Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks

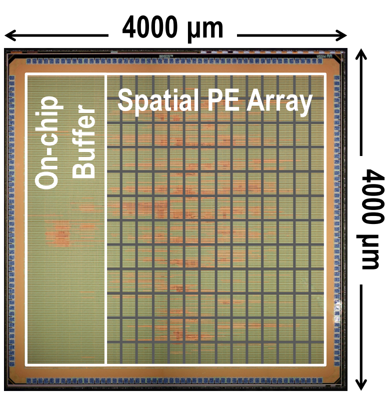
Yu-Hsin Chen[1], Joel Emer[1, 2], Vivienne Sze[1]

[1] MIT [2] NVIDIA

Massachusetts Institute of Technology

Contributions of This Work

- A novel **energy-efficient CNN dataflow** that has been verified in a fabricated chip, ***Eyeriss***.
- A **taxonomy of CNN dataflows** that classifies previous work into three categories.
- A **framework** that compares the energy efficiency of different dataflows under same area and CNN setup.



Eyeriss [*ISSCC*, 2016]

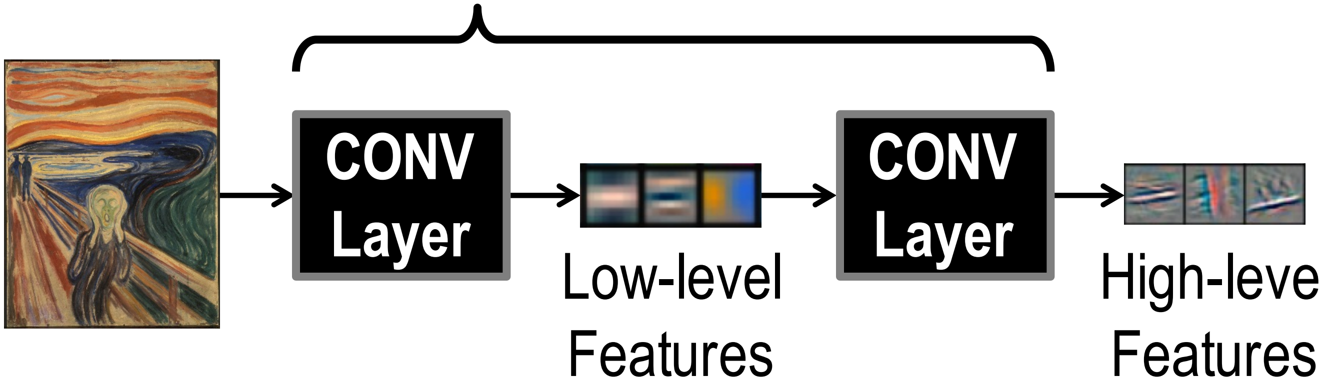
A reconfigurable CNN processor

35 fps @ 278 mW*

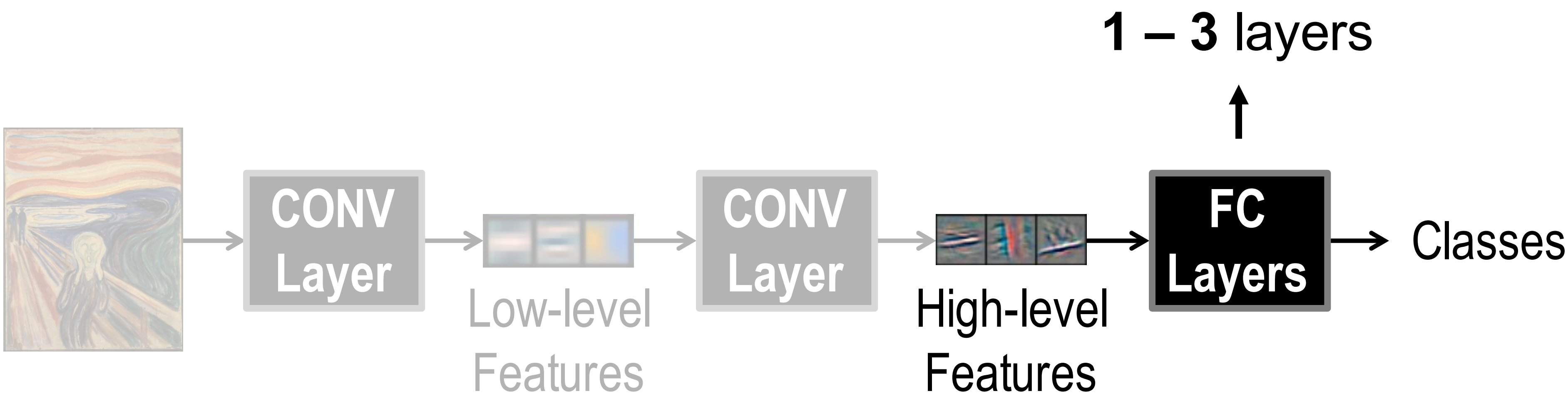
* AlexNet CONV layers

Deep Convolutional Neural Networks

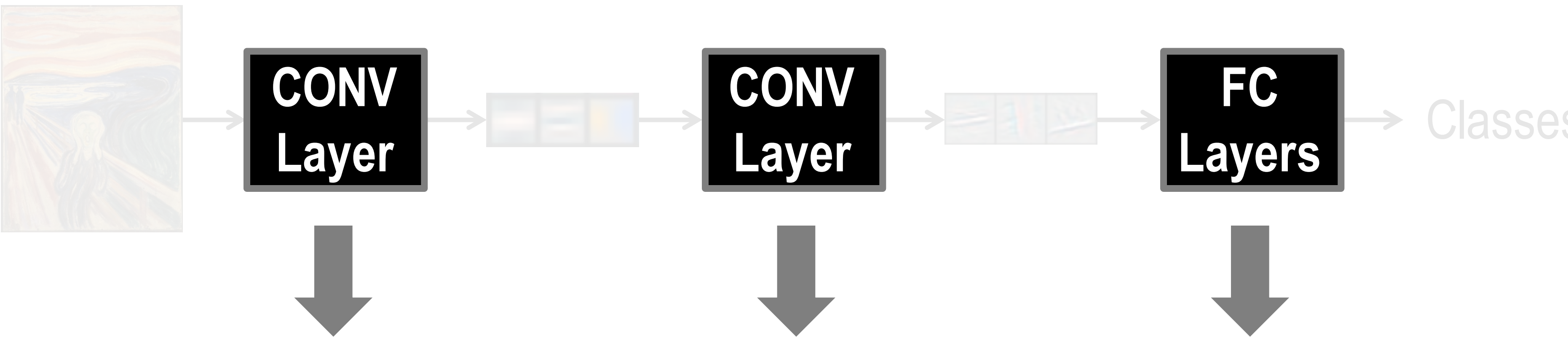
Modern ***deep*** CNN: up to **1000** CONV layers

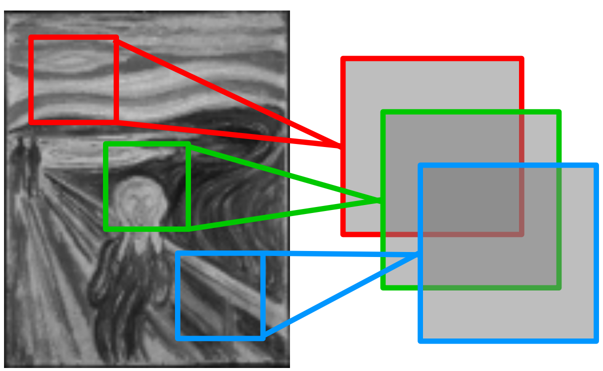


Deep Convolutional Neural Networks



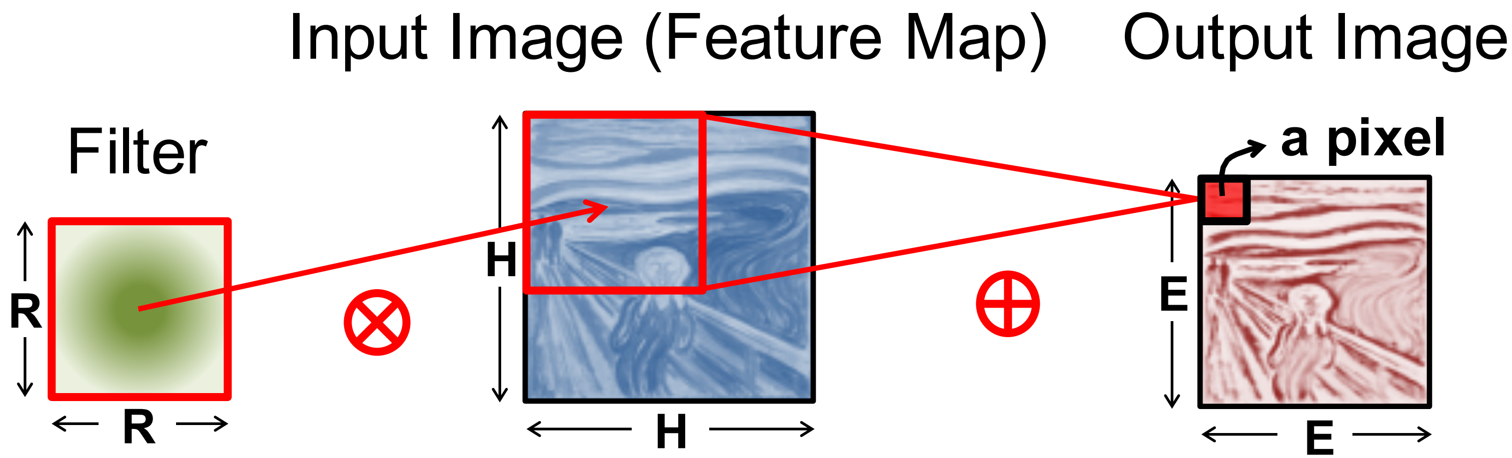
Deep Convolutional Neural Networks





Convolutions account for more than 90% of overall computation, dominating **runtime** and **energy consumption**

