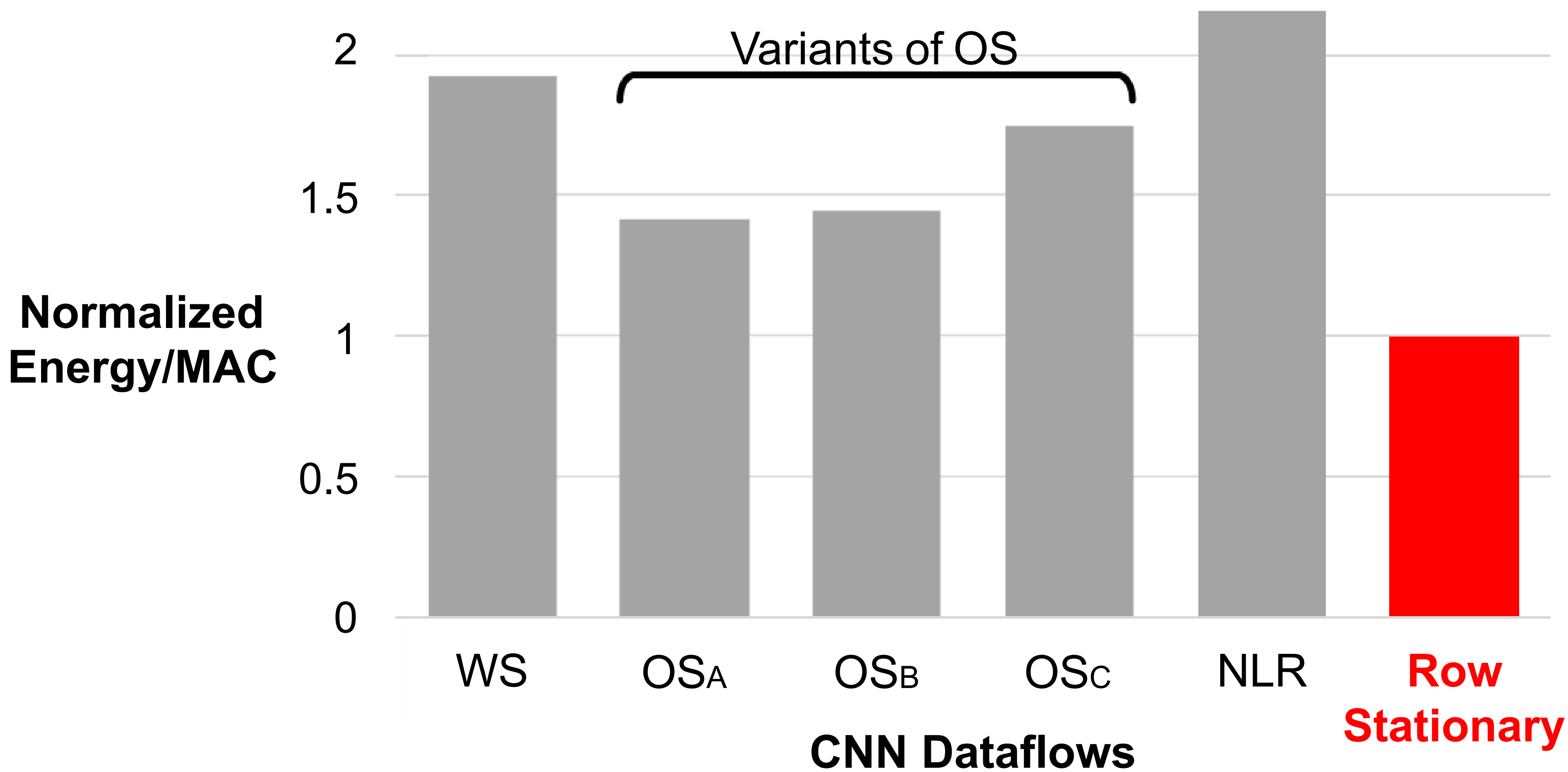
- Same total area
- AlexNet CONV layers
- 256 PEs
- Batch size = 16



[Chen et al., ISCA 2016]

Energy Efficiency Comparison

- Same total area
- AlexNet CONV layers
- 256 PEs
- Batch size = 16



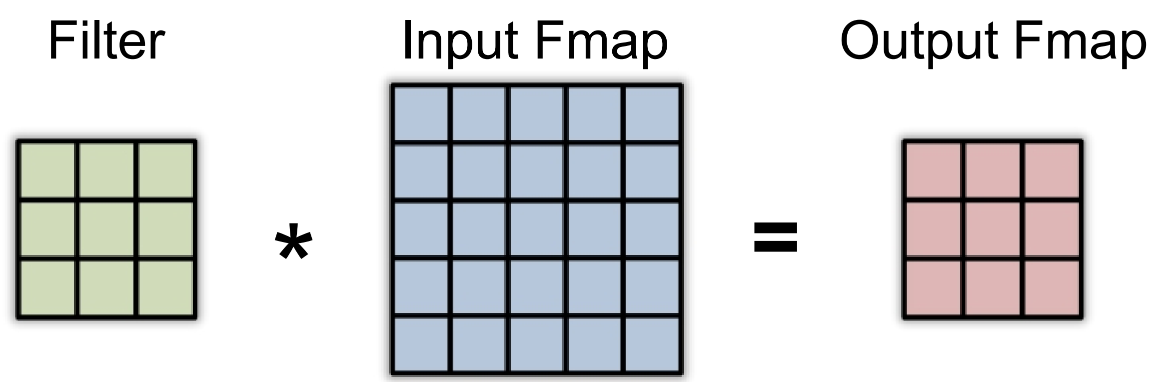
[Chen et al., ISCA 2016]

Energy-Efficient Dataflow:
Row Stationary (RS)

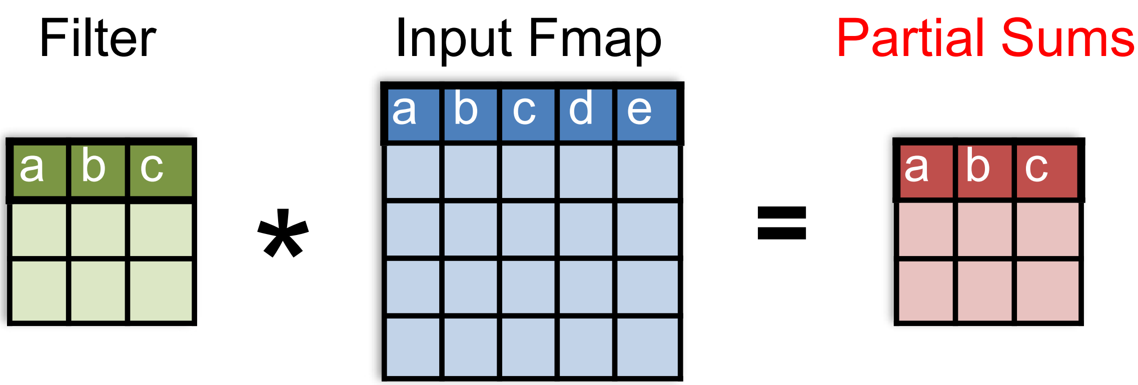
- **Maximize** reuse and accumulation at **RF**
- Optimize for **overall** energy efficiency instead for *only* a certain data type

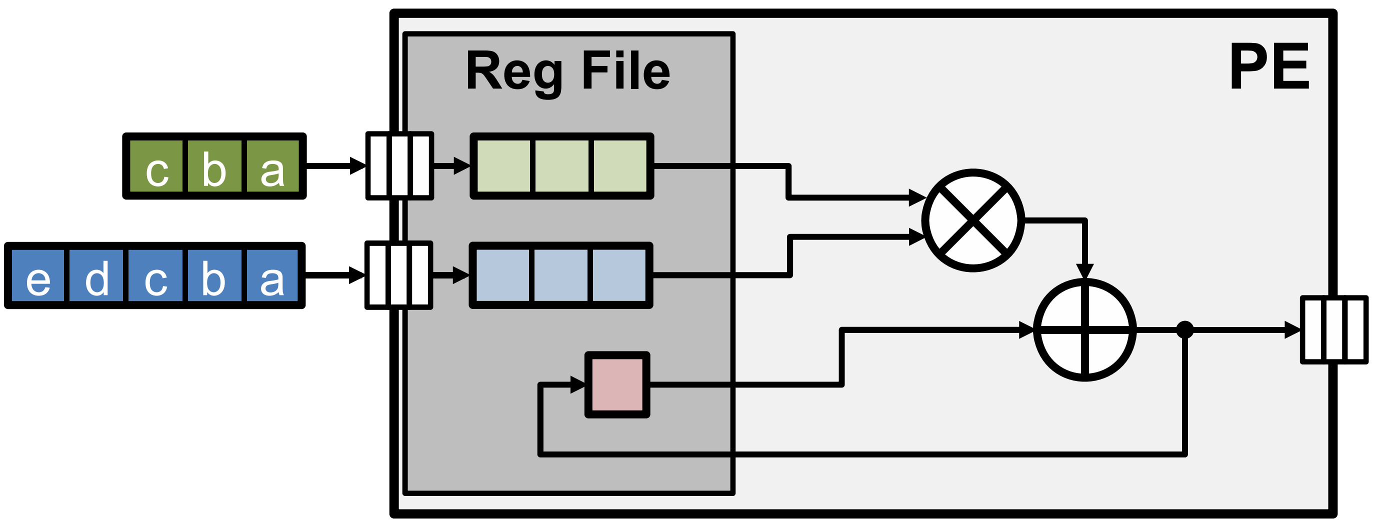
[Chen et al., ISCA 2016]

