DNN Accelerator Architectures

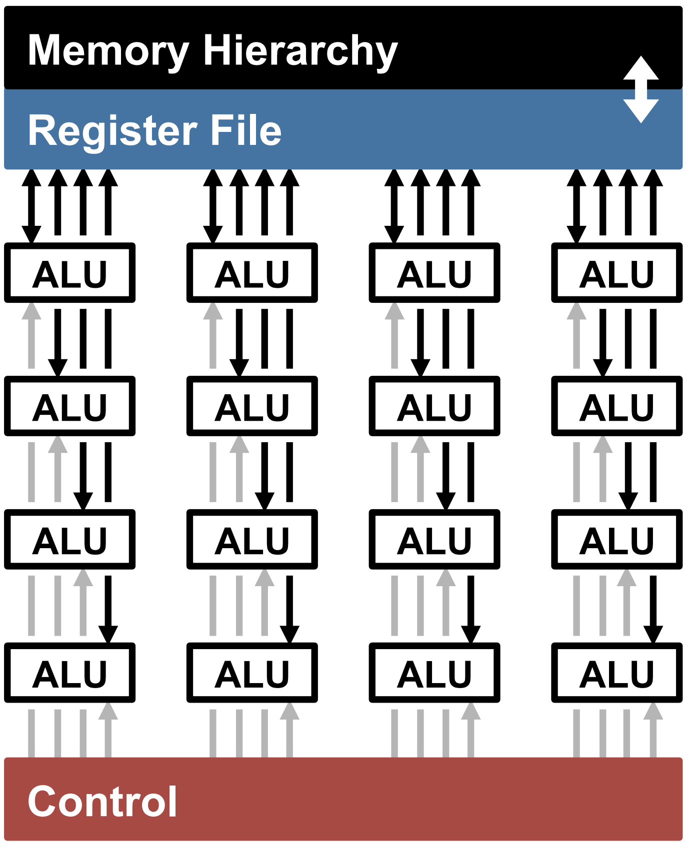
ISCA Tutorial (2017)