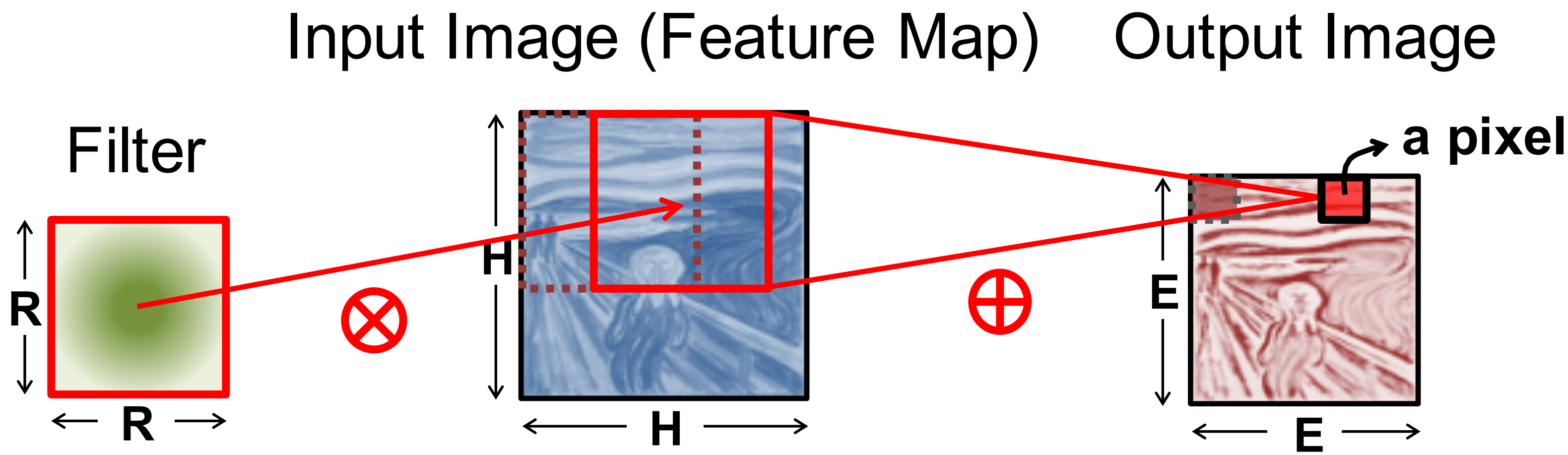
High-Dimensional CNN Convolution



Element-wise Multiplication

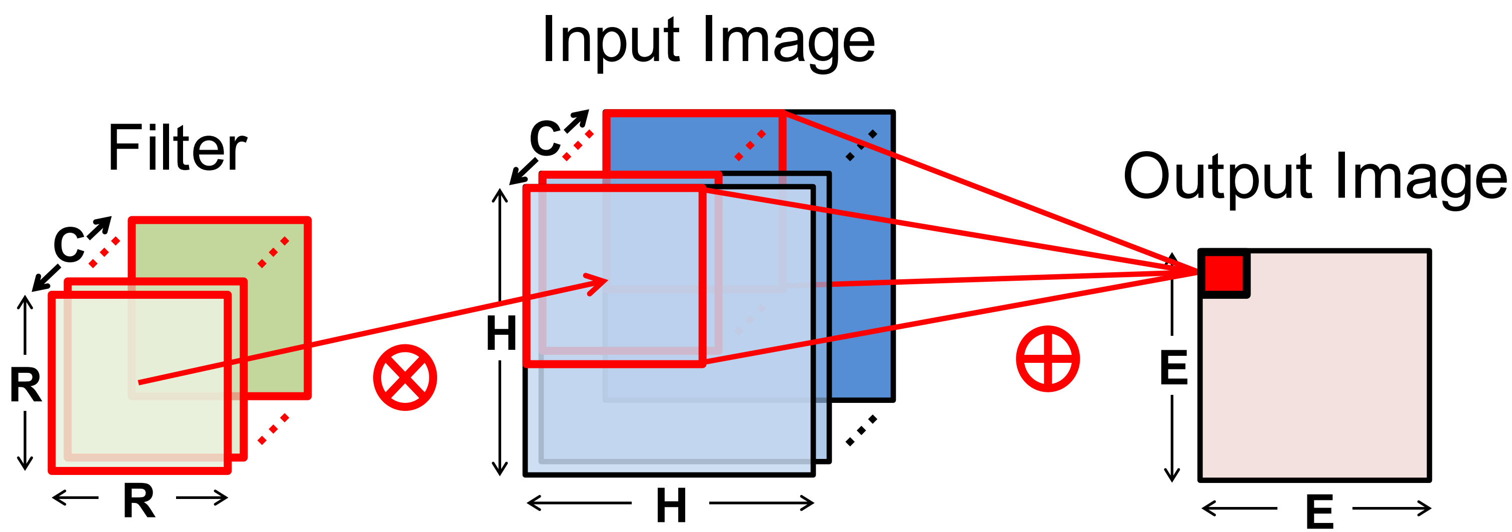
Partial Sum (psum) **Accumulation**

High-Dimensional CNN Convolution



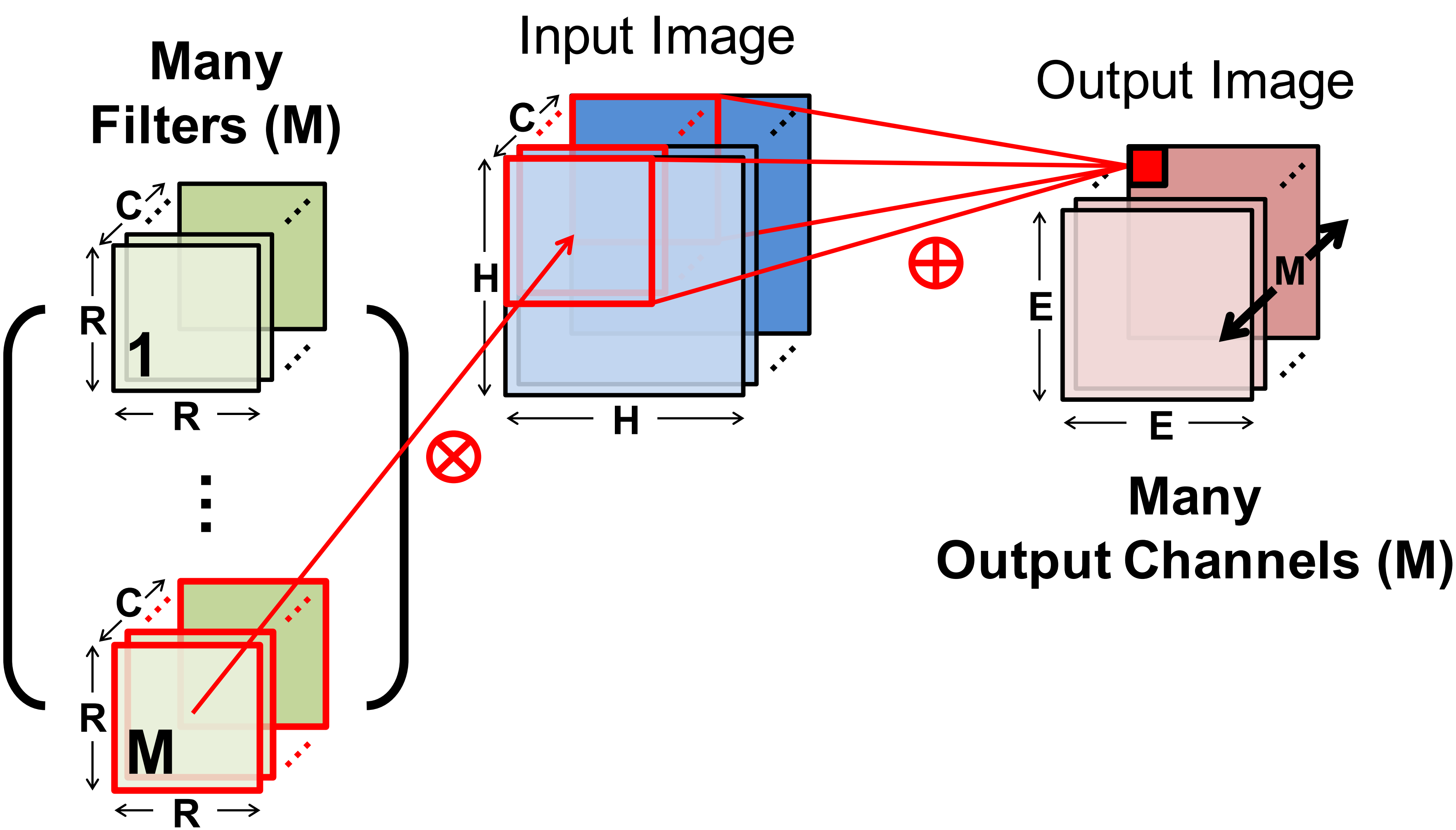
Sliding Window Processing

High-Dimensional CNN Convolution

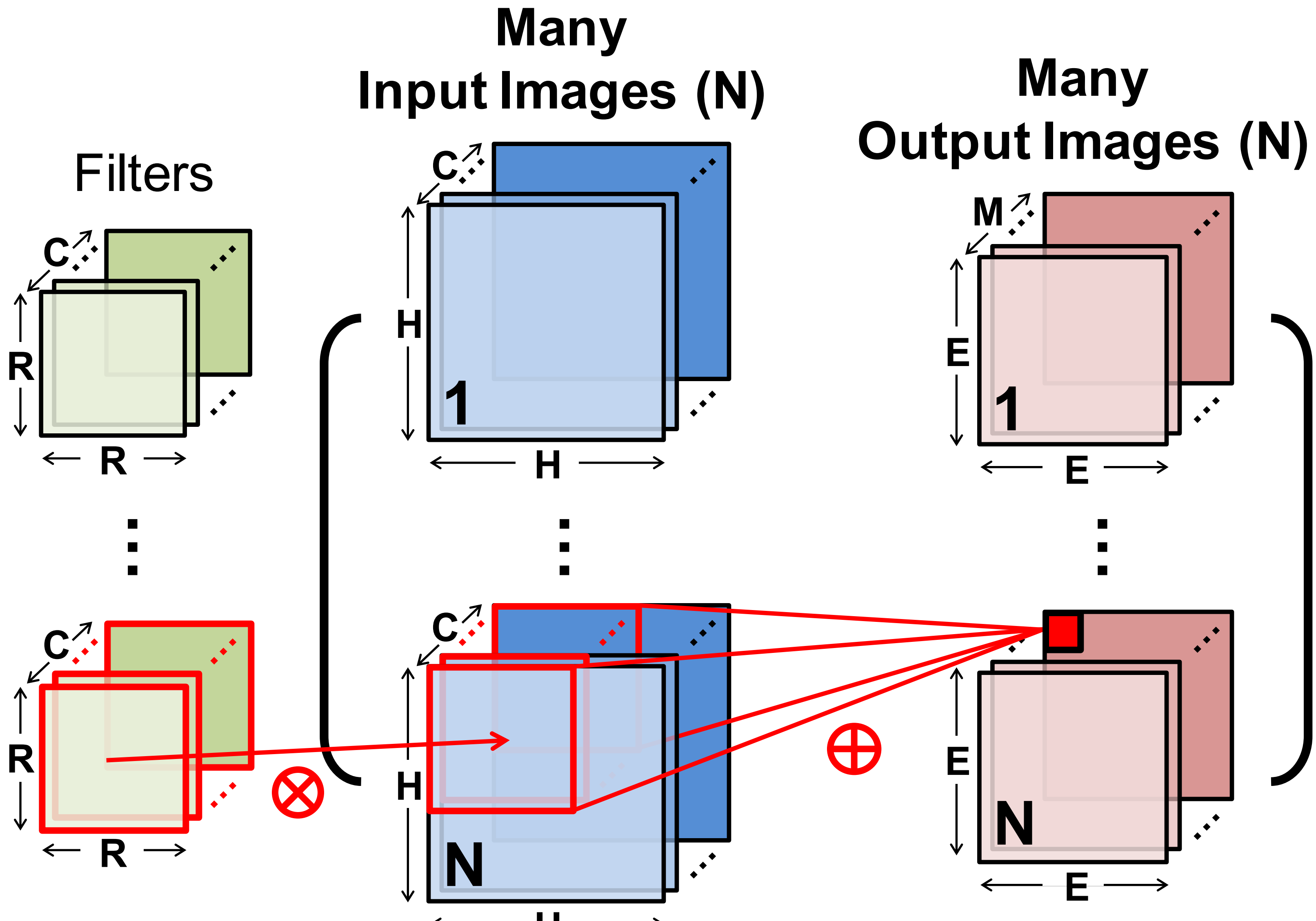


Many Input Channels (C)

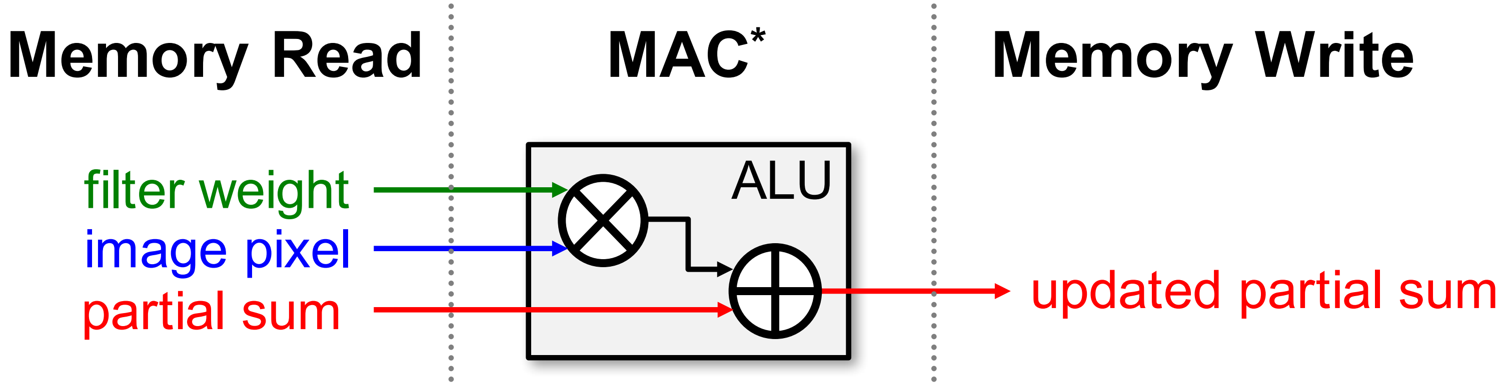
High-Dimensional CNN Convolution



High-Dimensional CNN Convolution

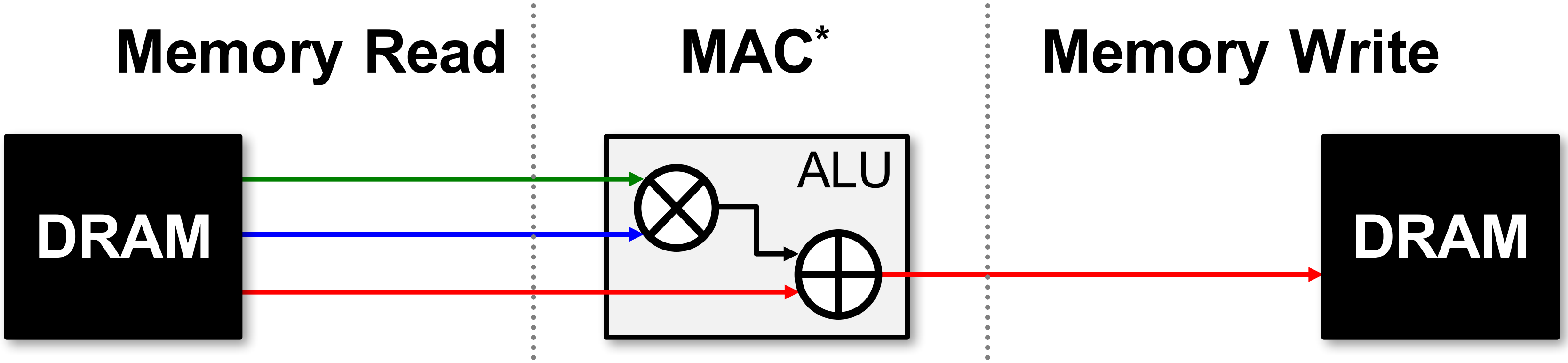


Memory Access is the Bottleneck



* multiply-and-accumulate

Memory Access is the Bottleneck

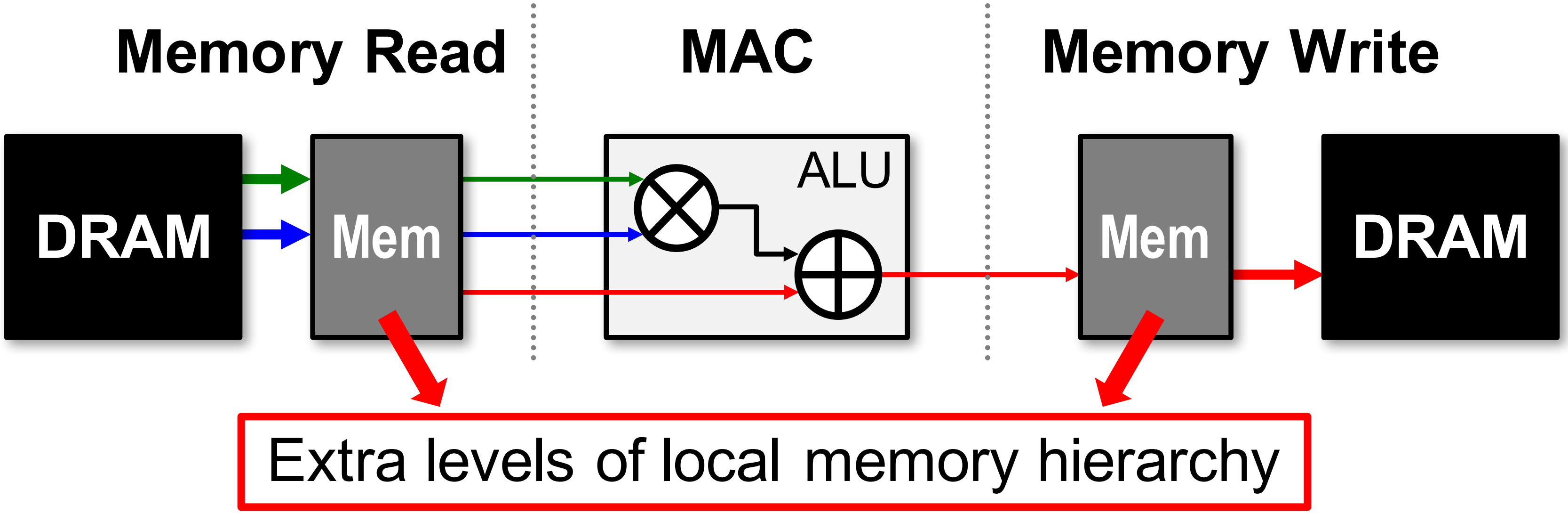


* multiply-and-accumulate

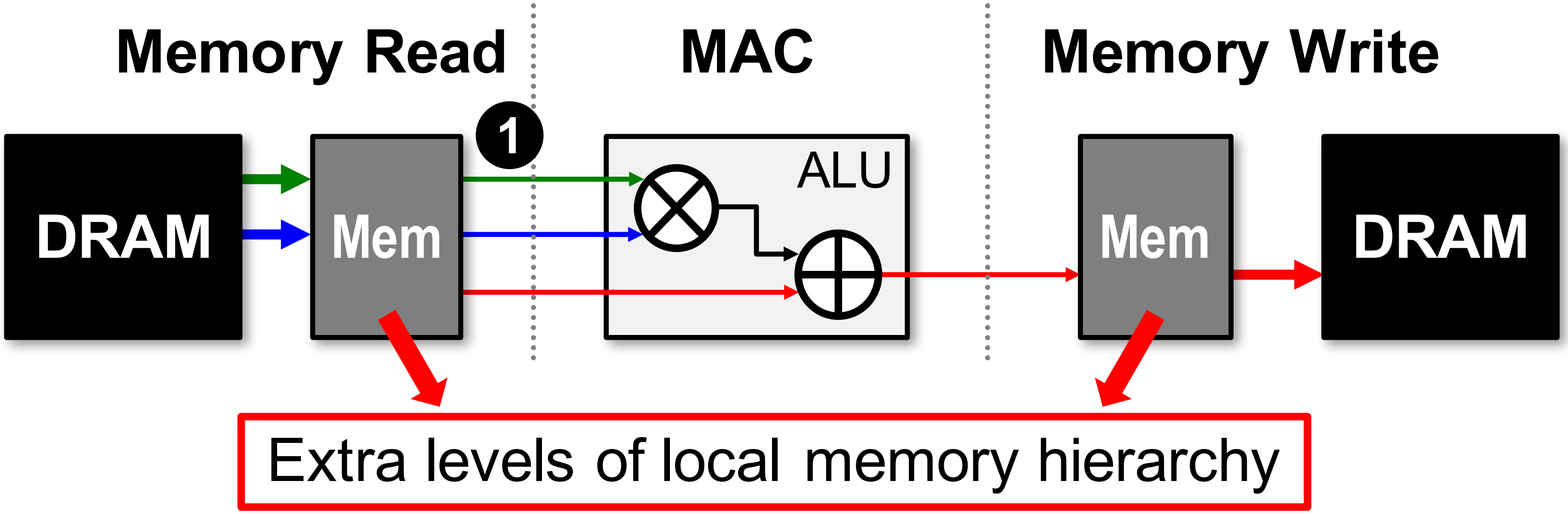
<u>Worst Case</u>: all memory R/W are **DRAM** accesses