Row Stationary: Energy-efficient Dataflow

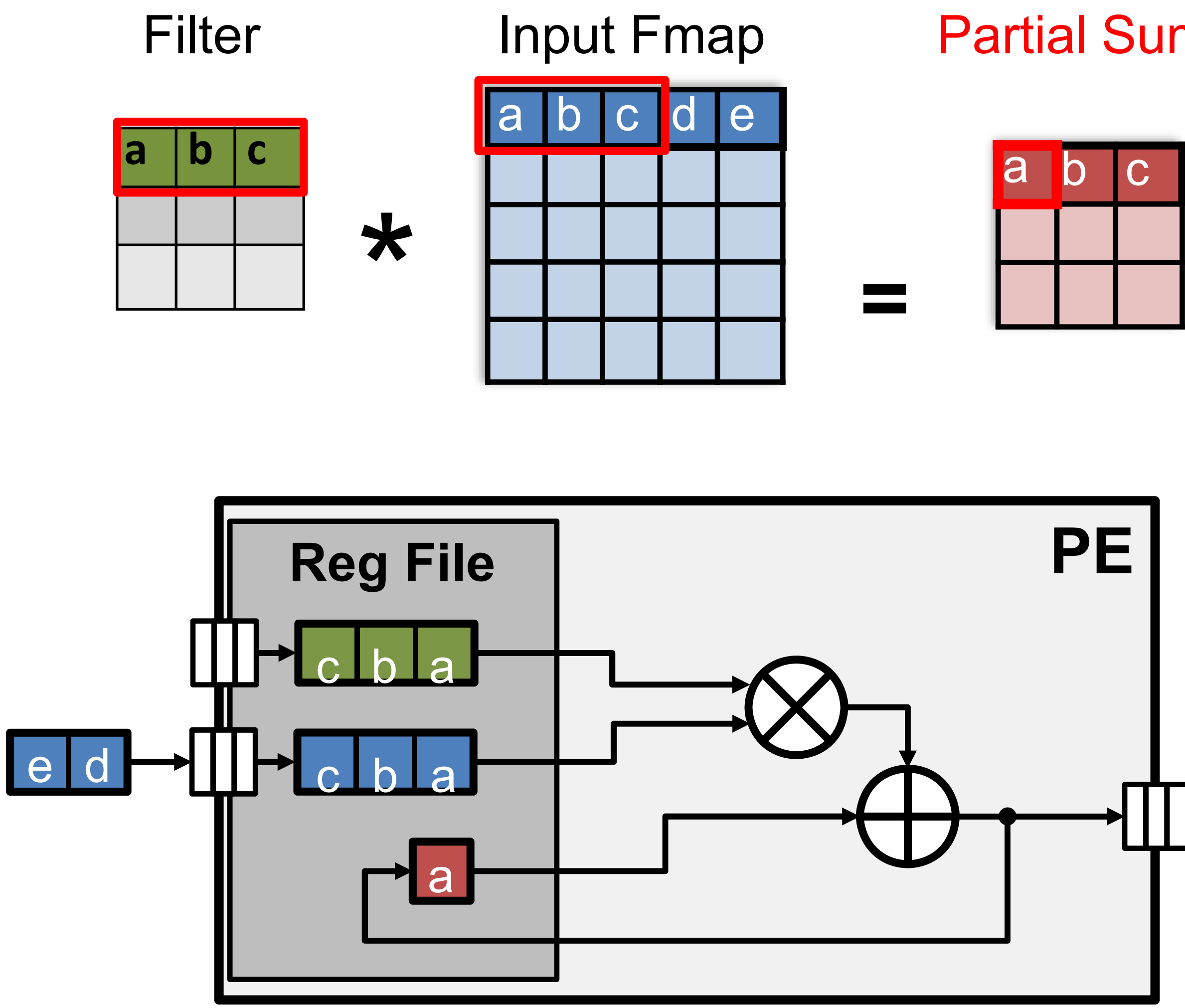


1D Row Convolution in PE

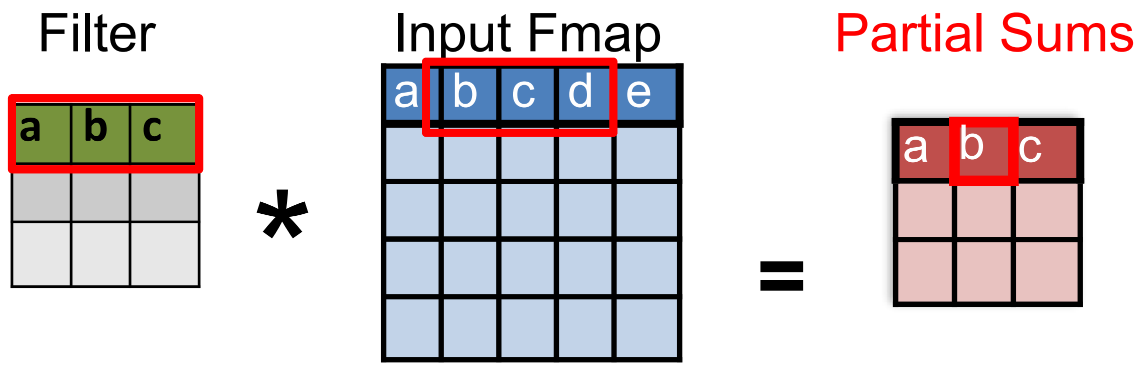


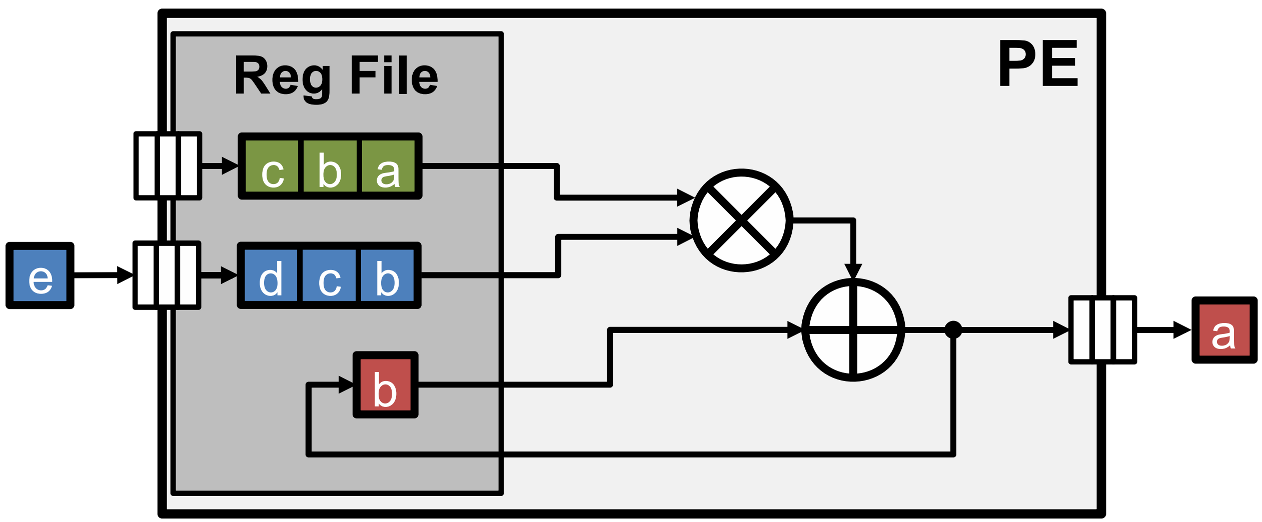


1D Row Convolution in PE

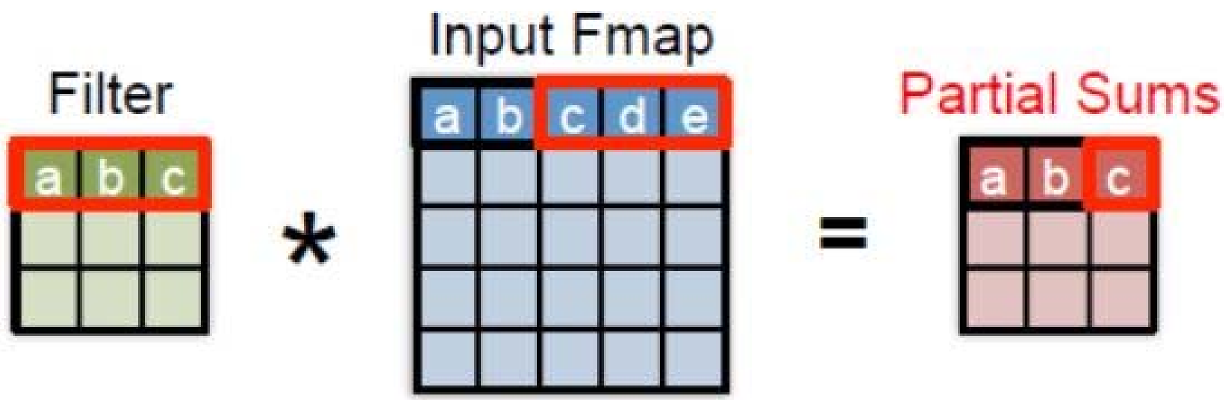


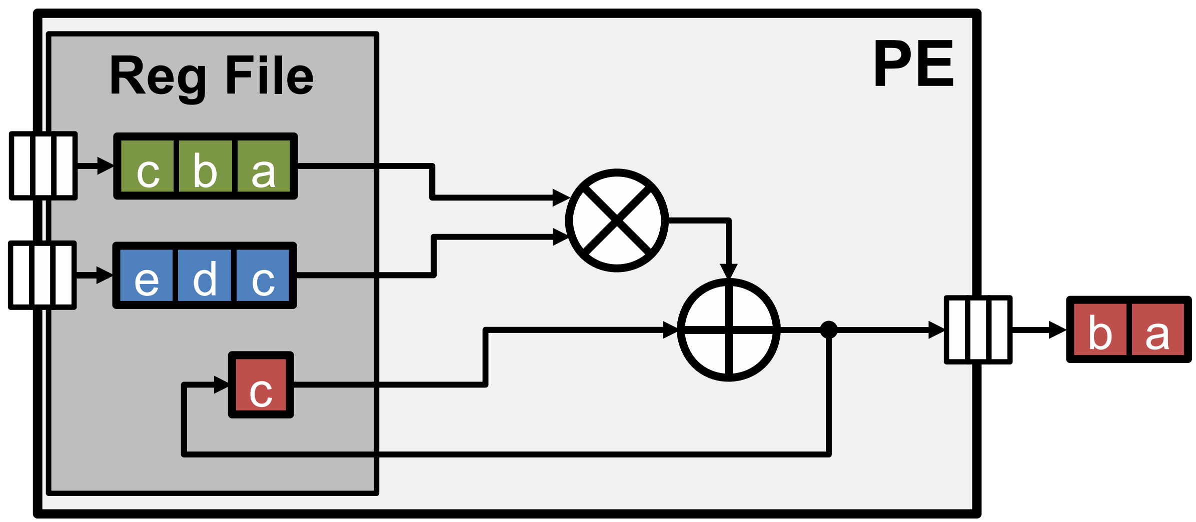
1D Row Convolution in PE