Website: http://eyeriss.mit.edu/tutorial.html

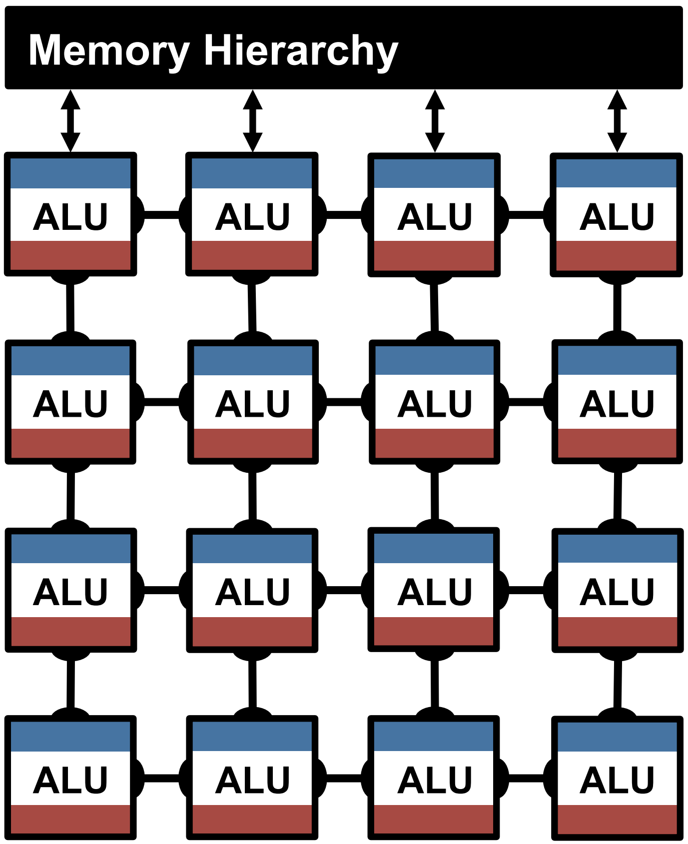
Joel Emer, Vivienne Sze, Yu-Hsin Chen

Highly-Parallel Compute Paradigms

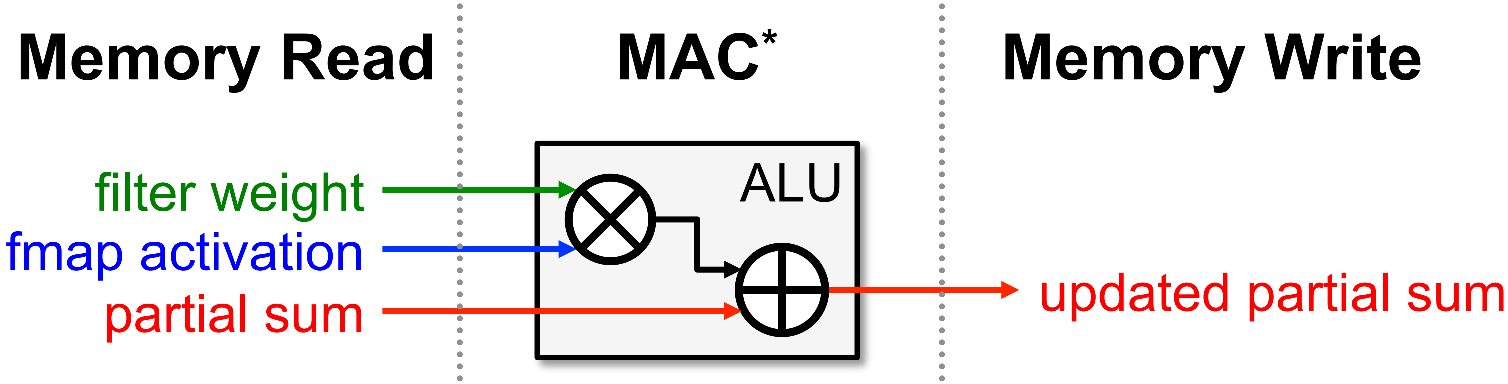
Temporal Architecture (SIMD/SIMT)



Spatial Architecture (Dataflow Processing)