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

Memory Access is the Bottleneck



Memory Access is the Bottleneck

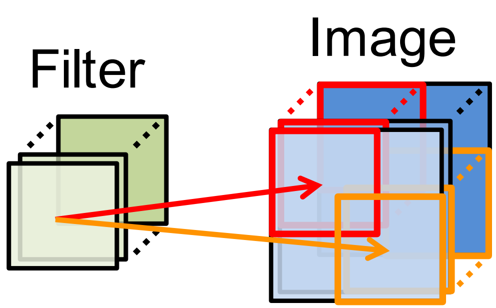


Opportunities: 1 **data reuse**

Types of Data Reuse in CNN

Convolutional Reuse

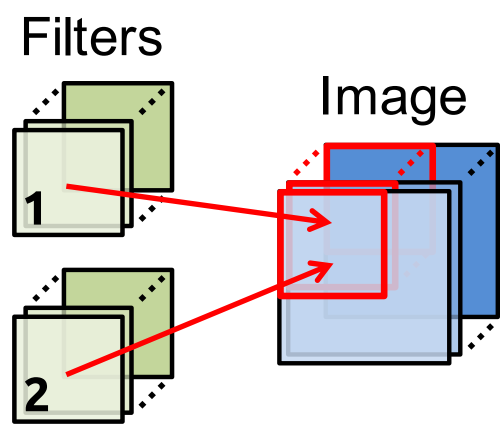
CONV layers only
(sliding window)



Reuse: Image pixels
Filter weights

Image Reuse

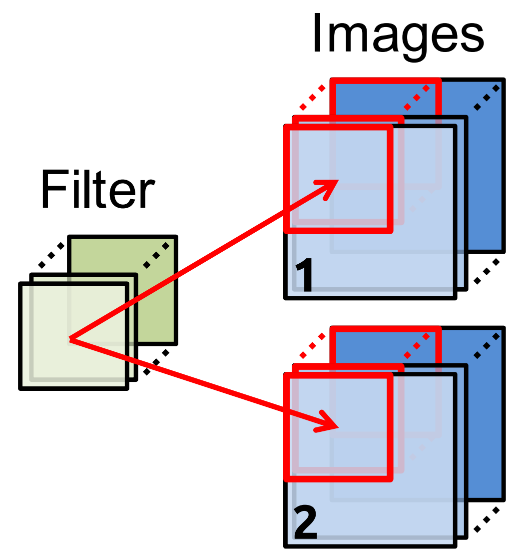
CONV and FC layers



Reuse: Image pixels

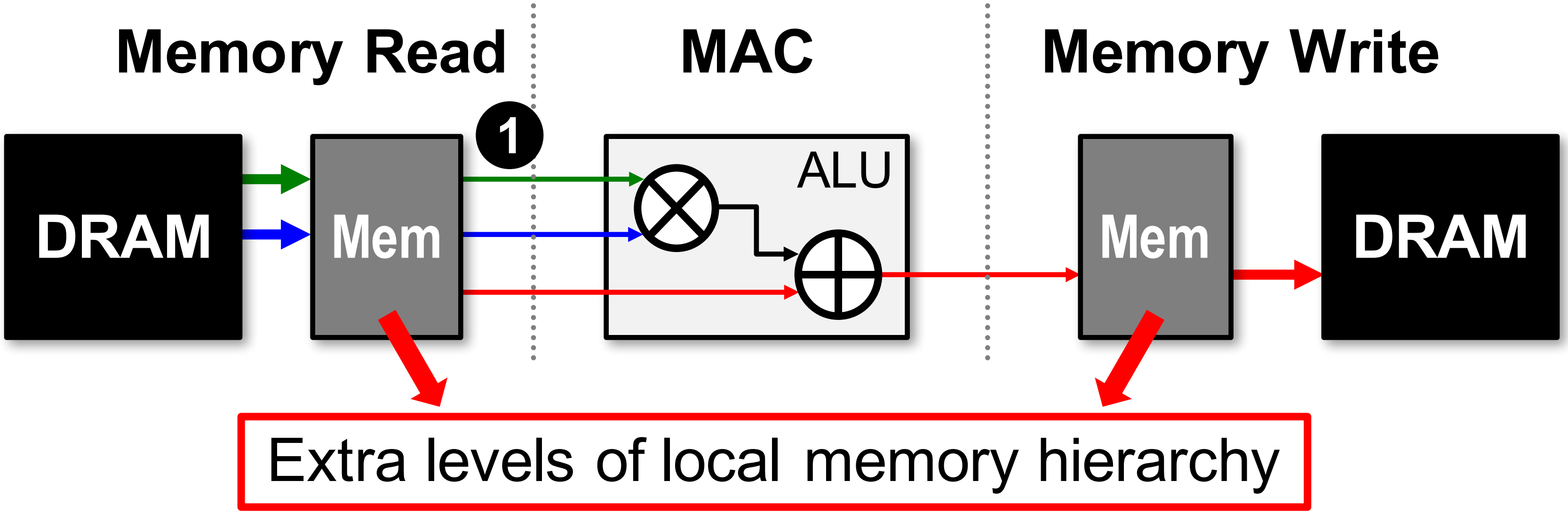
Filter Reuse

CONV and FC layers
(batch size > 1)