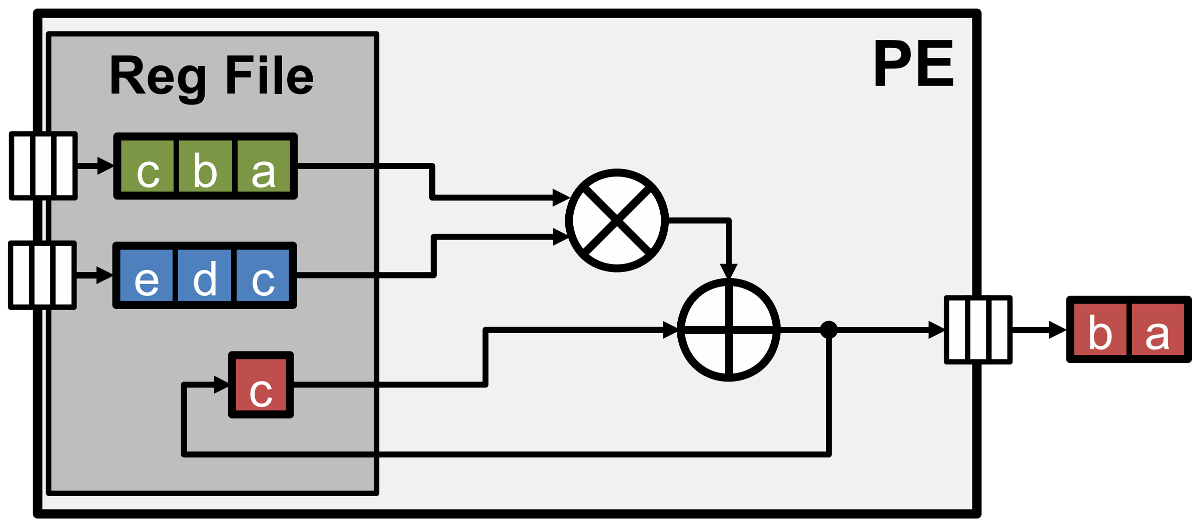
1D Row Convolution in PE



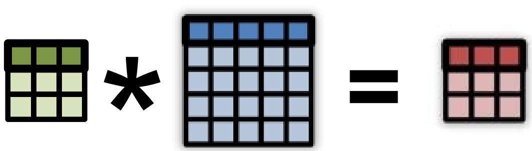


1D Row Convolution in PE

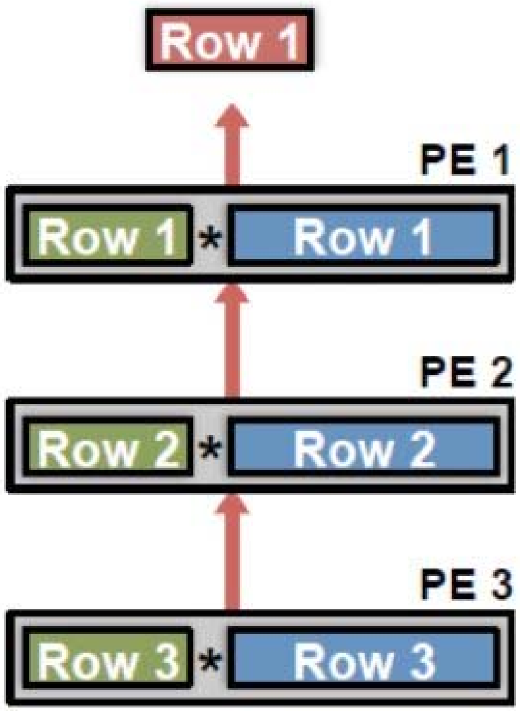
- Maximize row **convolutional reuse** in RF
  - Keep a **filter** row and **fmap** sliding window in RF
- Maximize row **psum** **accumulation** in RF