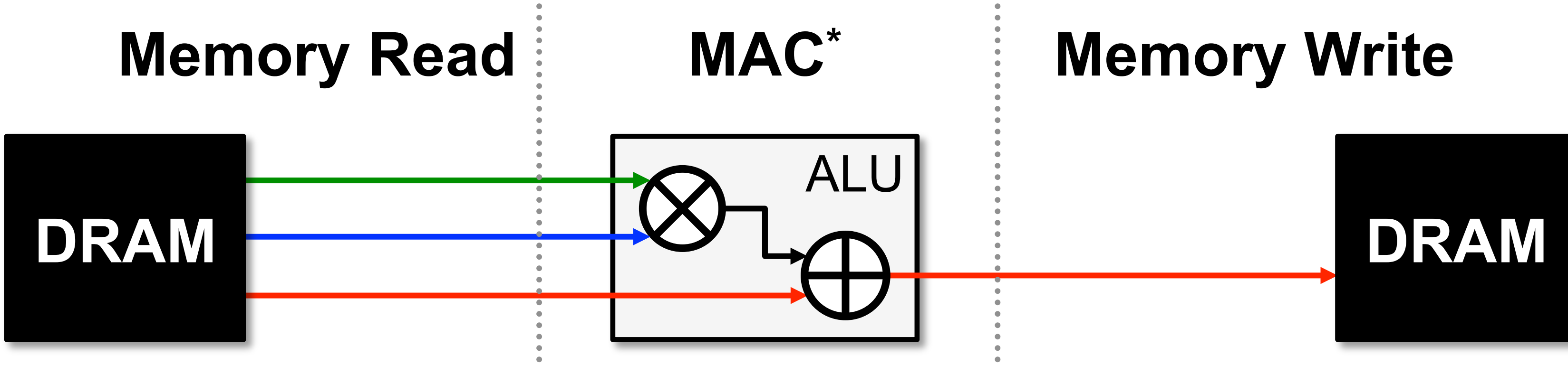


Memory Access is the Bottleneck



* multiply-and-accumulate

Memory Access is the Bottleneck

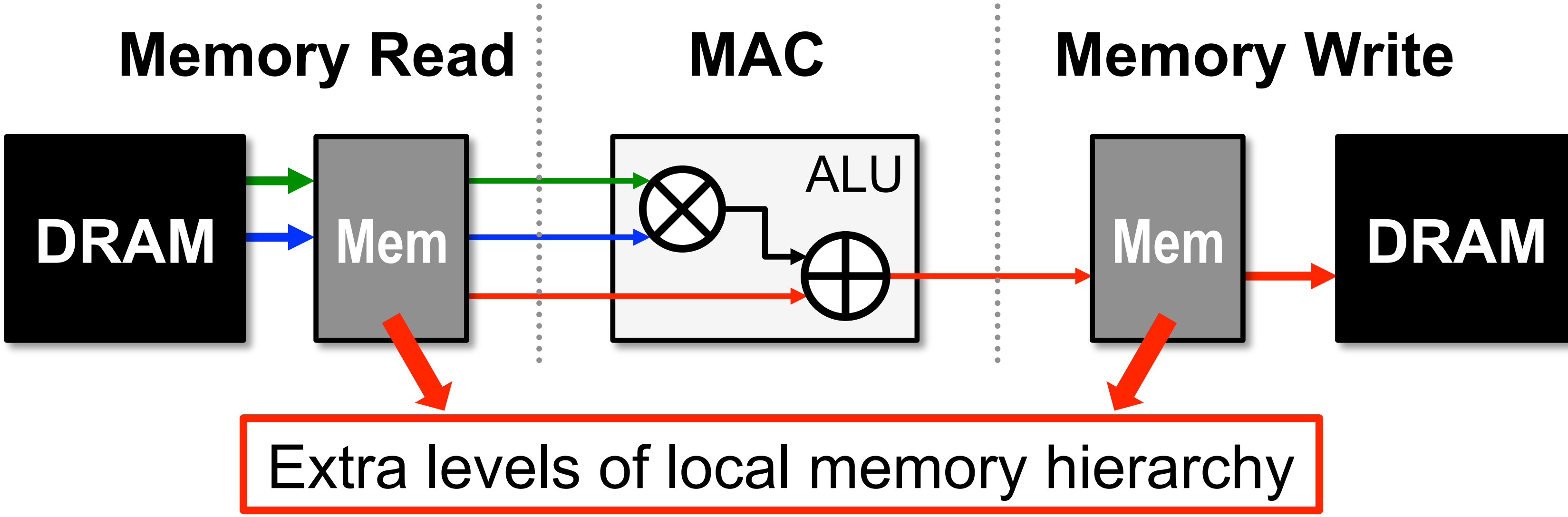


* multiply-and-accumulate

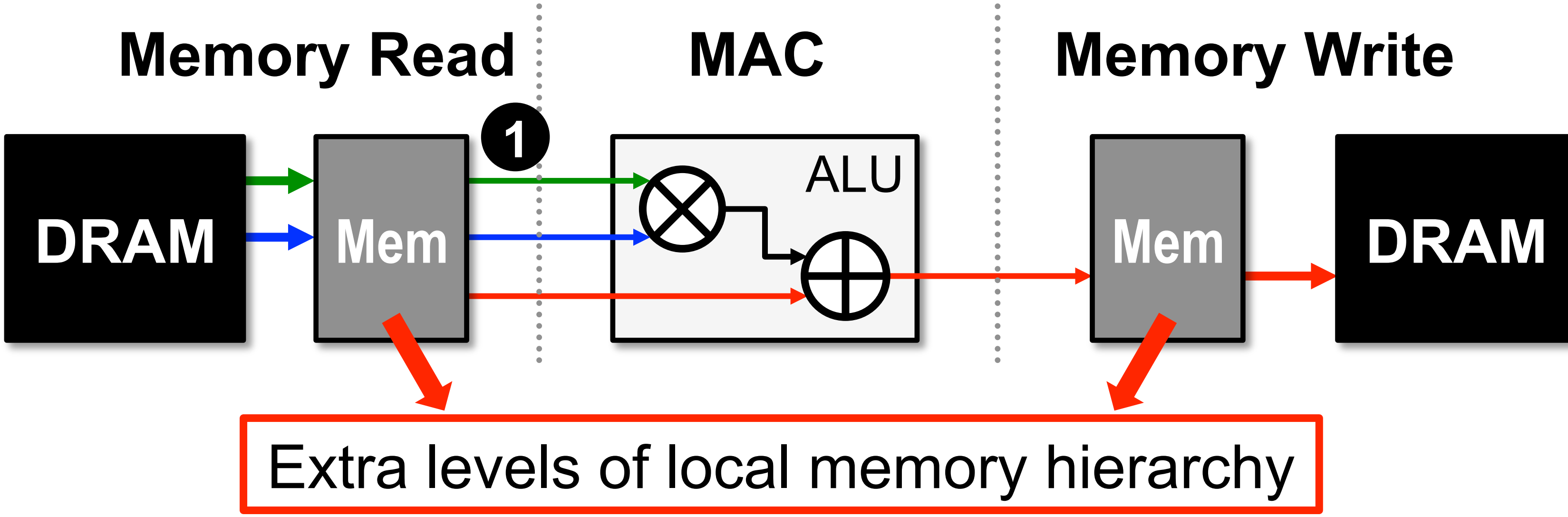
Worst Case: all memory R/W are **DRAM** accesses

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

Memory Access is the Bottleneck