Reuse: Filter weights

Memory Access is the Bottleneck

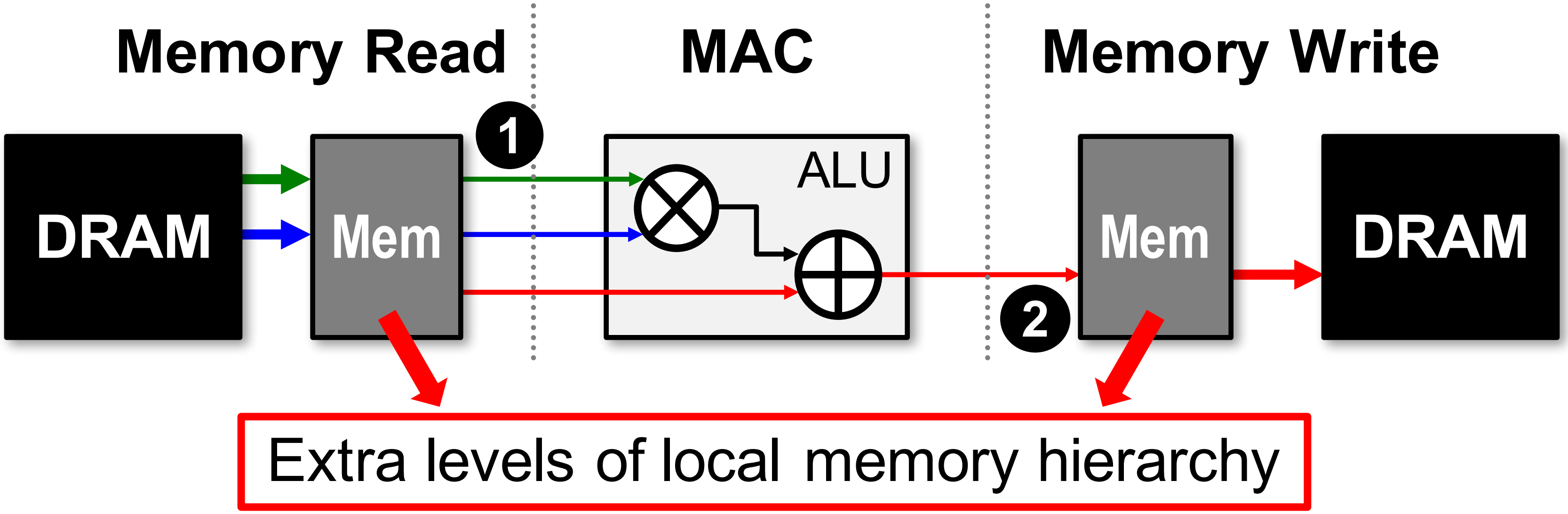


Opportunities: 1 **data reuse**

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

** AlexNet CONV layers

Memory Access is the Bottleneck

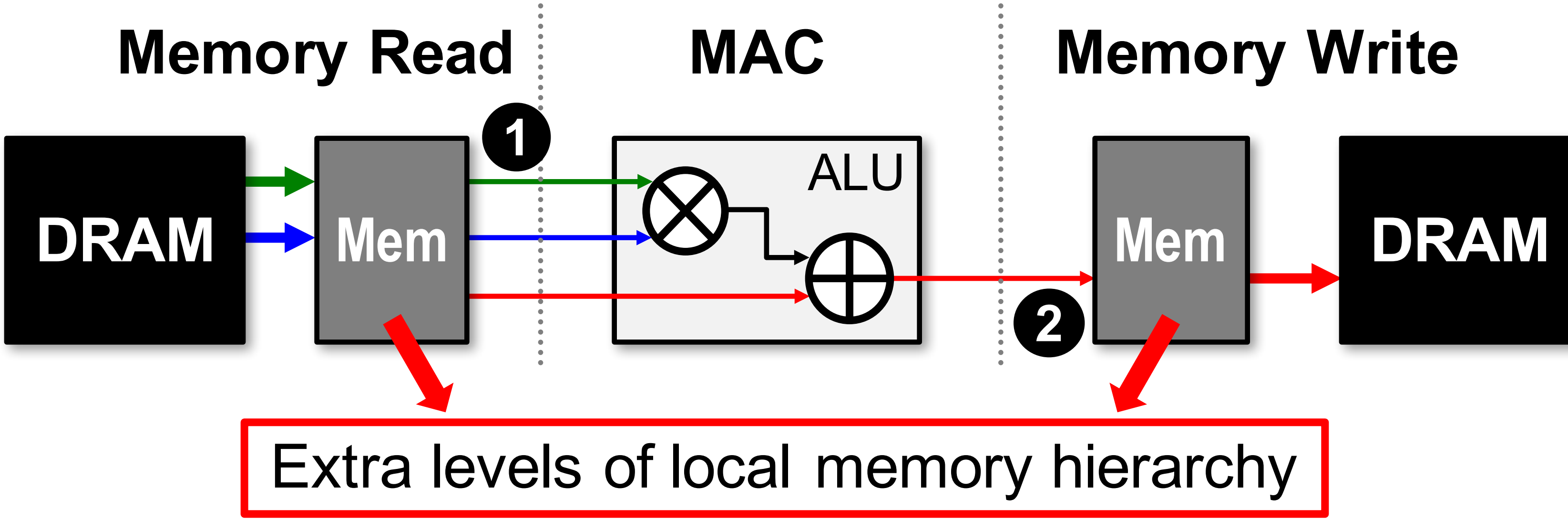


Opportunities: 1 **data reuse** 2 **local accumulation**

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

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

Memory Access is the Bottleneck

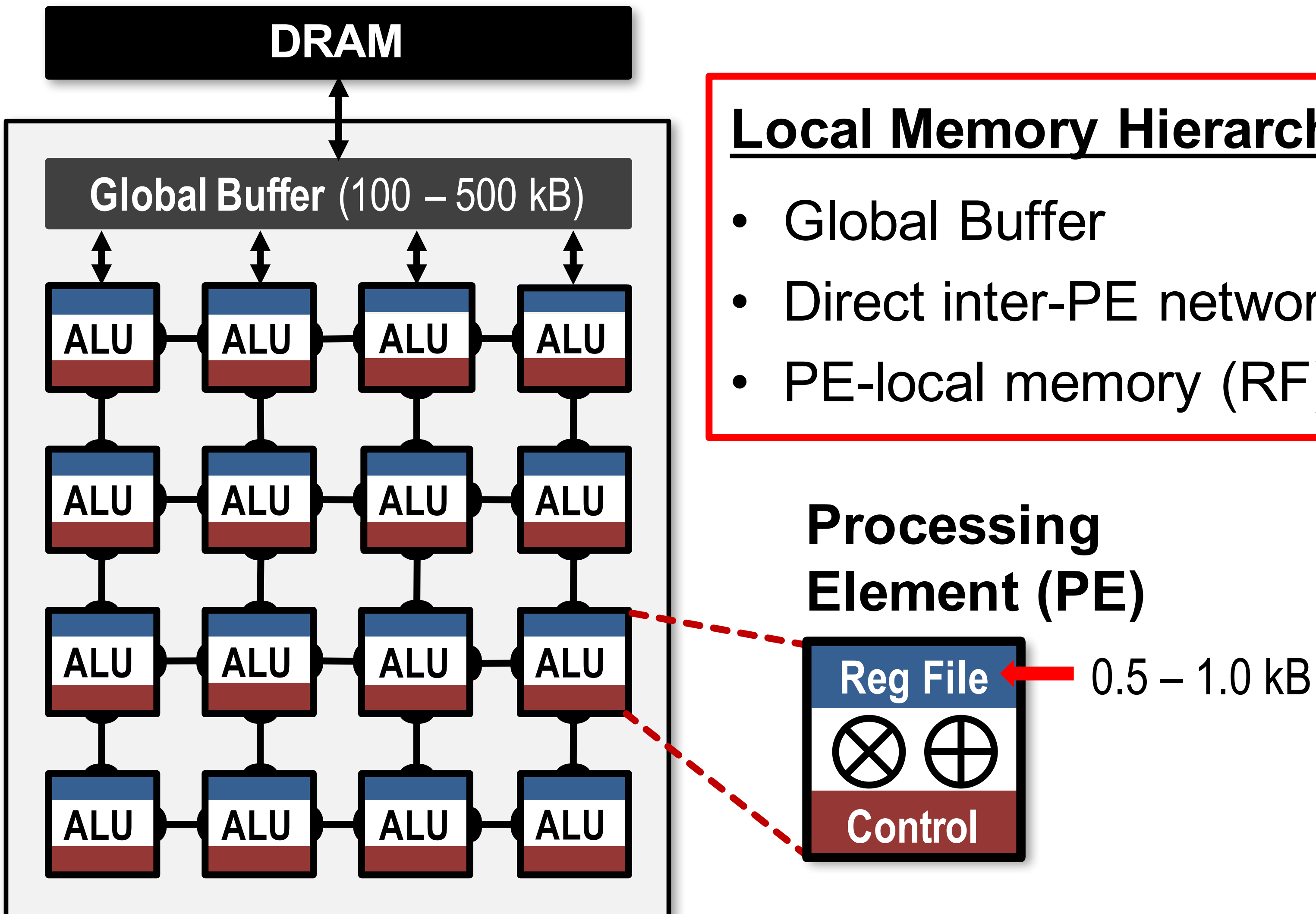


Opportunities: **1 data reuse** **2 local accumulation**

1. Can reduce DRAM reads of filter/image 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 CNN

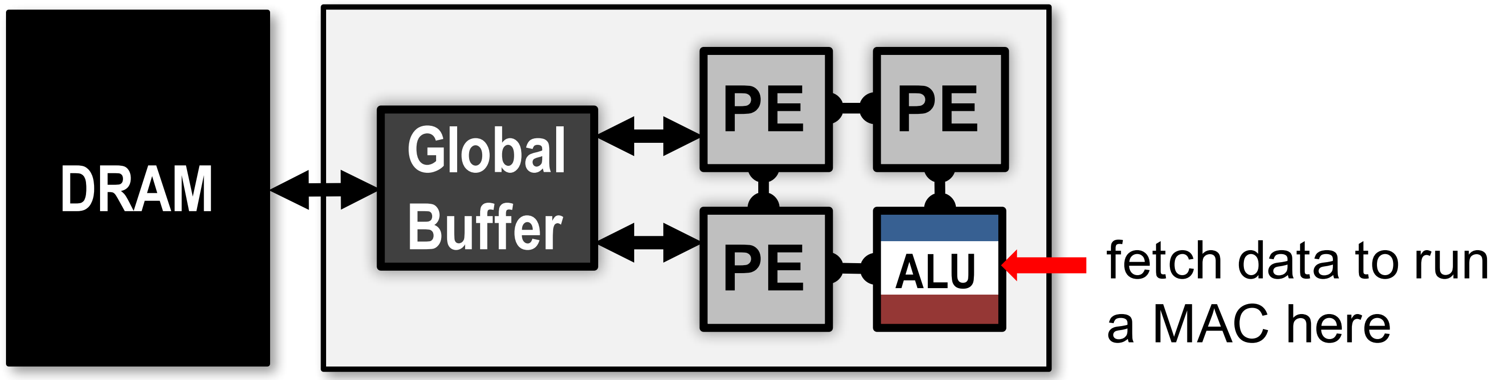


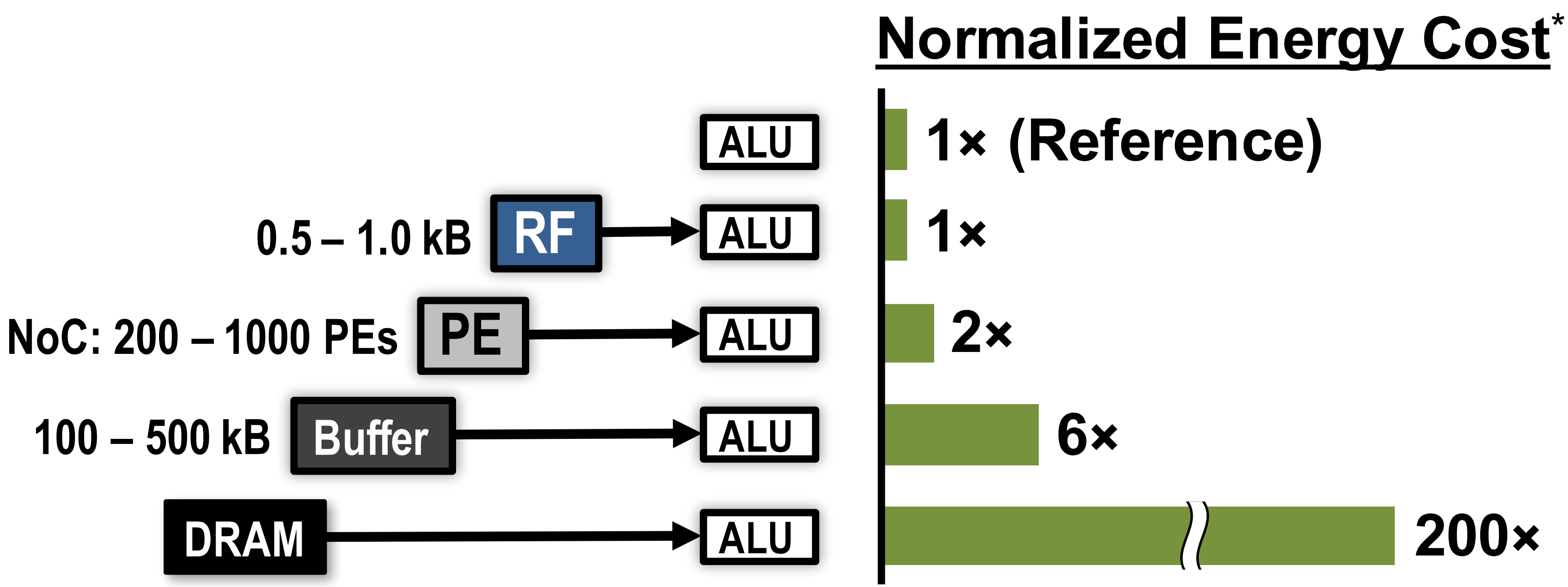
Local Memory Hierarchy

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

Processing Element (PE)

Low-Cost Local Data Access



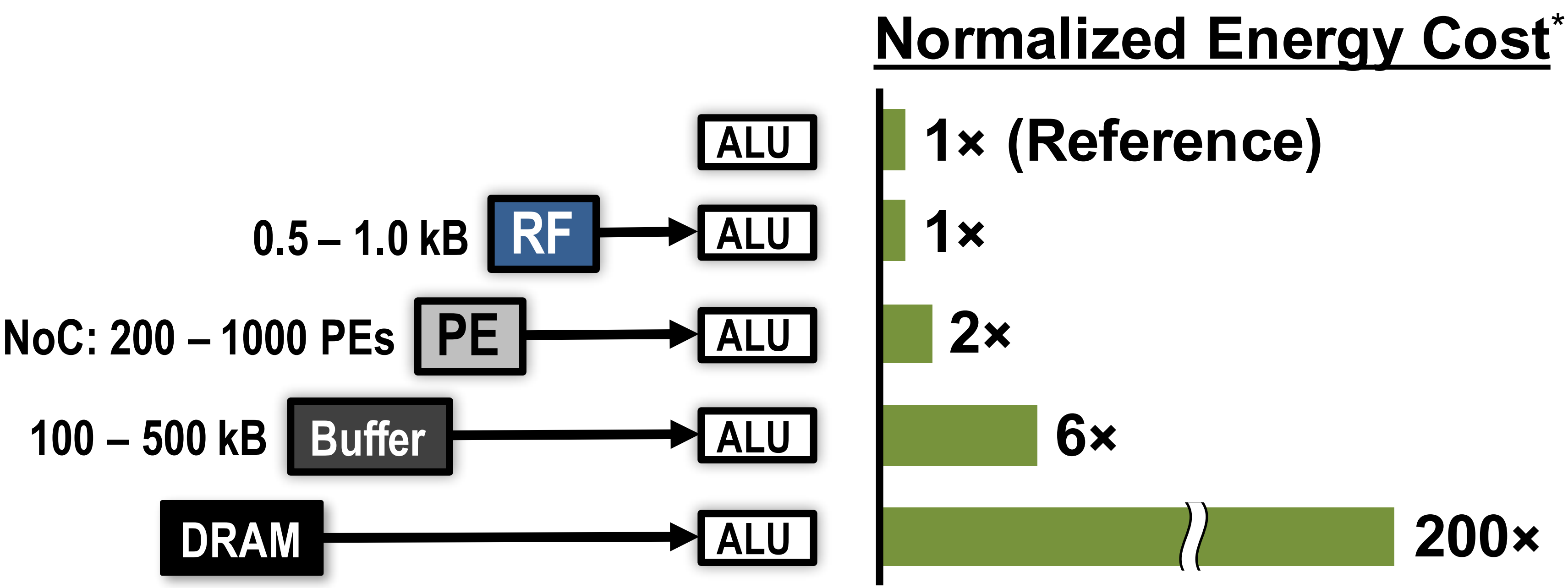


* measured from a commercial 65nm process

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!

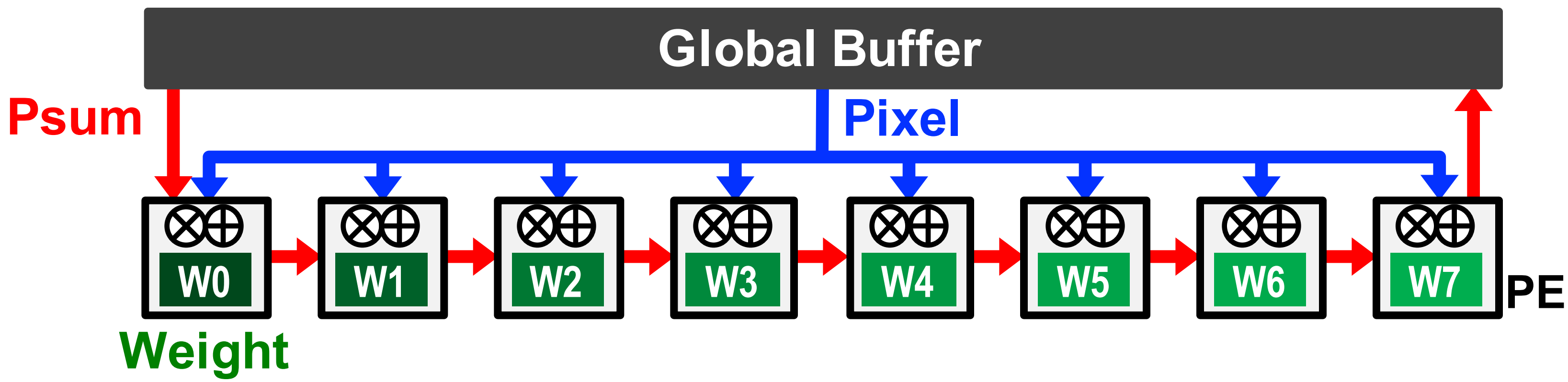


* measured from a commercial 65nm process

Taxonomy of Existing Dataflows

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

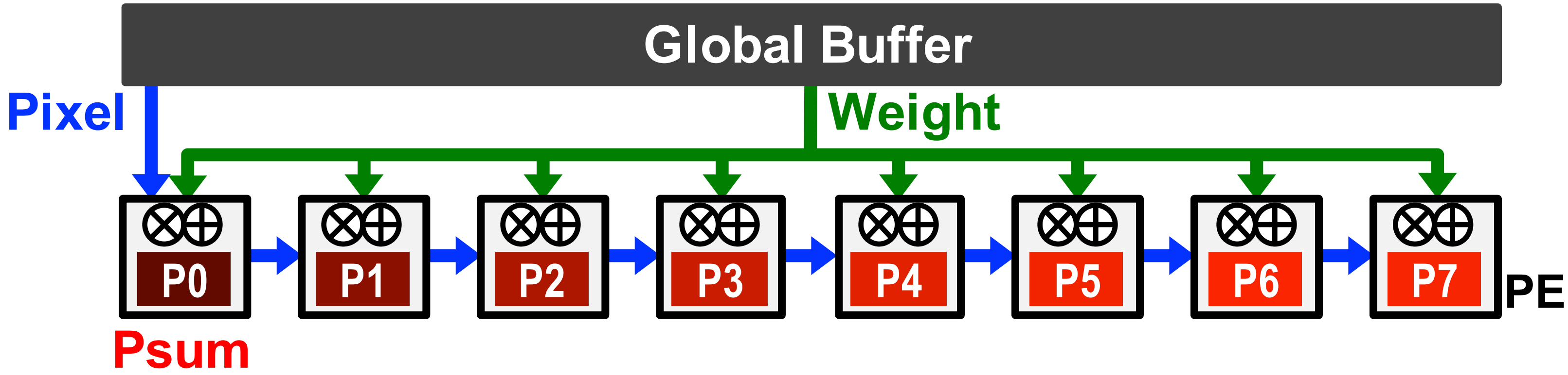
Weight Stationary (WS)



- **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]

Output Stationary (OS)

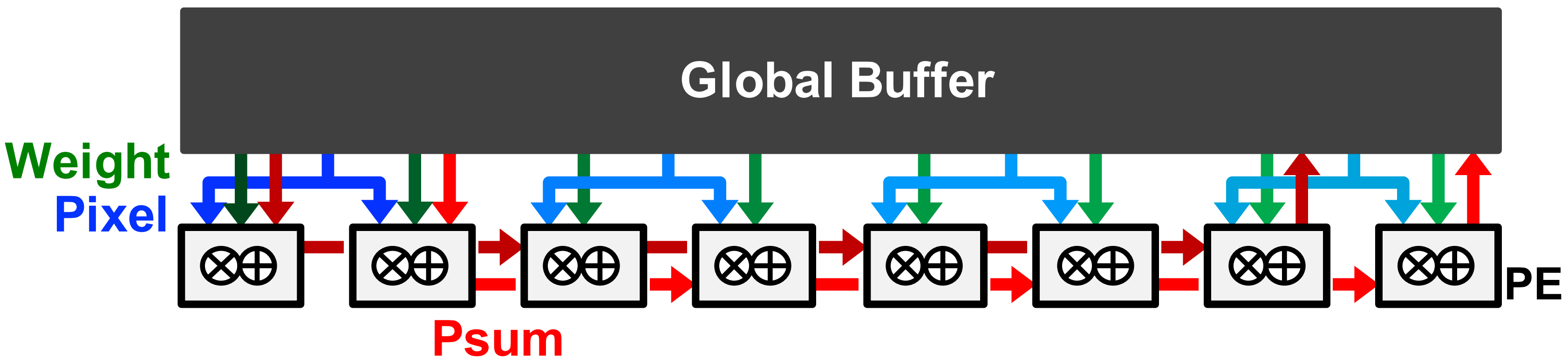


- **Minimize** **partial sum** R/W energy consumption
  - maximize local accumulation
- **Examples:**

  [**Gupta**, *ICML* 2015] [**ShiDianNao**, *ISCA* 2015]

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

No Local Reuse (NLR)

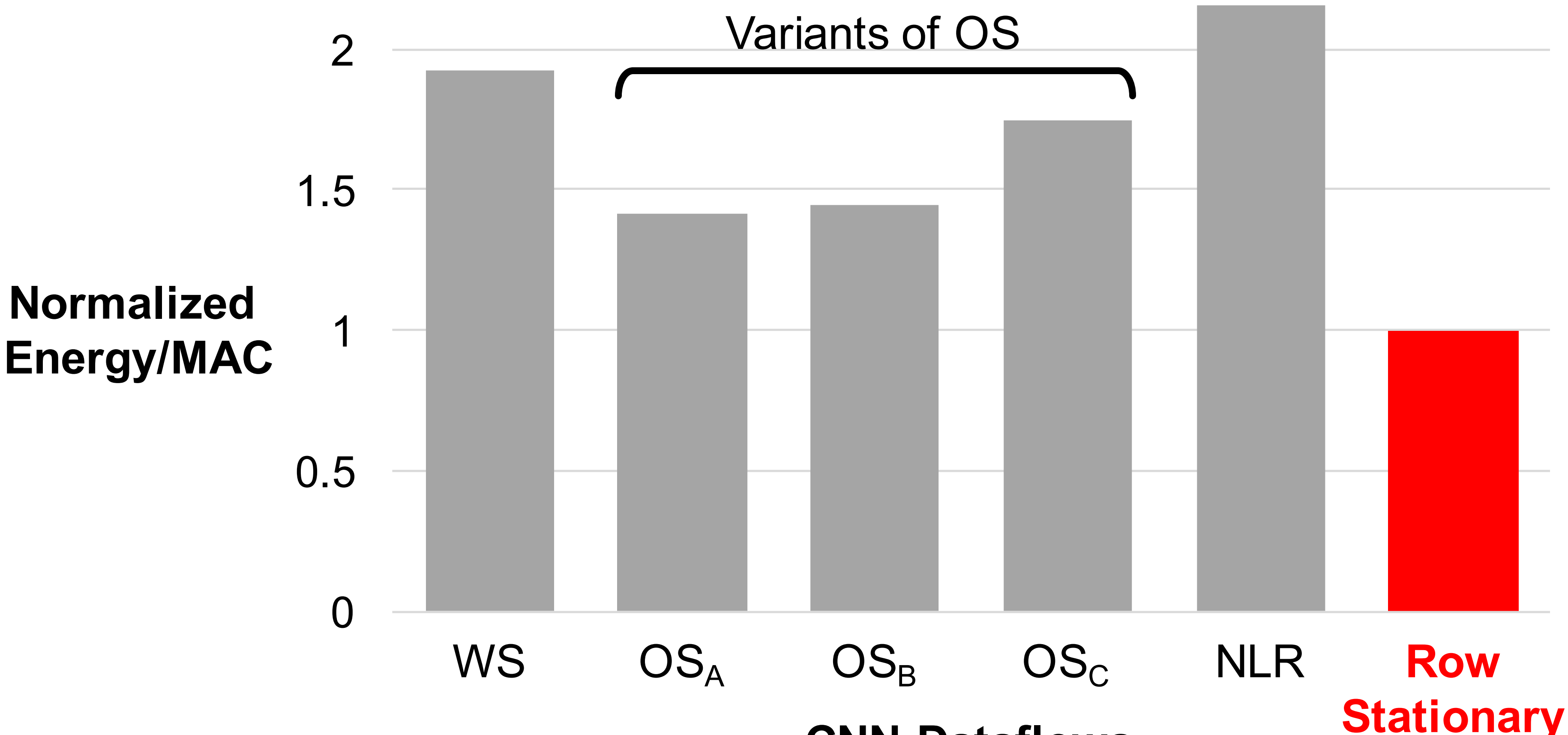


- Use a **large global buffer** as shared storage
  - Reduce **DRAM** access energy consumption
- **Examples:**