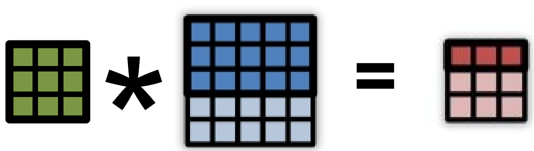


2D Convolution in PE Array

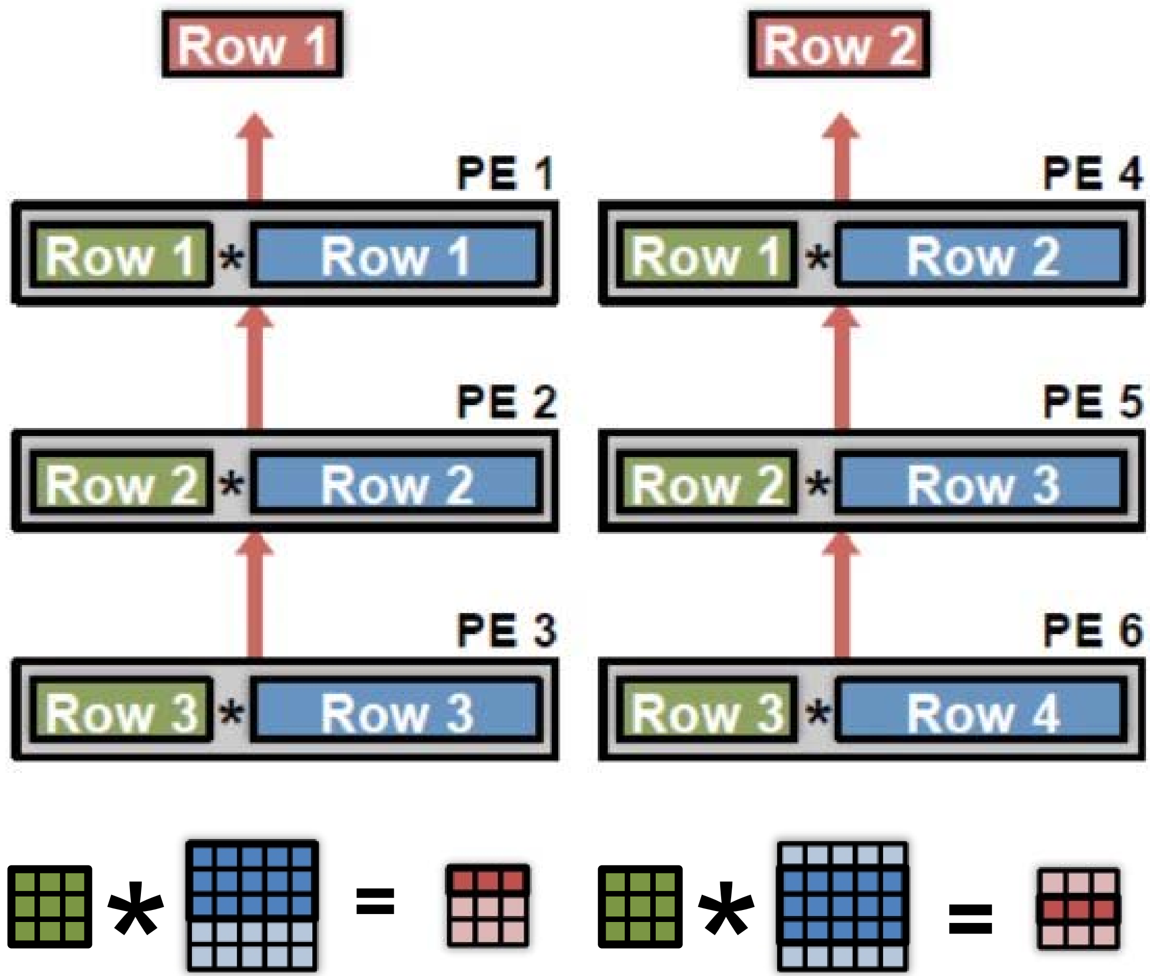


2D Convolution in PE Array

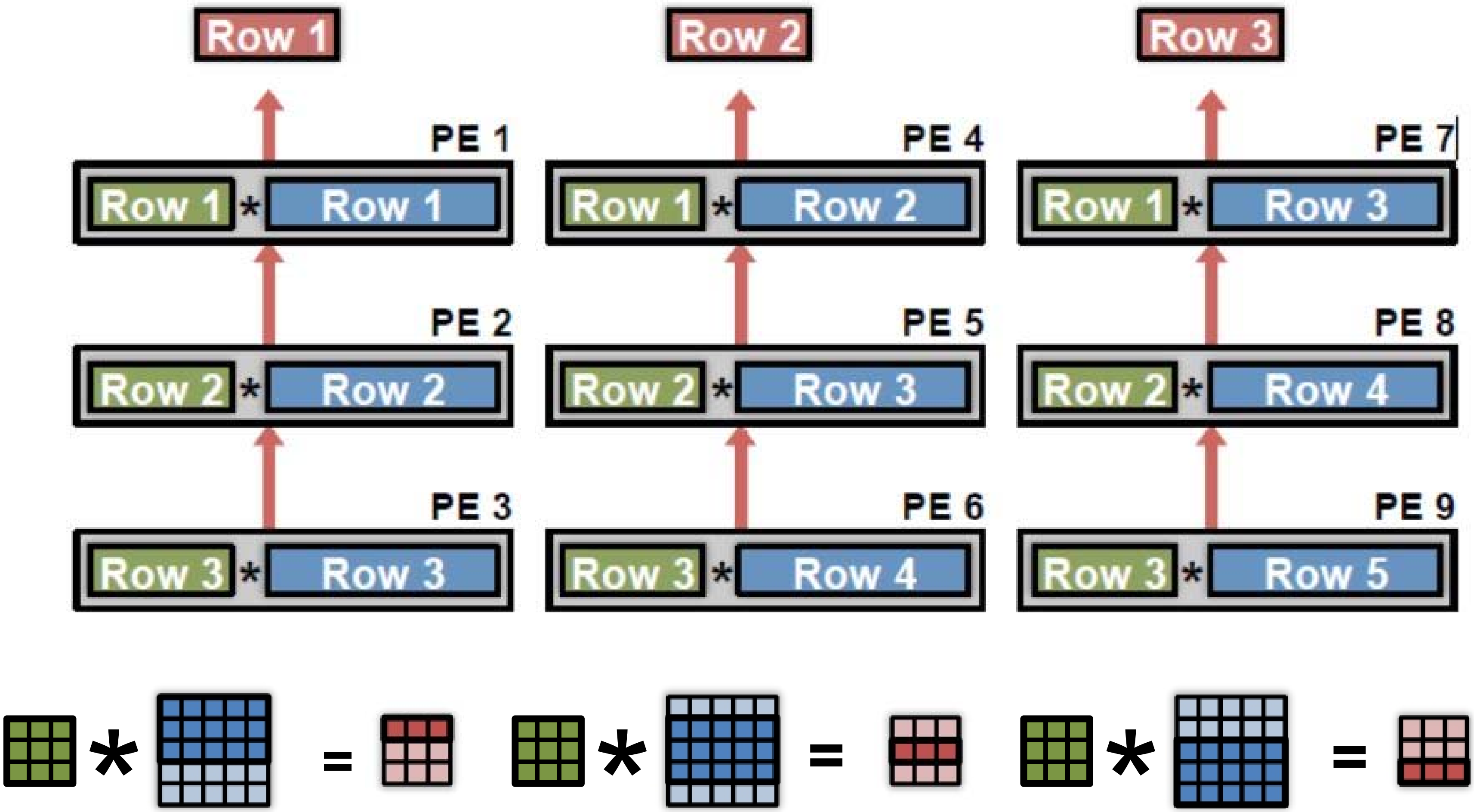




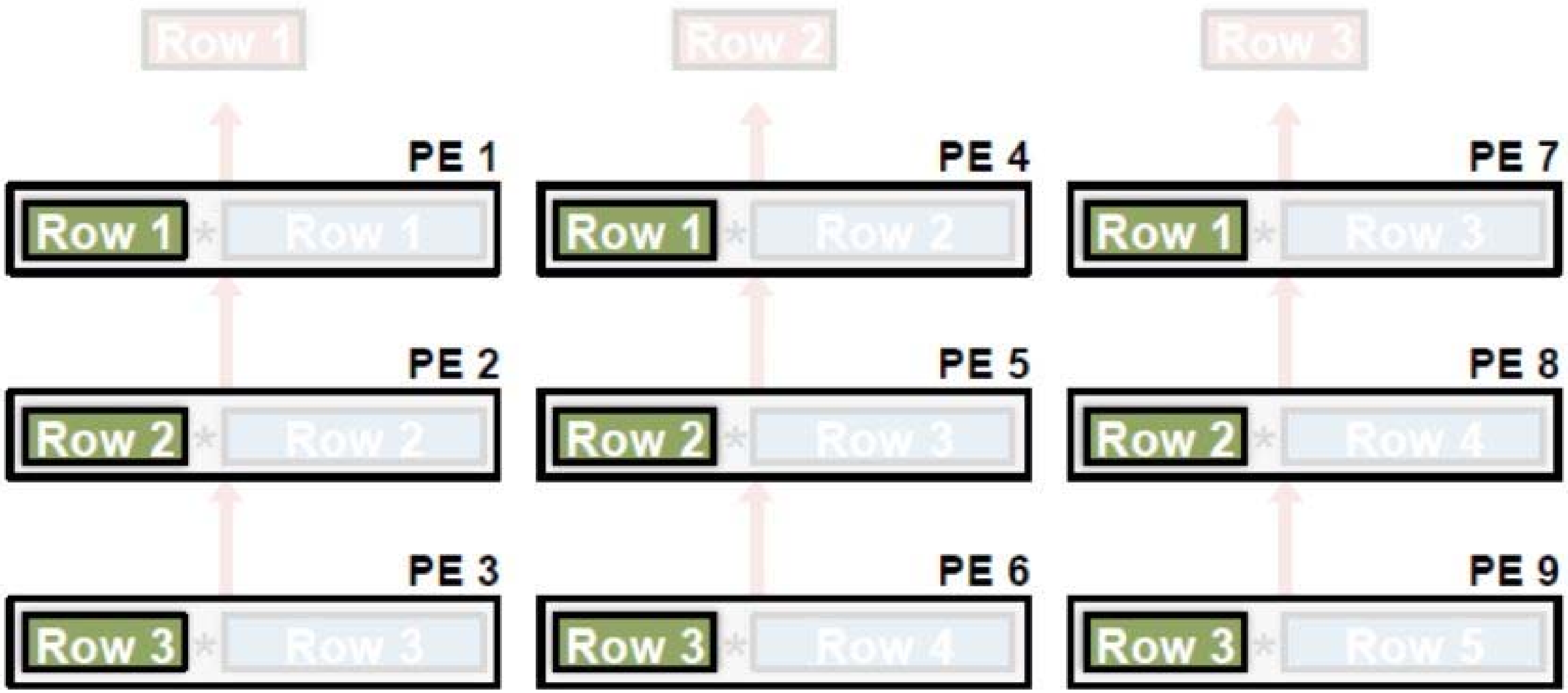