Memory Access is the Bottleneck

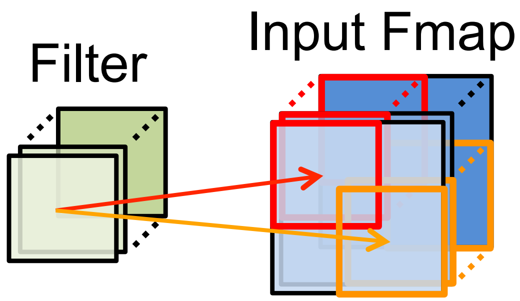


Opportunities: 1 **data reuse**

Types of Data Reuse in DNN

Convolutional Reuse

CONV layers only
(sliding window)

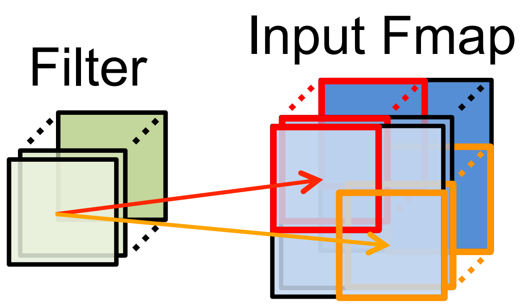


Reuse: Activations
Filter weights

Types of Data Reuse in DNN

Convolutional Reuse

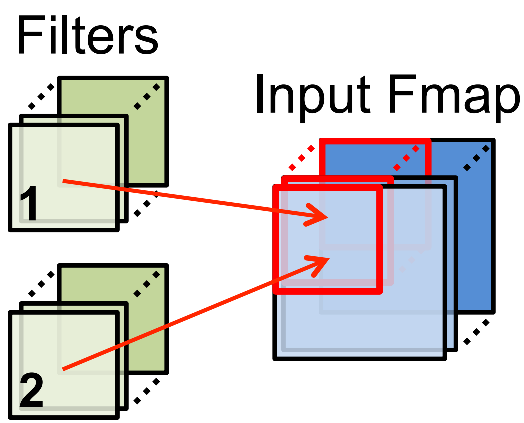
CONV layers only
(sliding window)