  [**DianNao**, *ASPLOS* 2014] [**DaDianNao**, *MICRO* 2014]
  [**Zhang**, *FPGA* 2015]

Energy Efficiency Comparison

- Same total area
- AlexNet Configuration*
- 256 PEs
- Batch size = 16

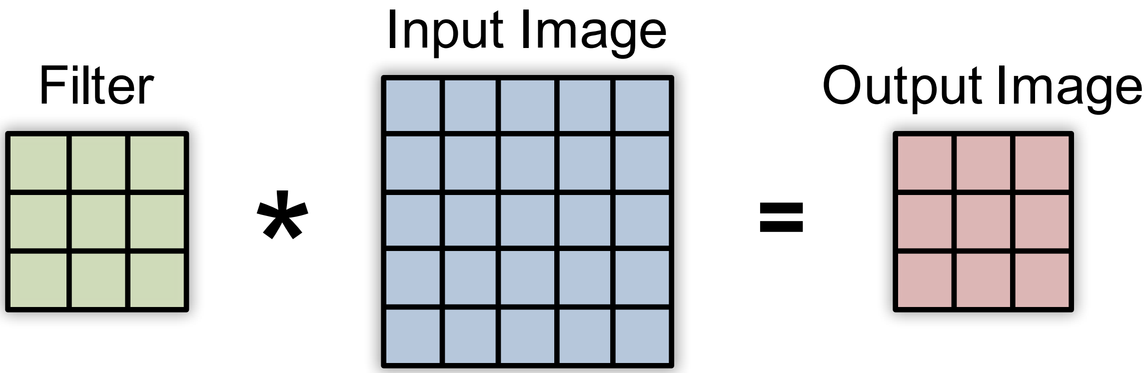


* AlexNet CONV layers

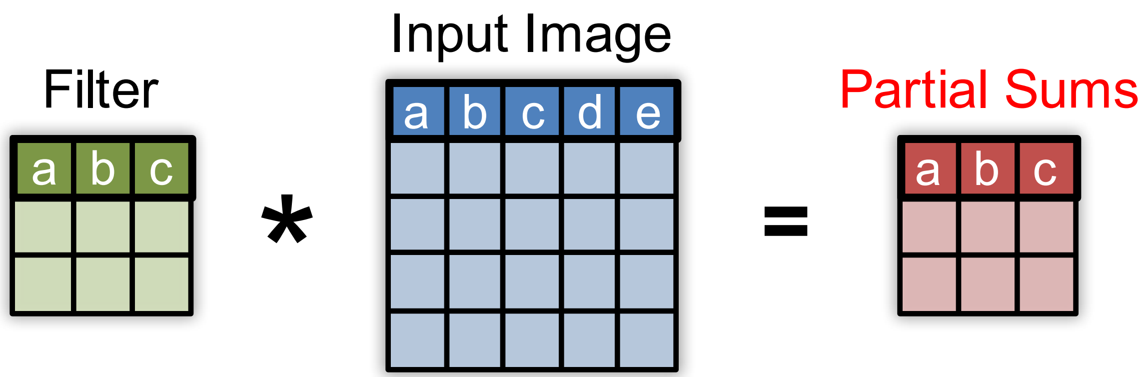
Energy-Efficient Dataflow:
Row Stationary (RS)

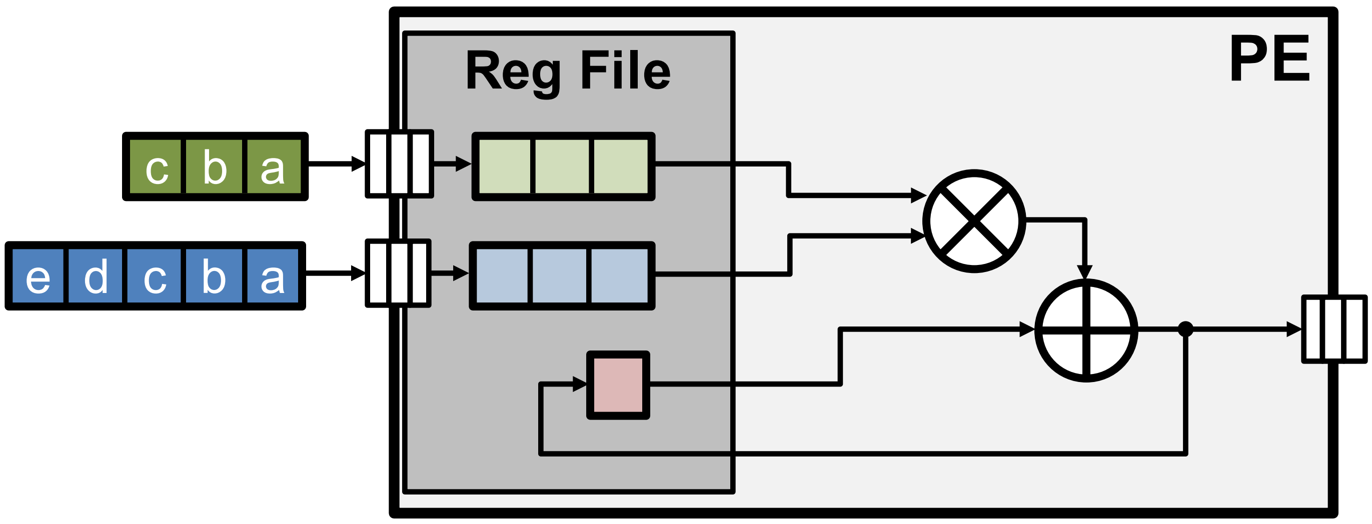
- **Maximize** reuse and accumulation at **RF**
- Optimize for **overall** energy efficiency instead for *only* a certain data type