2D Convolution in PE Array



2D Convolution in PE Array

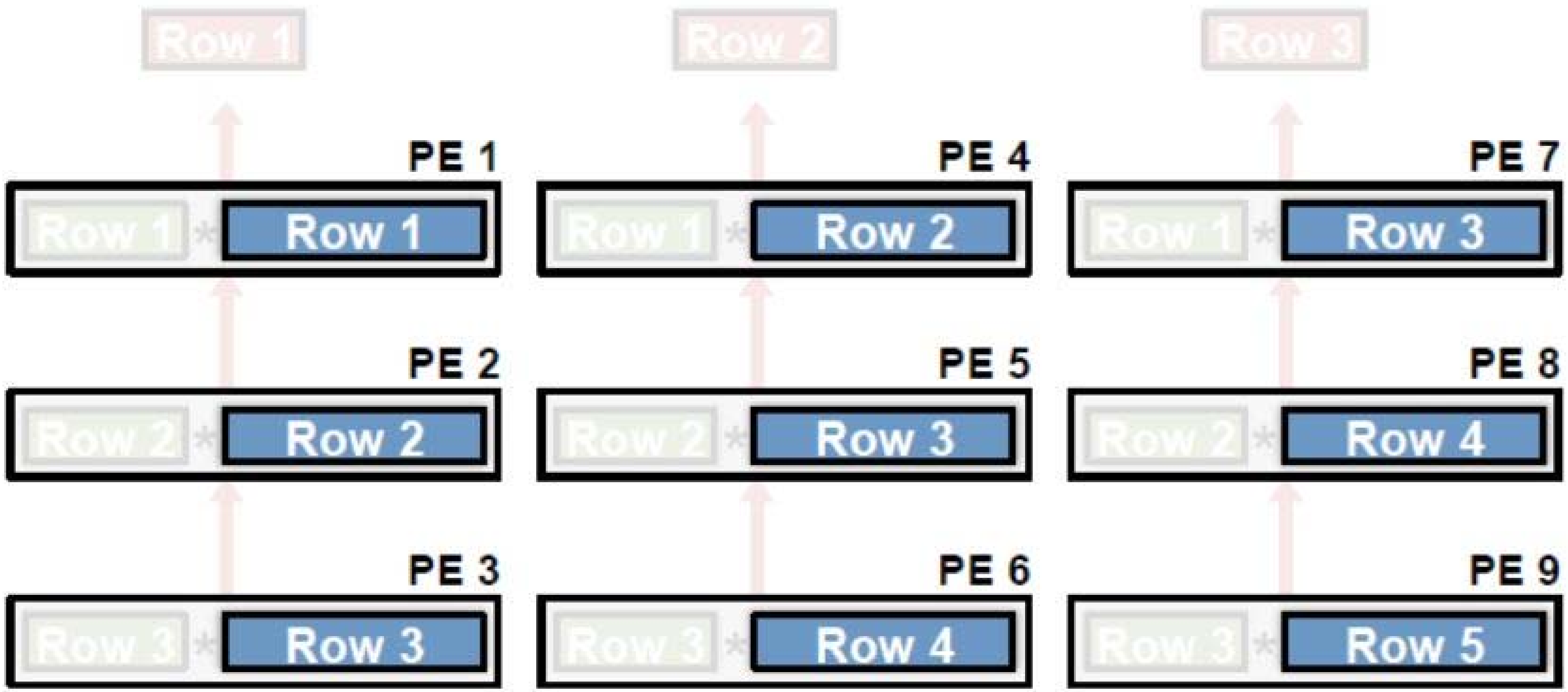


Convolutional Reuse Maximized



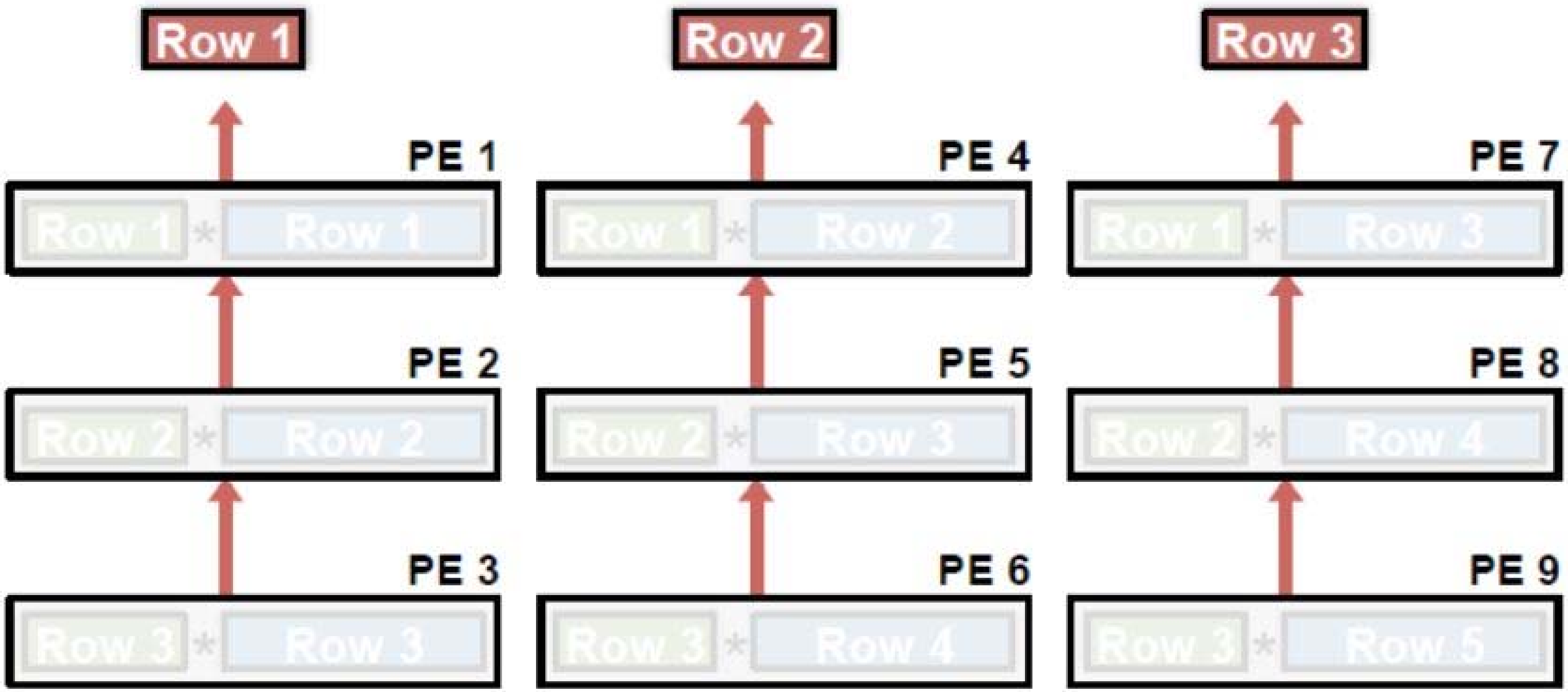
Filter rows are reused across PEs **horizontally**