Reuse: Activations
Filter weights

Fmap Reuse

CONV and FC layers

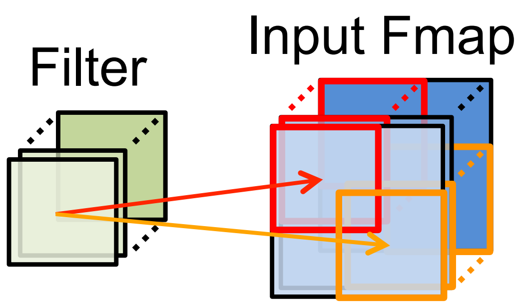


Reuse: Activations

Types of Data Reuse in DNN

Convolutional Reuse

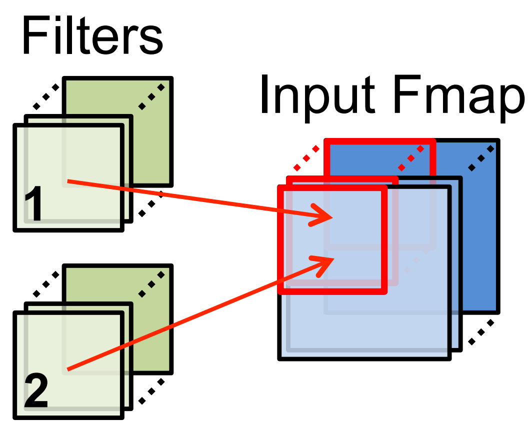
CONV layers only
(sliding window)