1D Row Convolution in PE

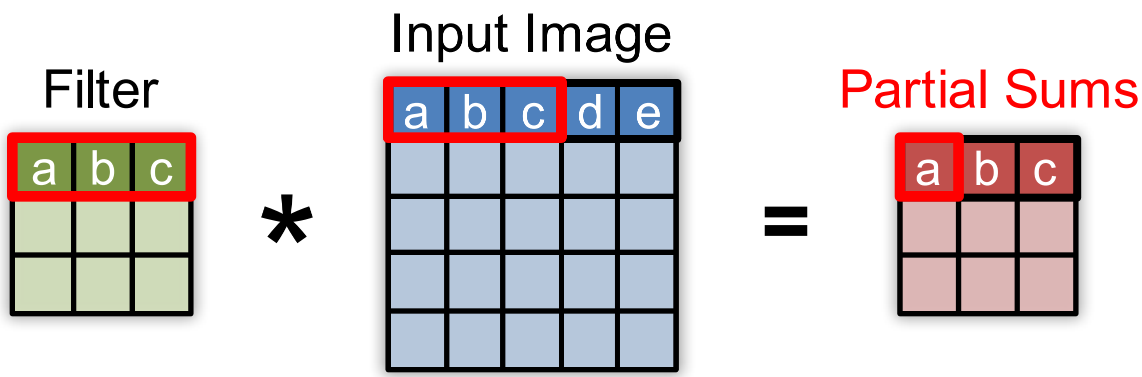


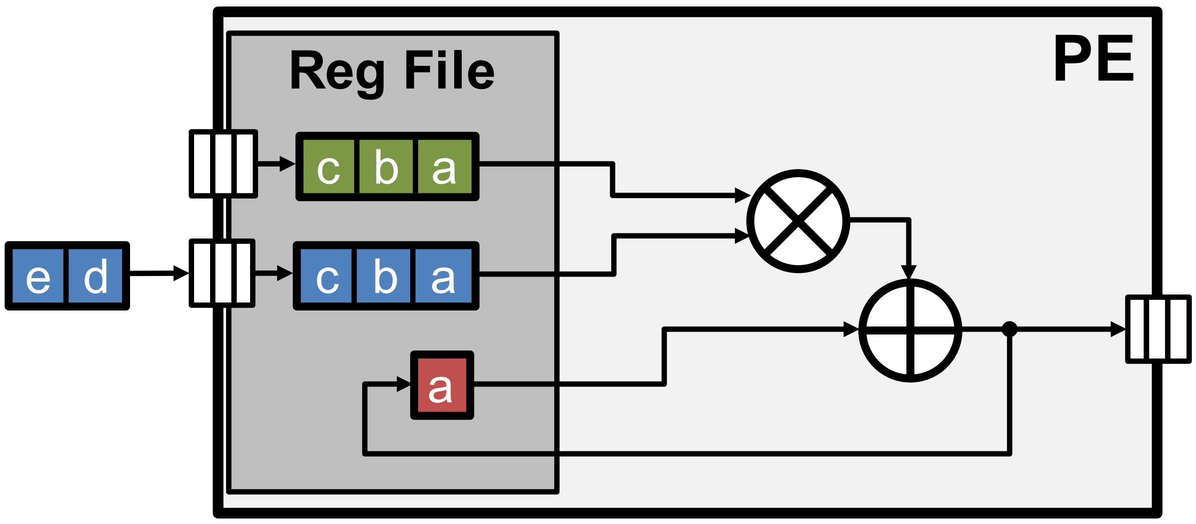
1D Row Convolution in PE



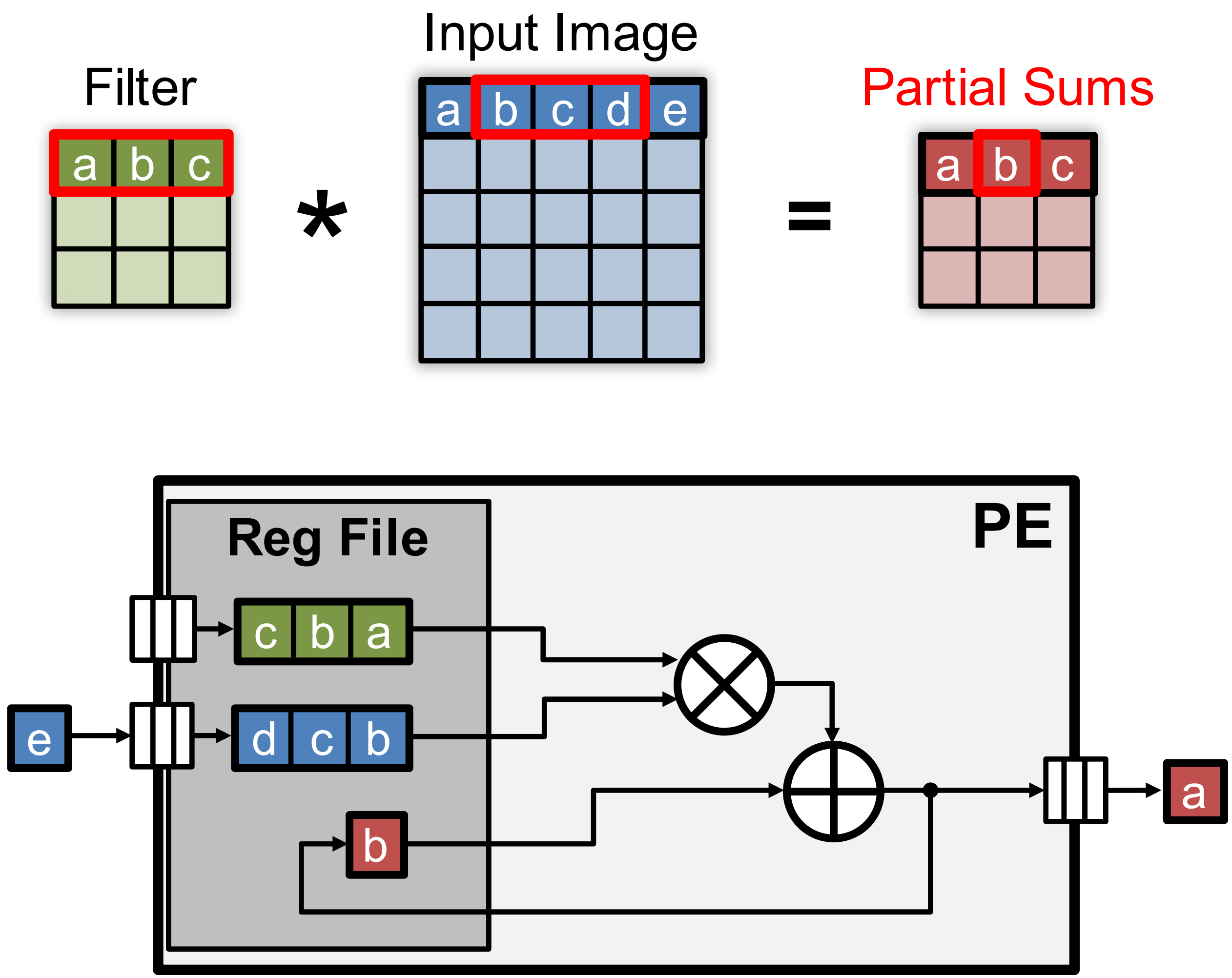


1D Row Convolution in PE

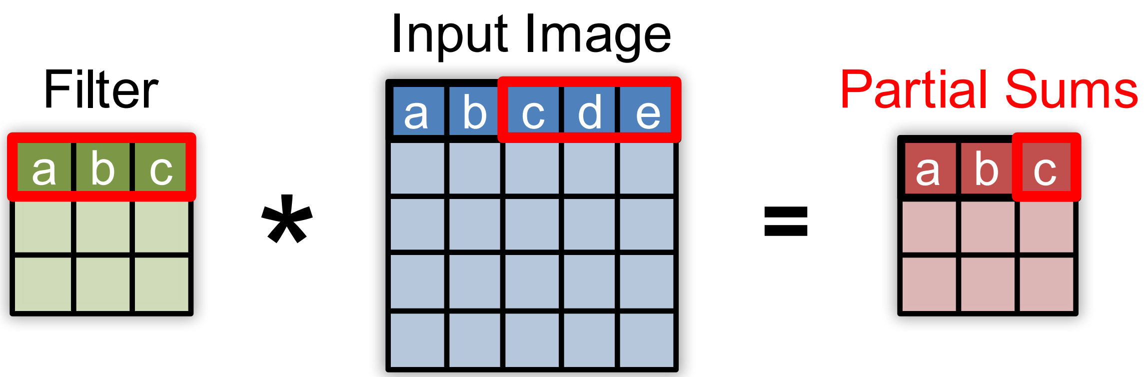


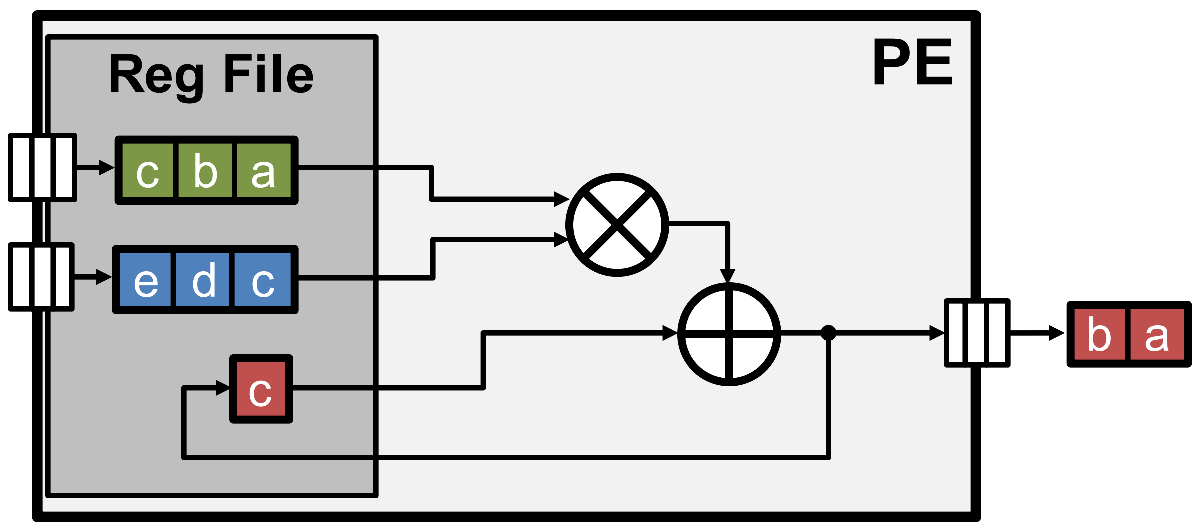


1D Row Convolution in PE



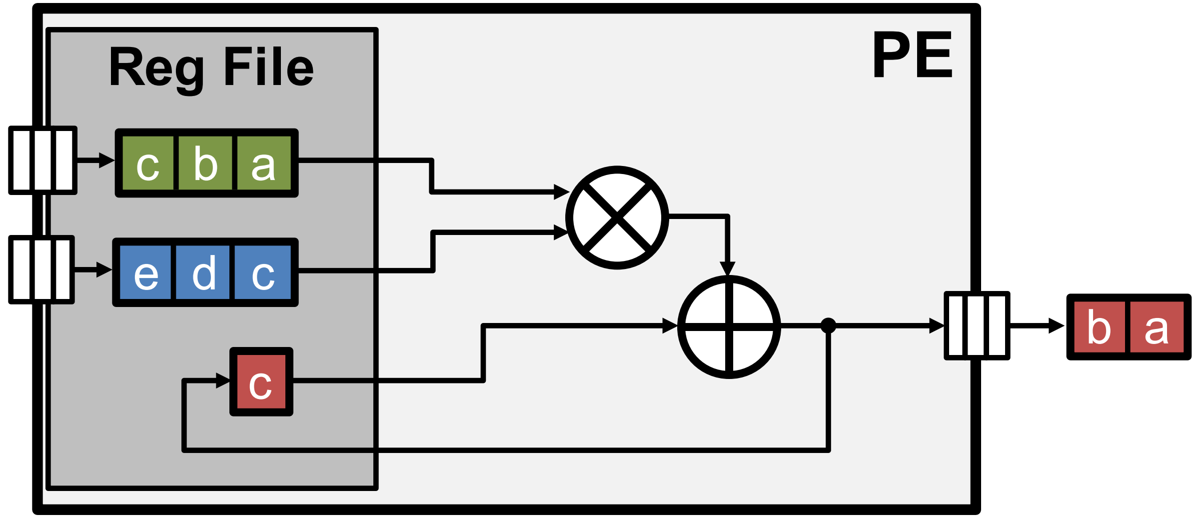
1D Row Convolution in PE



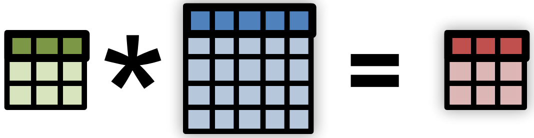


1D Row Convolution in PE

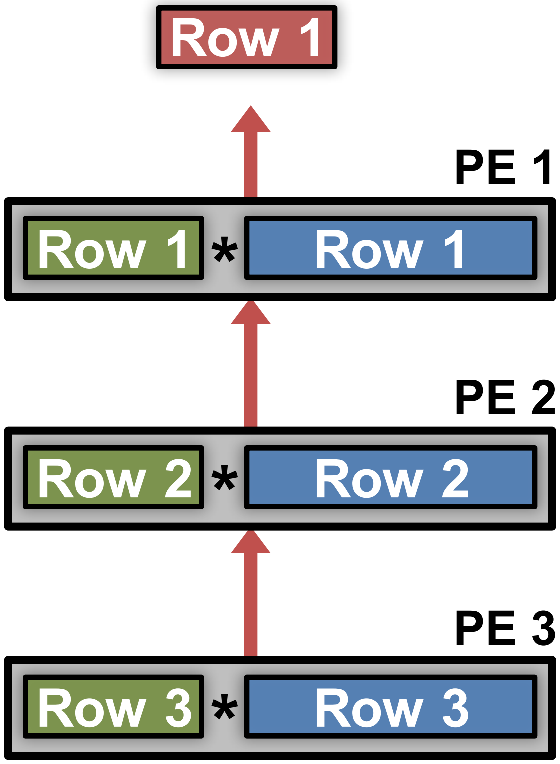
- Maximize row **convolutional reuse** in RF
  - Keep a **filter** row and **image** sliding window in RF
- Maximize row **psum accumulation** in RF