Convolutional Reuse Maximized



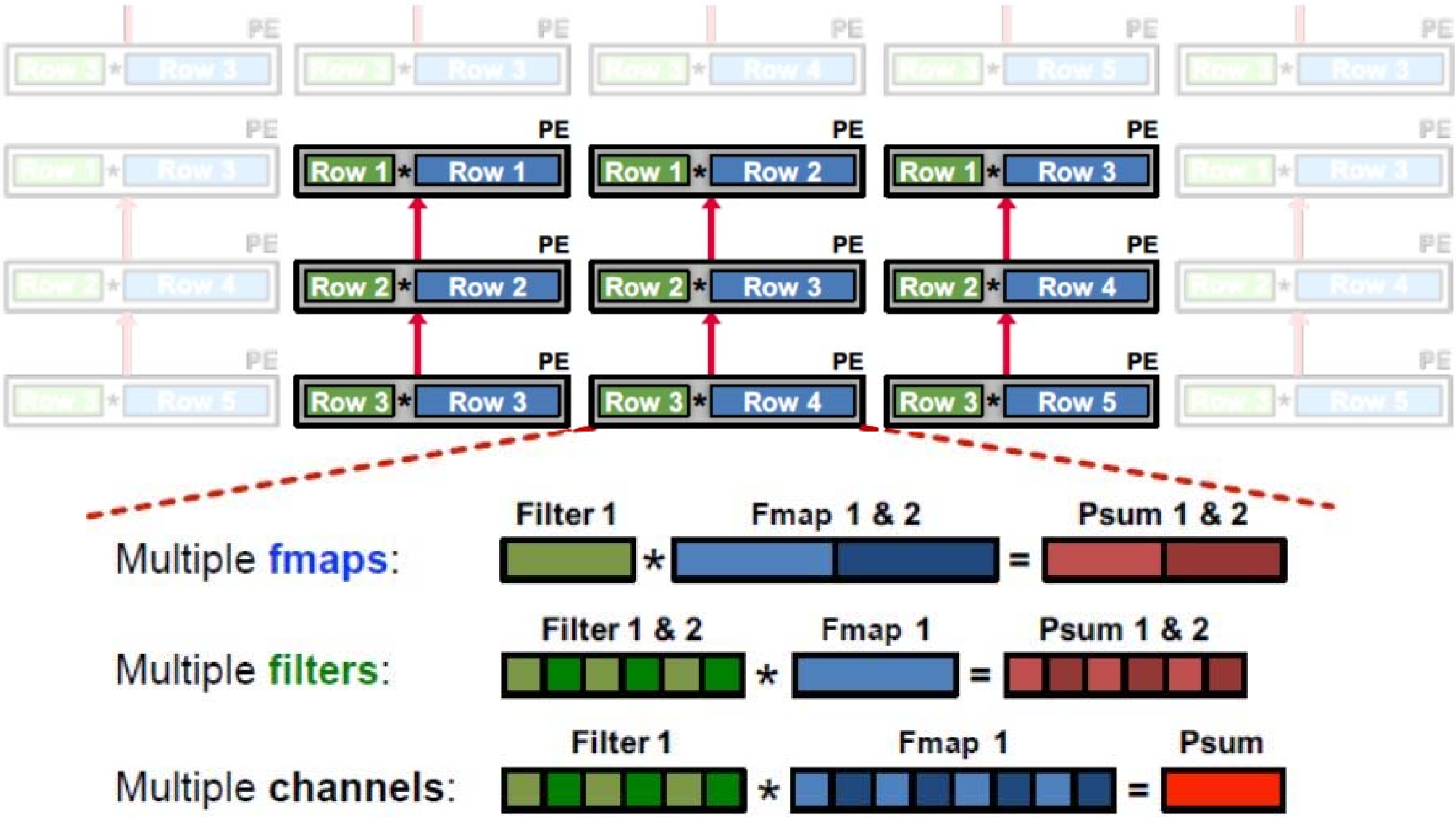
Fmap rows are reused across PEs **diagonally**

Maximize 2D Accumulation in PE Array



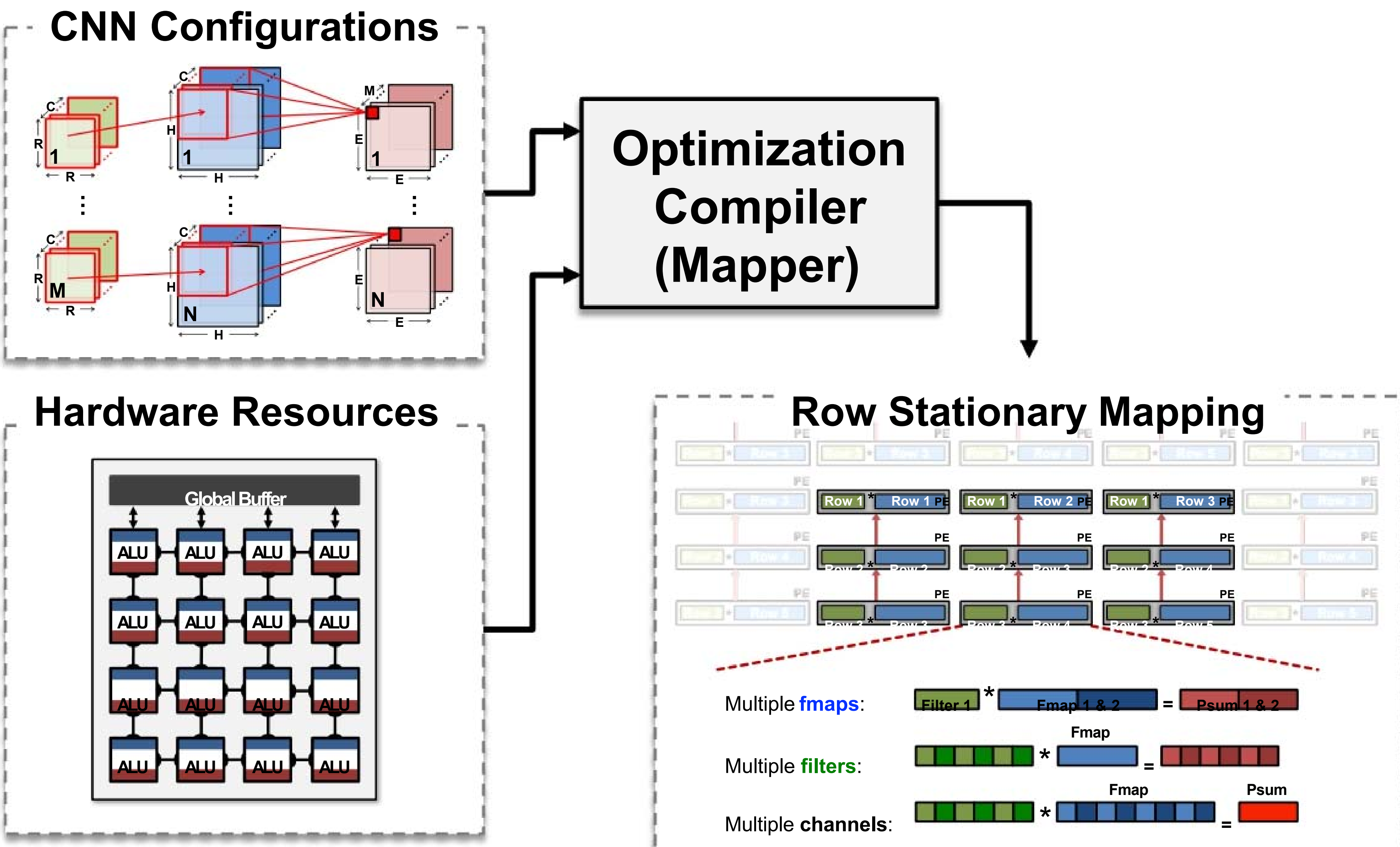
Partial sums accumulate across PEs **vertically**