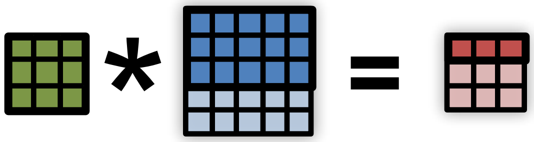


2D Convolution in PE Array

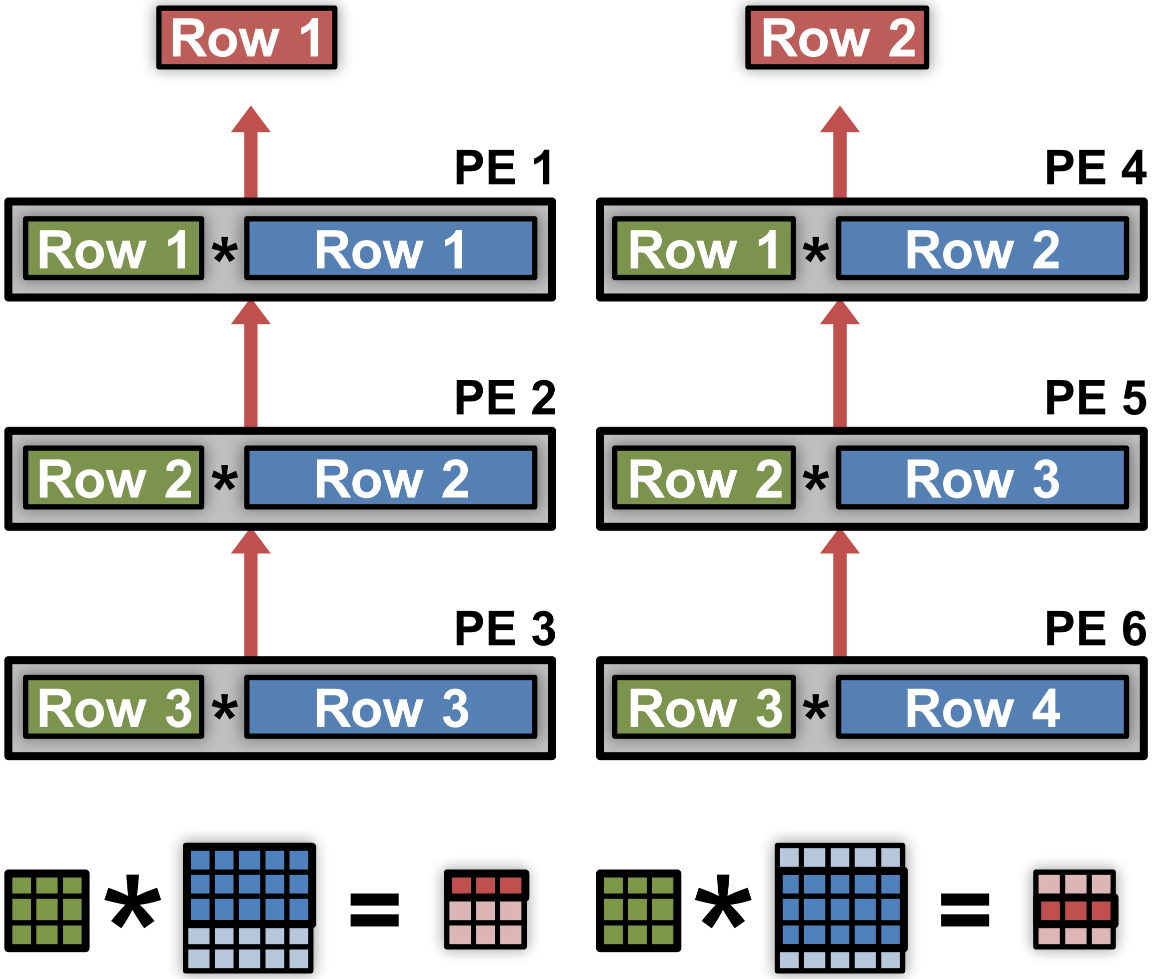


2D Convolution in PE Array

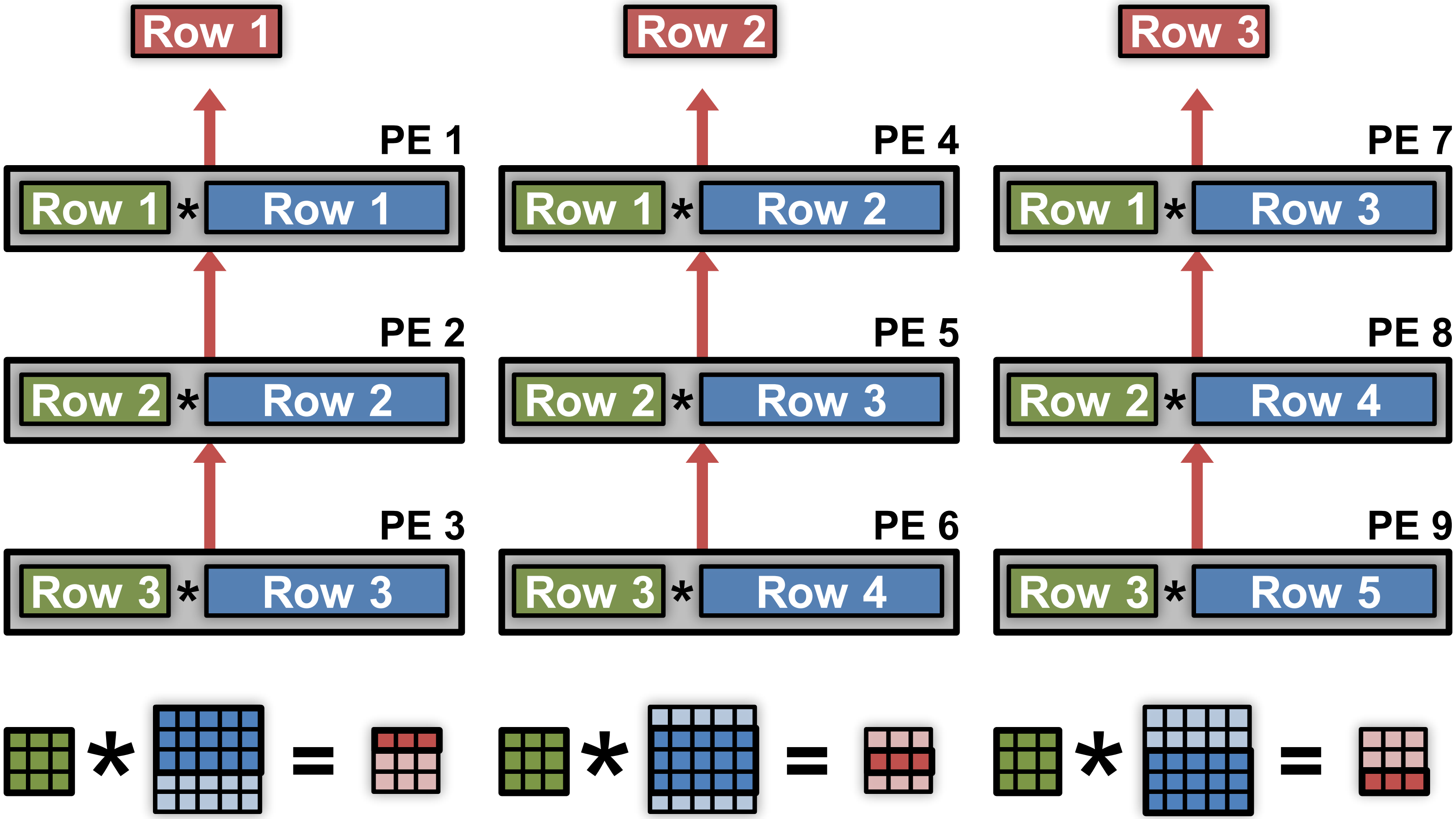




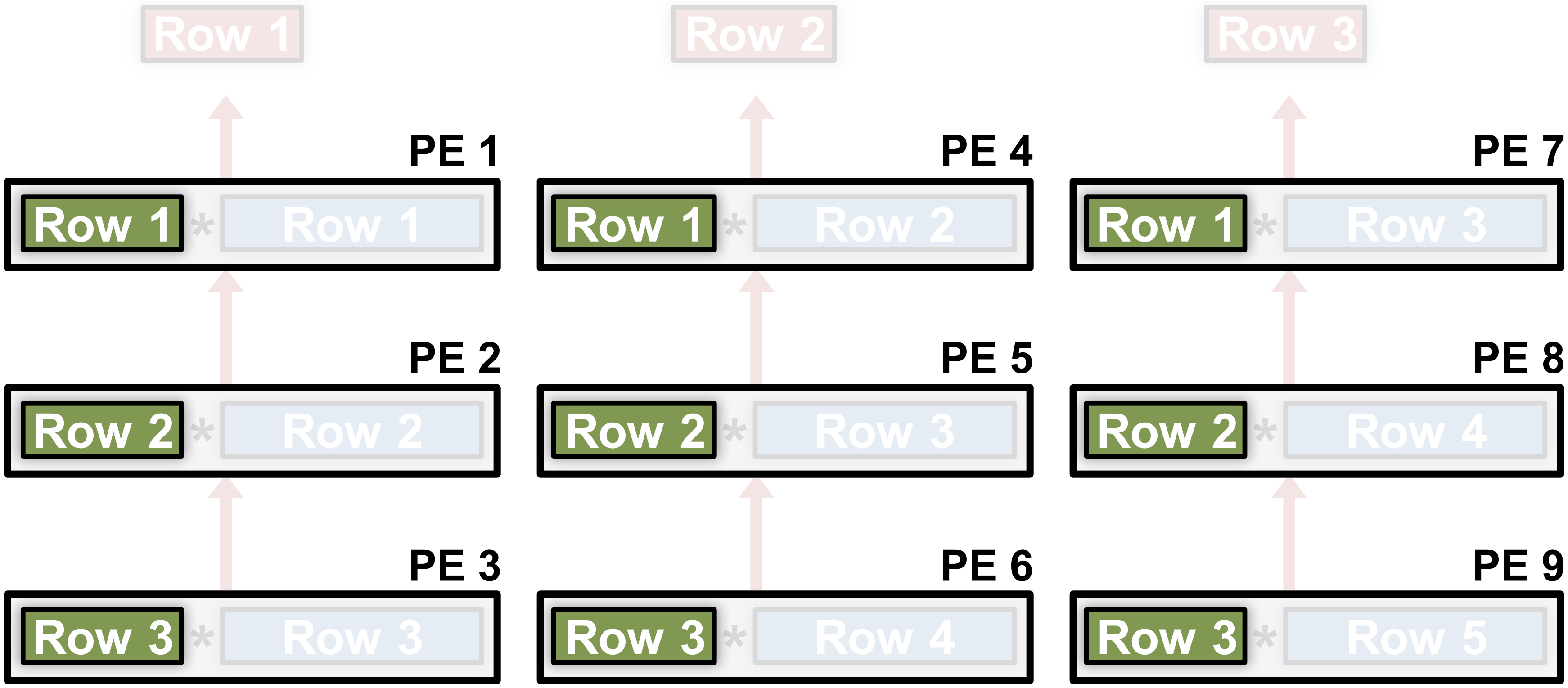
2D Convolution in PE Array



2D Convolution in PE Array



Convolutional Reuse Maximized



Filter rows are reused across PEs **horizontally**

Convolutional Reuse Maximized

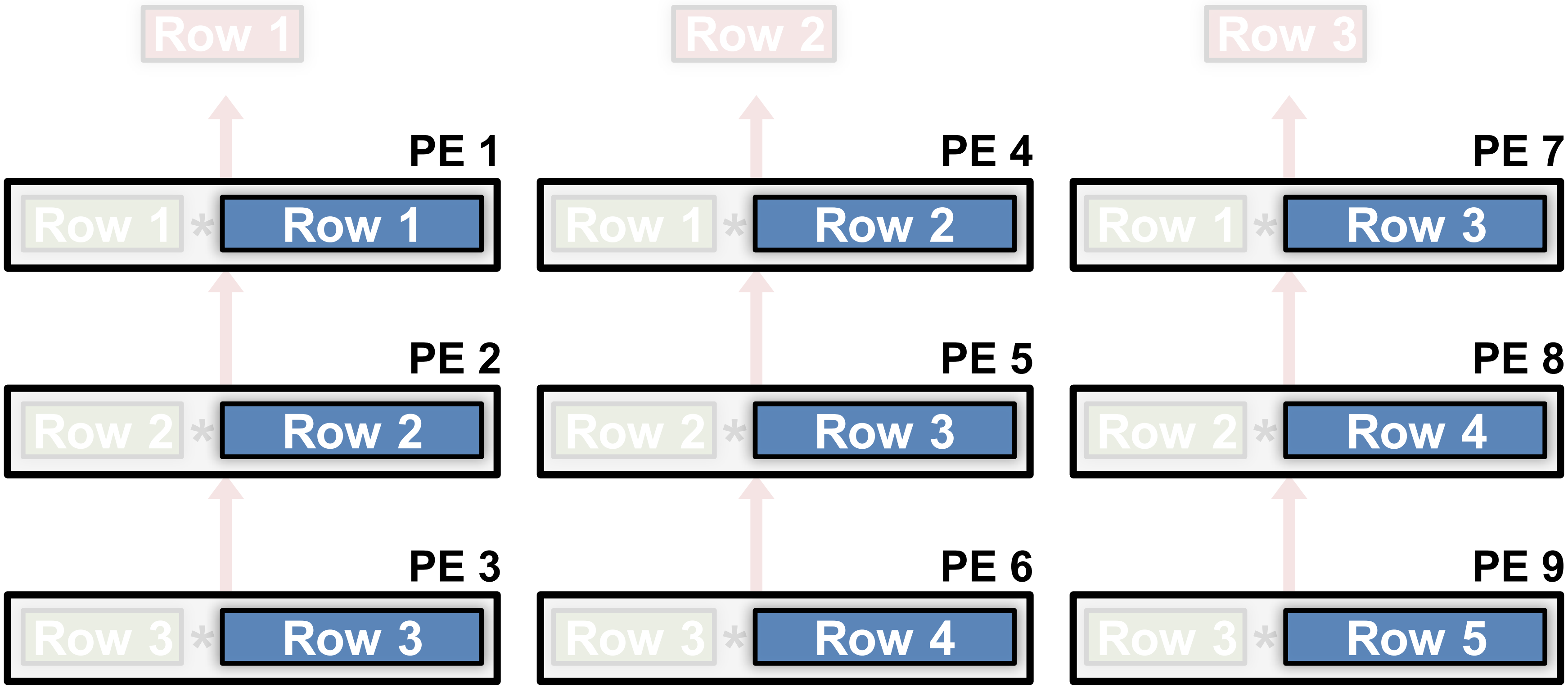
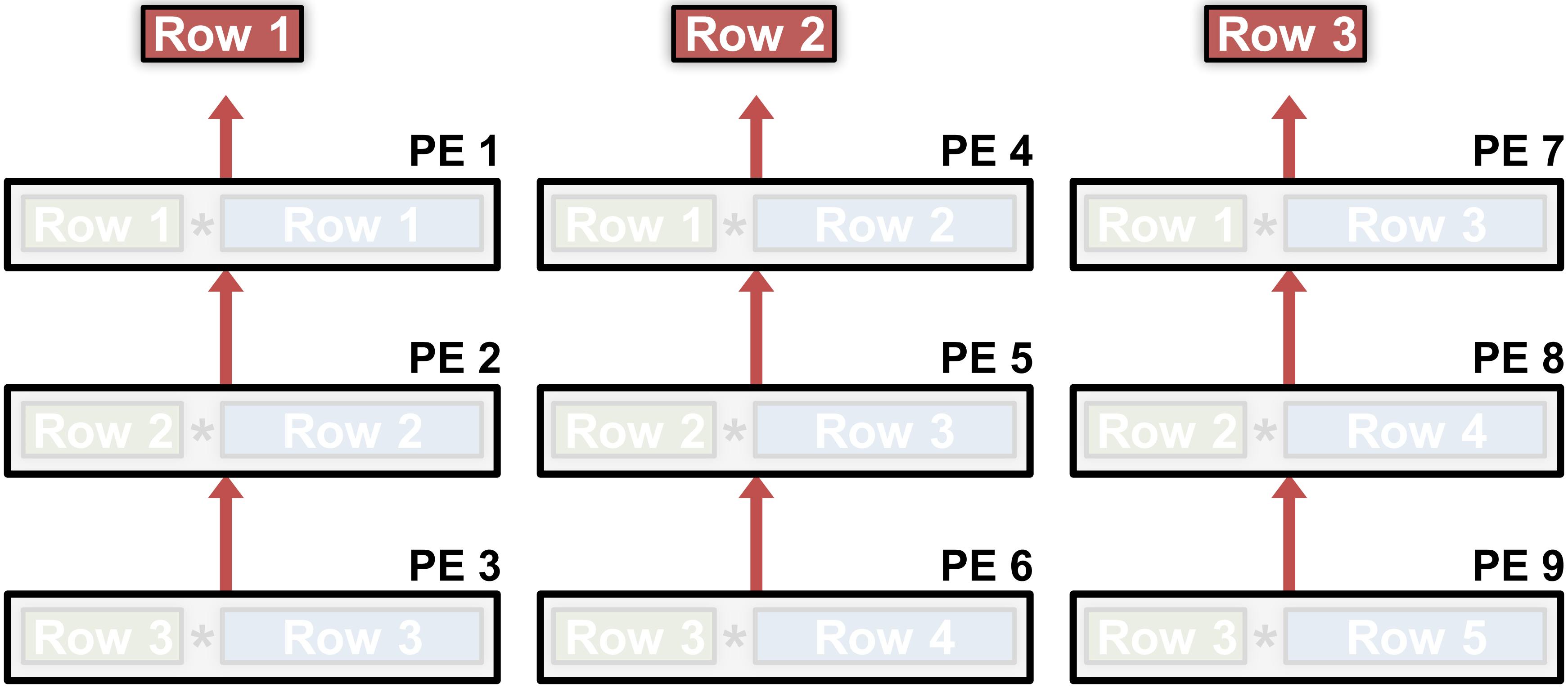


Image rows are reused across PEs **diagonally**

Maximize 2D Accumulation in PE Array



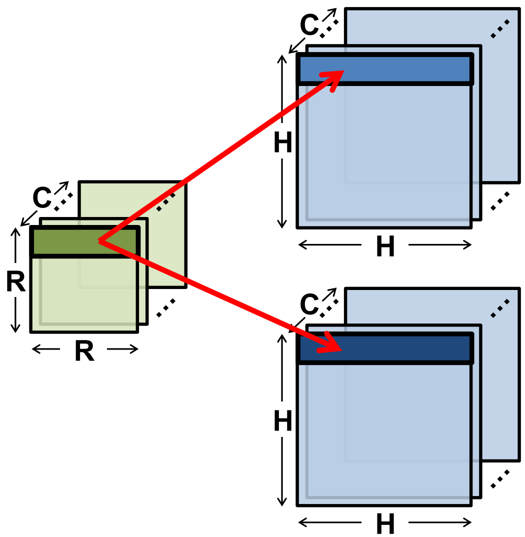
Partial sums accumulate across PEs **vertically**

Dimensions Beyond 2D Convolution

1 Multiple Images 2 Multiple Filters 3 Multiple Channels

Filter Reuse in PE

1 Multiple Images 2 Multiple Filters 3 Multiple Channels



	Filter 1		Image 1		Psum 1
Channel 1	Row 1	*	Row 1	=	Row 1
	Filter 1		**Image 2**		**Psum 2**
Channel 1	Row 1	*	Row 1	=	Row 1

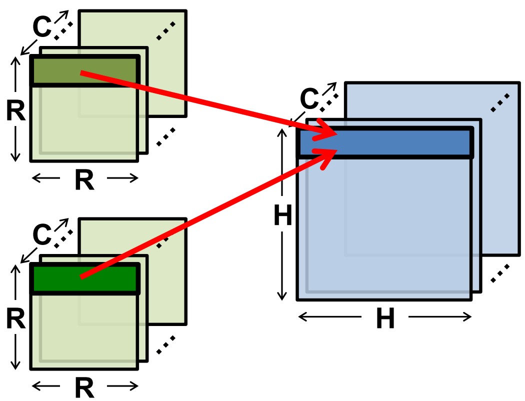
share the same filter row

Processing in PE: concatenate image rows

	Filter 1		Image 1 & 2			Psum 1 & 2	
Channel 1	Row 1	*	Row 1	Row 1	=	Row 1	Row 1

Image Reuse in PE

1 Multiple Images **2 Multiple Filters** 3 Multiple Channels



	Filter 1		Image 1		Psum 1
Channel 1	Row 1	*	Row 1	=	Row 1
	Filter 2		**Image 1**		**Psum 2**
Channel 1	Row 1	*	Row 1	=	Row 1

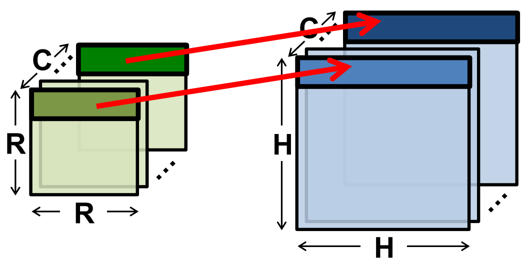
share the same image row

<u>Processing in PE</u>: **interleave filter rows**

	Filter 1 & 2		Image 1		Psum 1 & 2
Channel 1		*	Row 1	=	

Channel Accumulation in PE

1 Multiple Images 2 Multiple Filters **3 Multiple Channels**



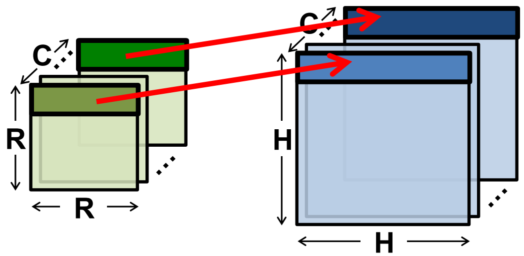
	Filter 1		Image 1		Psum 1
Channel 1	Row 1	*	Row 1	=	Row 1
	Filter 1		**Image 1**		**Psum 1**
Channel 2	Row 1	*	Row 1	=	Row 1

accumulate psums

Row 1 + Row 1 = Row 1

Channel Accumulation in PE

1 Multiple Images 2 Multiple Filters **3 Multiple Channels**



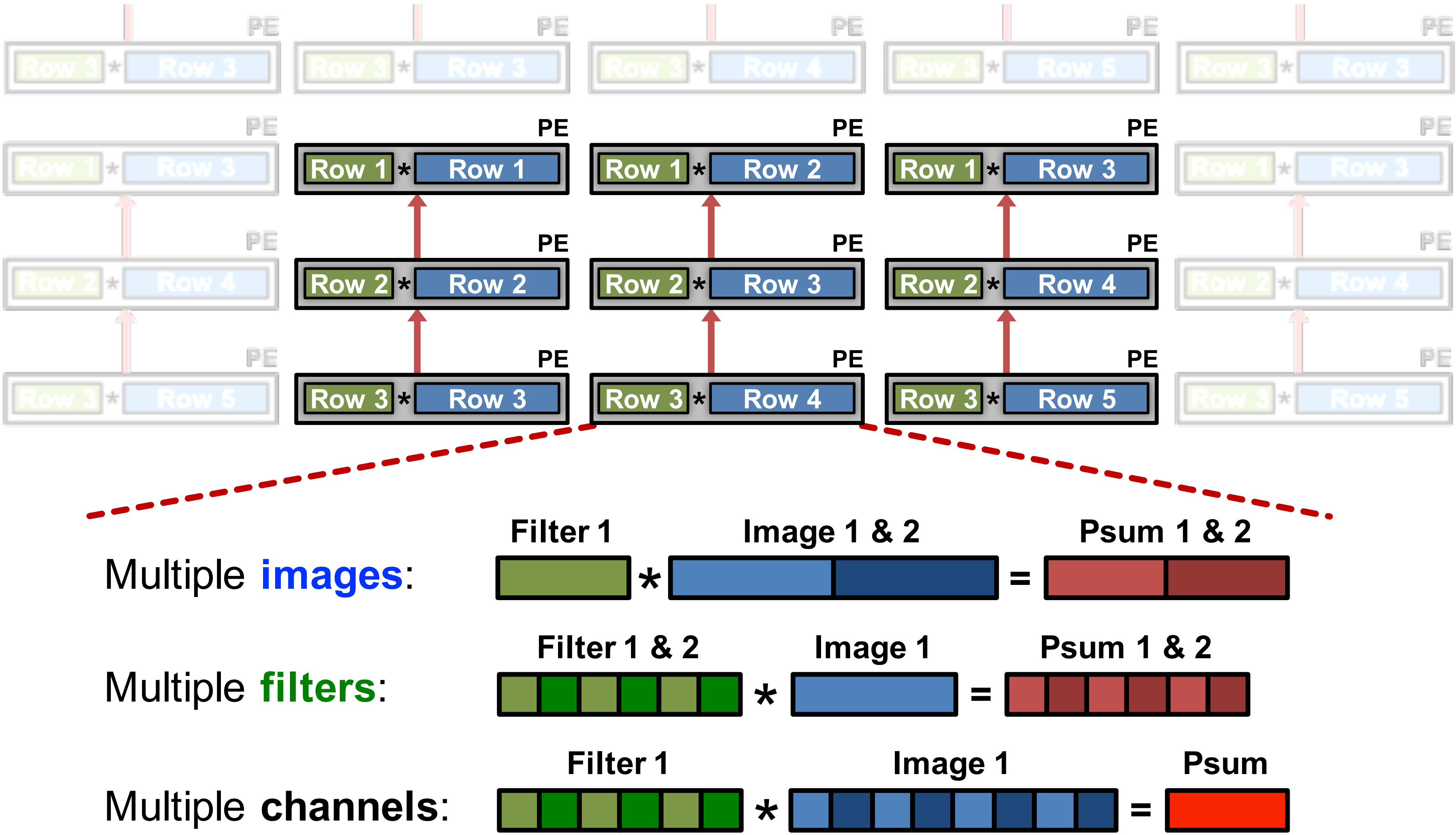
	Filter 1		Image 1		Psum 1
Channel 1	Row 1	*	Row 1	=	Row 1
	Filter 1		**Image 1**		**Psum 1**
Channel 2	Row 1	*	Row 1	=	Row 1