DNN Processing – The Full Picture



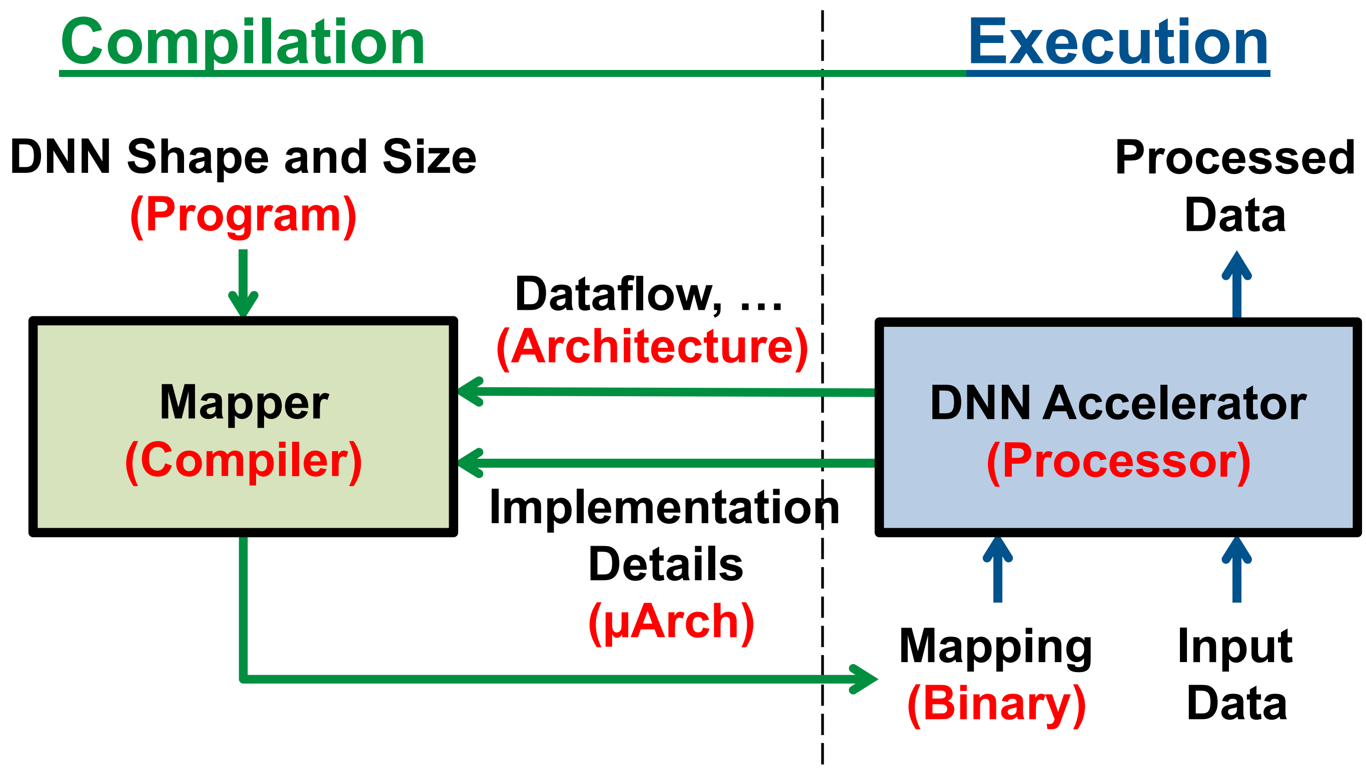
Map rows from **multiple** fmaps, filters and **channels** to same PE
to exploit other forms of reuse and local accumulation

Optimal Mapping in Row Stationary



[Chen et al., ISCA 2016]

Computer Architecture Analogy

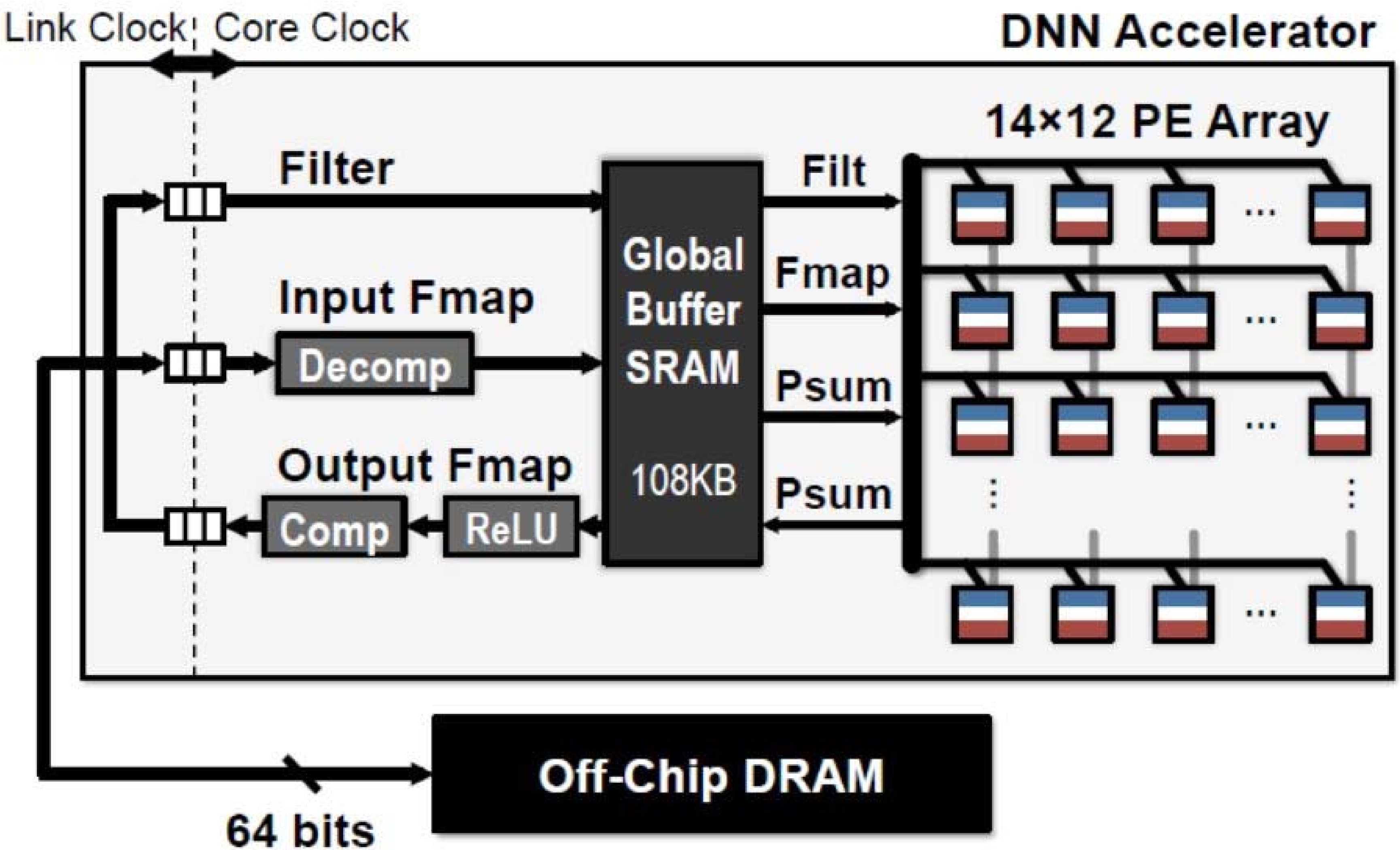


[Chen et al., Micro Top-Picks 2017]

Hardware Architecture for RS Dataflow

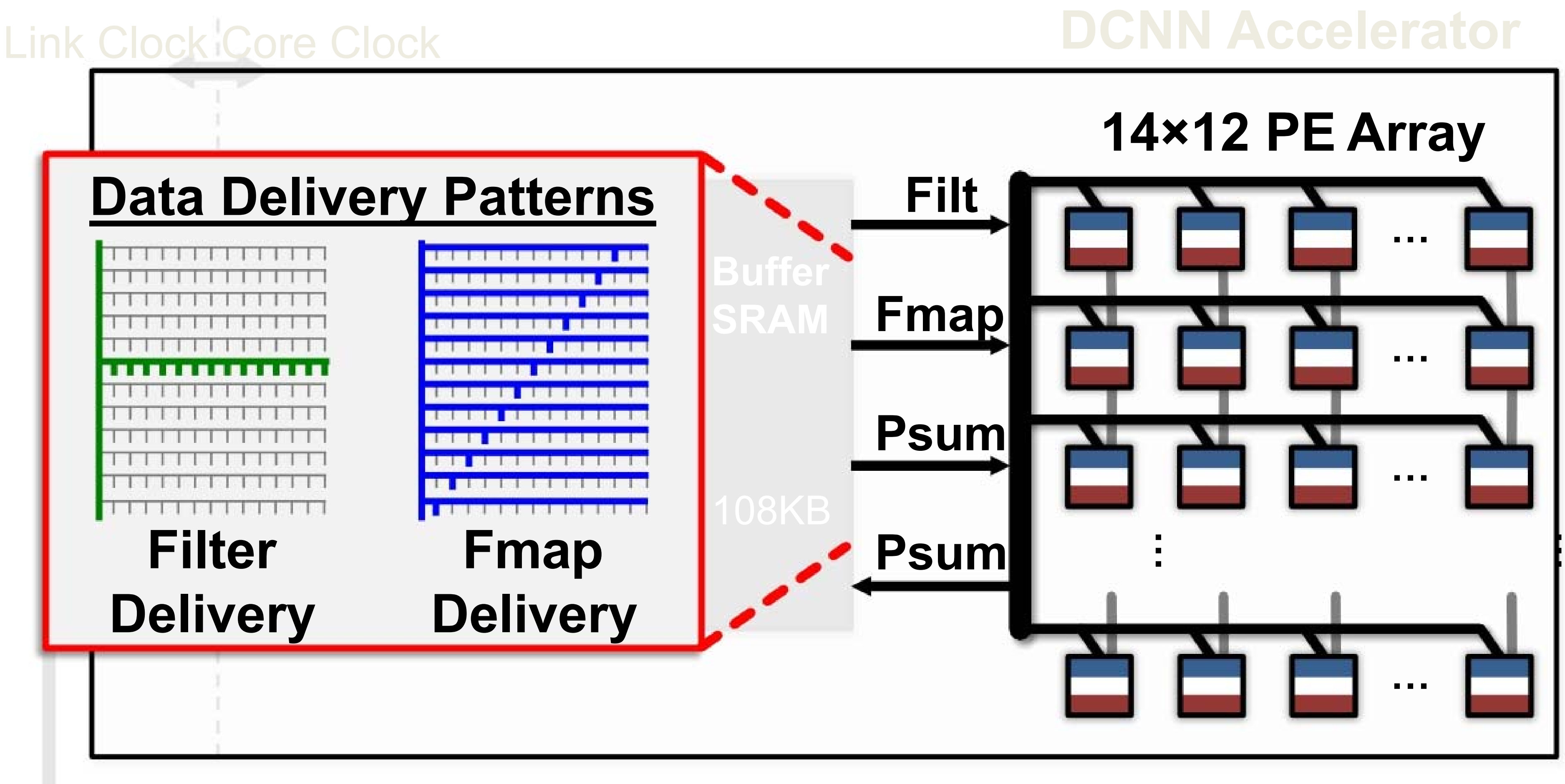
[Chen et al., ISSCC 2016]

Eyeriss DNN Accelerator

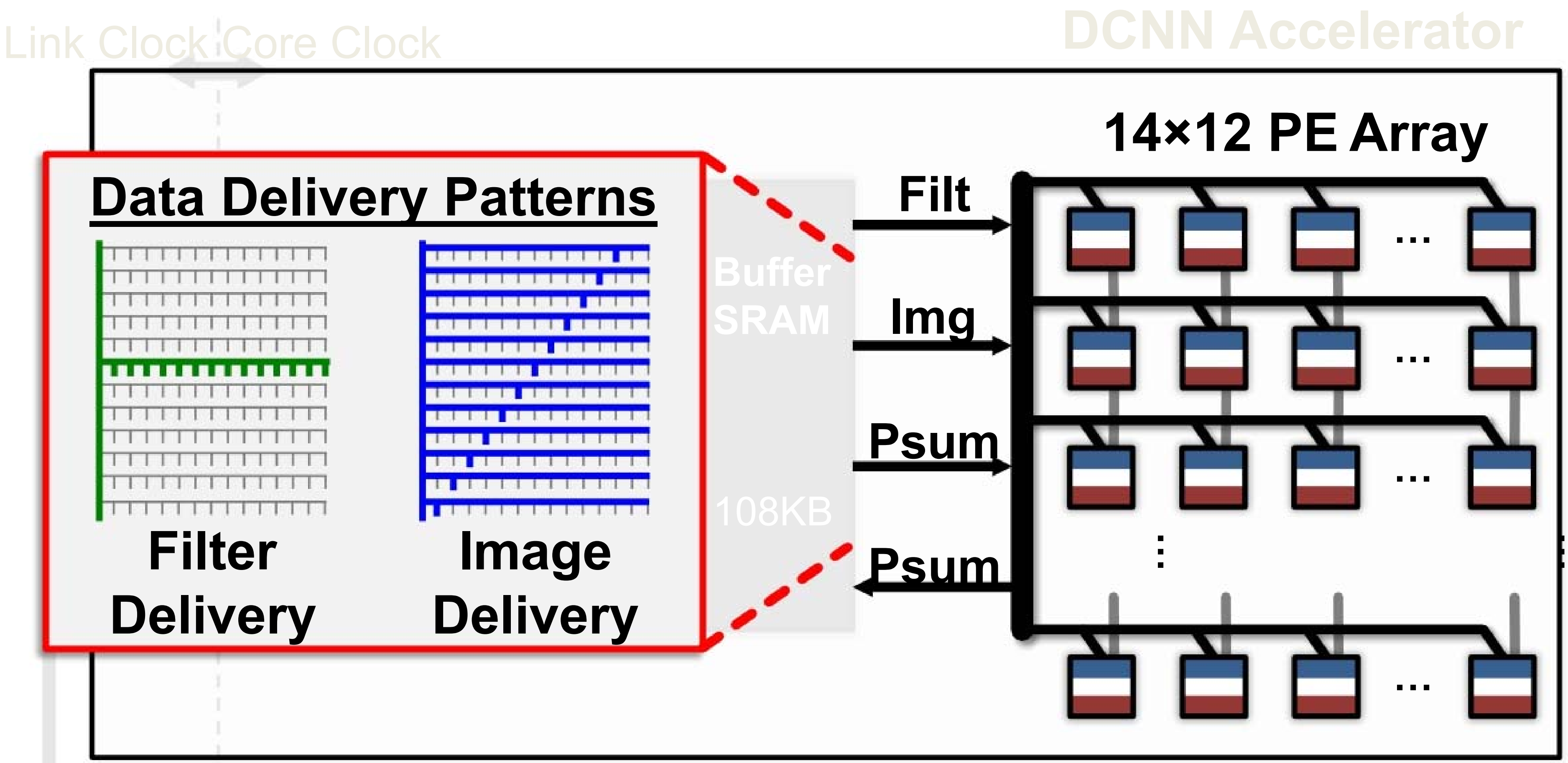


[Chen et al., ISSCC 2016]

Data Delivery with On-Chip Network



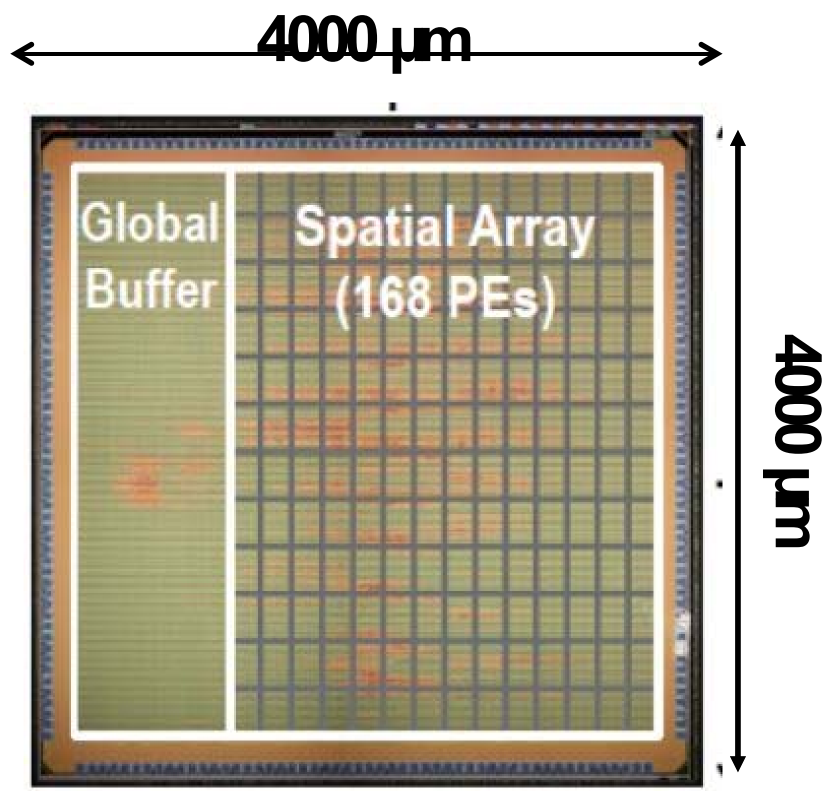
Data Delivery with On-Chip Network



Compared to Broadcast, **Multicast** saves >**80%** of NoC energy

Chip Spec & Measurement Results

Technology	TSMC 65nm LP 1P9M
On-Chip Buffer	108 KB
# of PEs	168
Scratch Pad / PE	0.5 KB
Core Frequency	100 – 250 MHz
Peak Performance	33.6 – 84.0 GOPS
Word Bit-width	16-bit Fixed-Point
Natively Supported DNN Shapes	Filter Width: 1 – 32 Filter Height: 1 – 12 Num. Filters: 1 – 1024 Num. Channels: 1 – 1024 Horz. Stride: 1–12 Vert. Stride: 1, 2, 4



To support 2.66 GMACs [8 billion 16-bit inputs (**16GB**) and 2.7 billion outputs (**5.4GB**)], only requires **208.5MB** (buffer) and **15.4MB** (DRAM)

[Chen et al., ISSCC 2016]

Summary of DNN Dataflows

- **Weight Stationary**
  - Minimize movement of filter weights
  - Popular with processing-in-memory architectures
- **Output Stationary**
  - Minimize movement of partial sums
  - Different variants optimized for CONV or FC layers
- **No Local Reuse**
  - No PE local storage → maximize global buffer size
- **Row Stationary**
  - Adapt to the NN shape and hardware constraints – Optimized for overall **system energy efficiency**

Backup Slides