accumulate psums

Processing in PE: **interleave channels**

	Filter 1		Image 1		Psum
Channel 1 & 2		*		=	Row 1

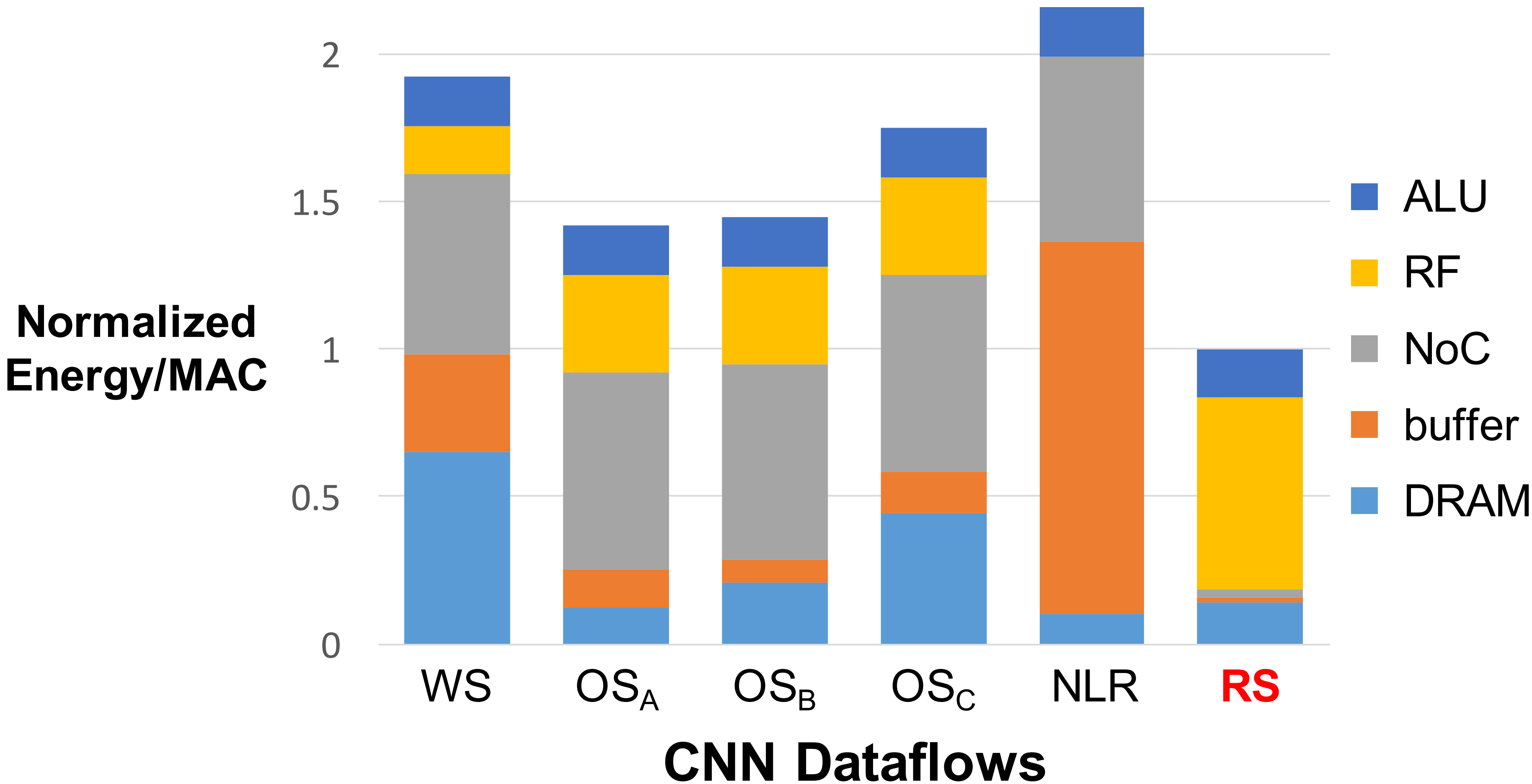
CNN Convolution – The Full Picture



Simulation Results

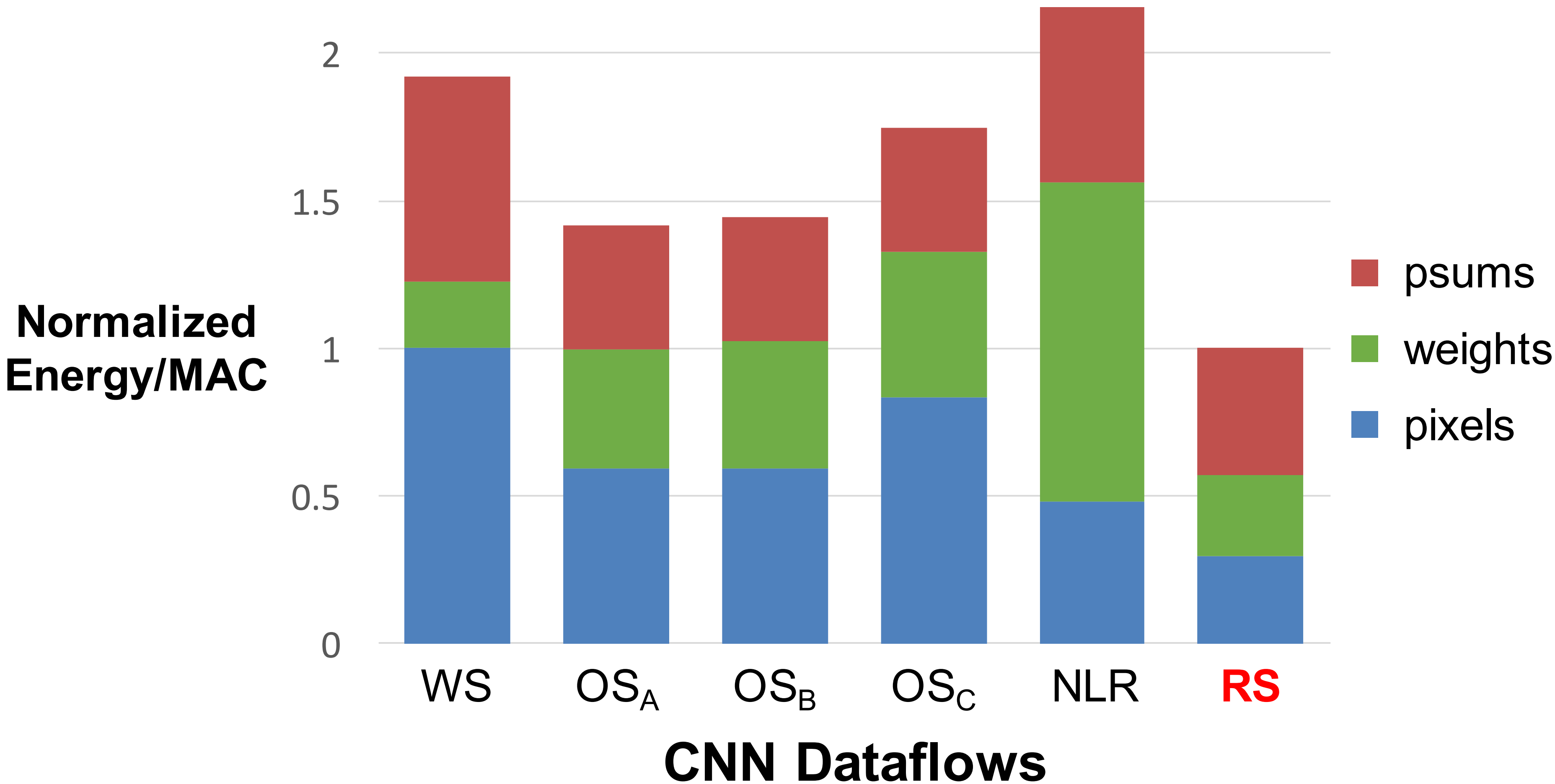
- Same total hardware area
- 256 PEs
- AlexNet Configuration
- Batch size = 16

Dataflow Comparison: CONV Layers



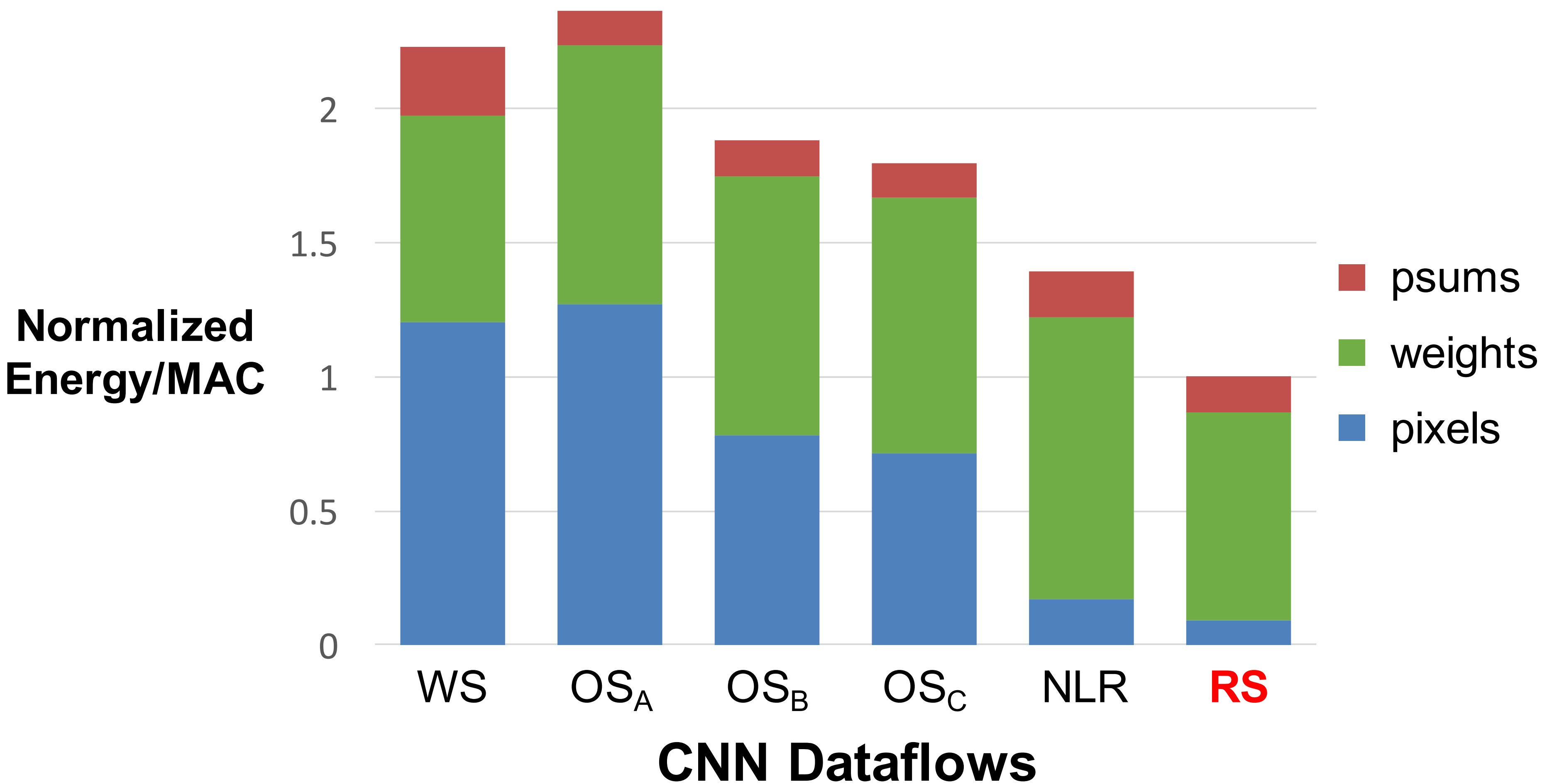
RS uses **1.4× – 2.5× lower** energy than other dataflows

Dataflow Comparison: CONV Layers



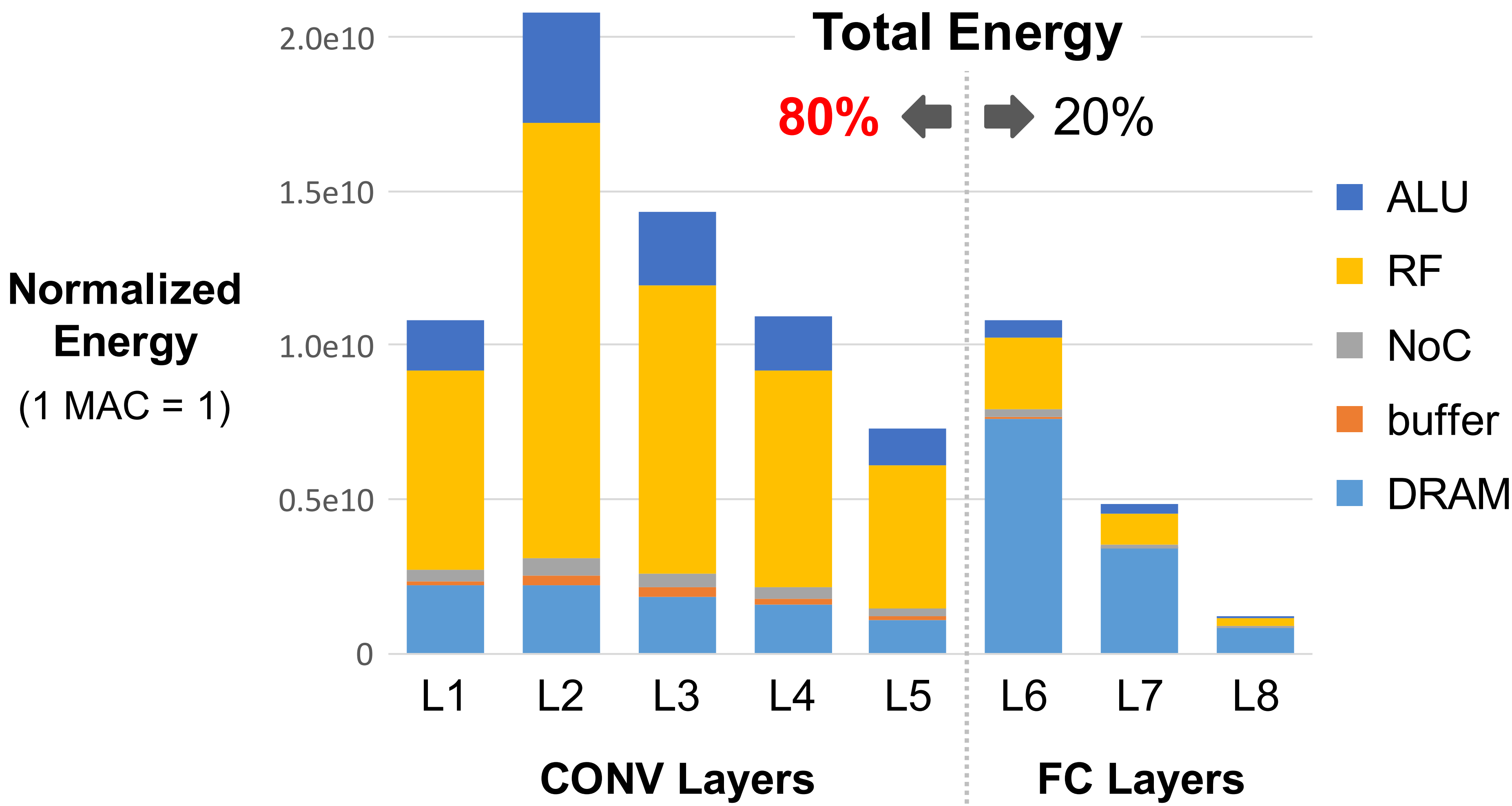
RS optimizes for the best **overall** energy efficiency

Dataflow Comparison: FC Layers



RS uses at least **1.3× lower** energy than other dataflows

Row Stationary: Layer Breakdown



RF dominates

DRAM dominates

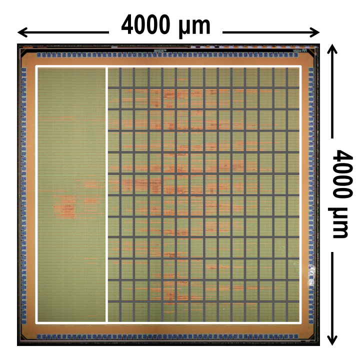
Summary

- We propose a **Row Stationary** (RS) dataflow to exploit the **low-cost local memories** in a spatial architecture.

- RS optimizes for best **overall** energy efficiency while existing CNN dataflows only focus on certain data types.

- RS has higher energy efficiency than existing dataflows
  - **1.4× – 2.5×** higher in **CONV** layers
  - at least **1.3×** higher in **FC** layers. (batch size ≥ 16)

- We have verified RS in a fabricated CNN processor chip, ***Eyeriss***



Thank You

Learn more about **Eyeriss** at

http://eyeriss.mit.edu