Reuse: Activations
Filter weights

Fmap Reuse

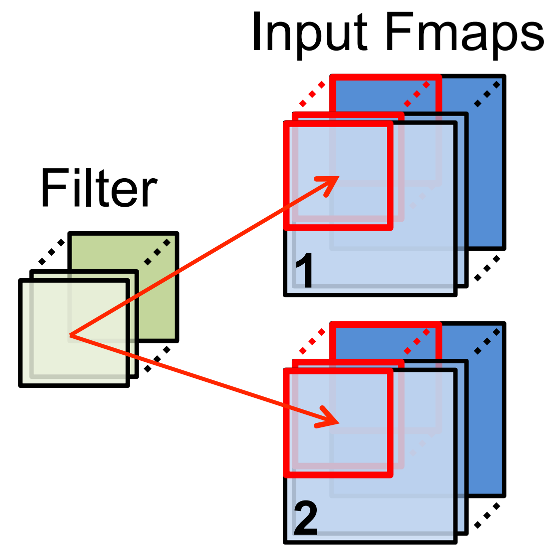
CONV and FC layers



Reuse: Activations

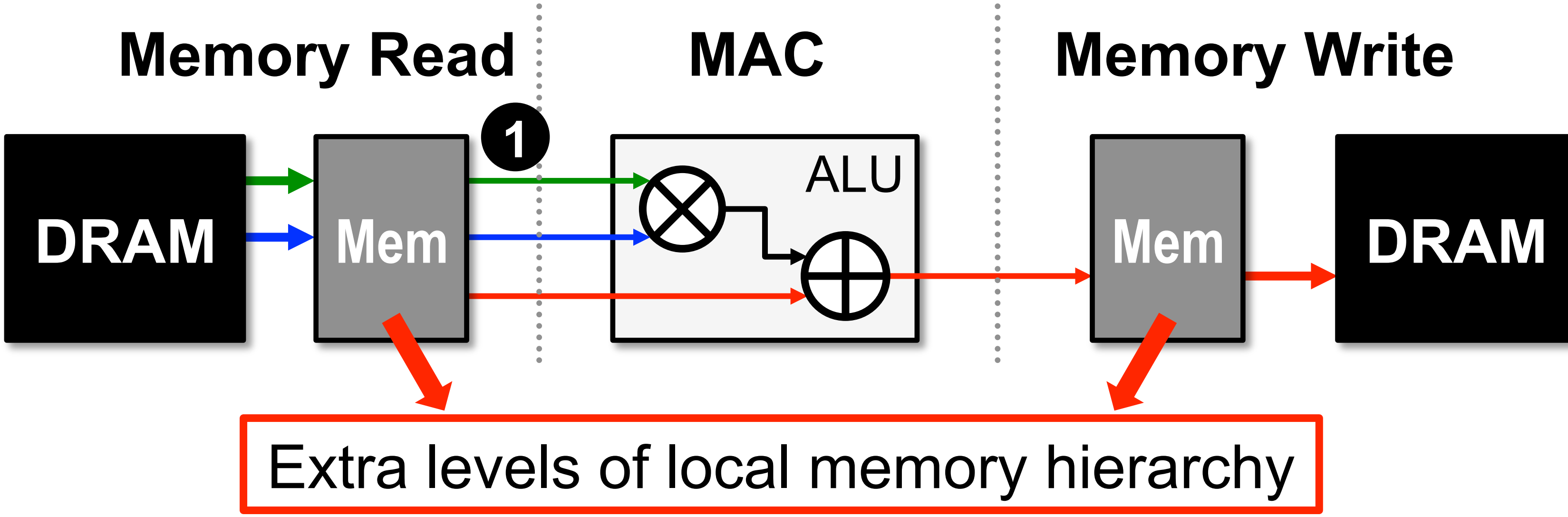
Filter Reuse

CONV and FC layers
(batch size > 1)



Reuse: Filter weights

Memory Access is the Bottleneck

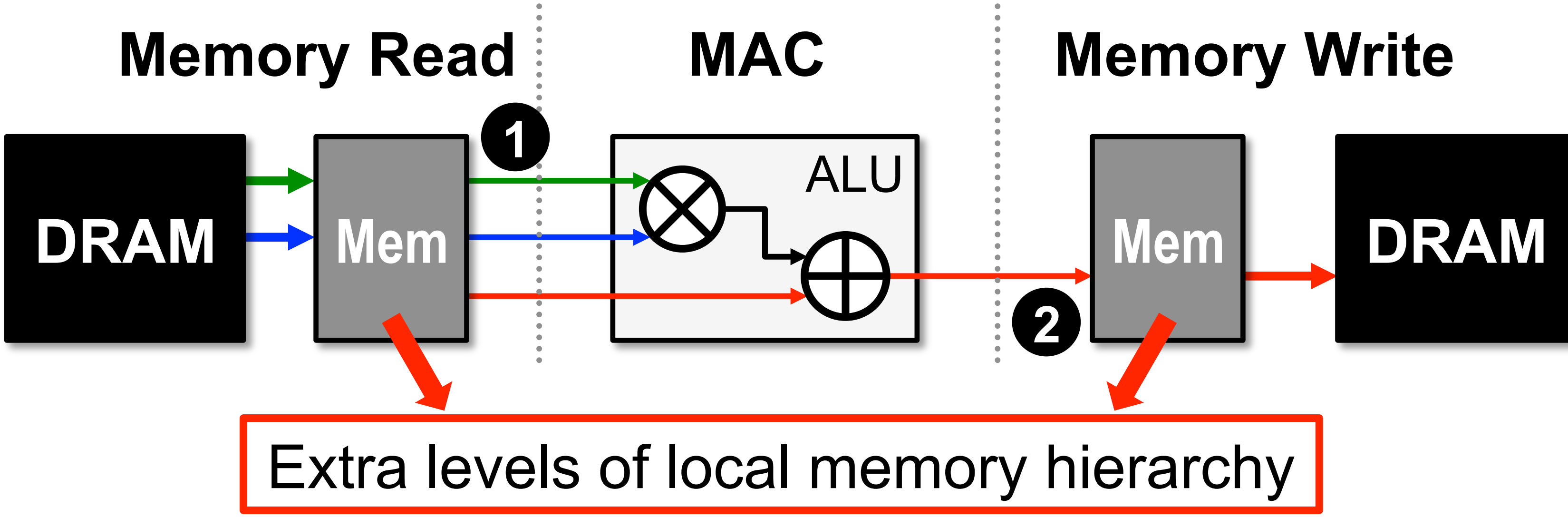


Opportunities: **1 data reuse**

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

** AlexNet CONV layers

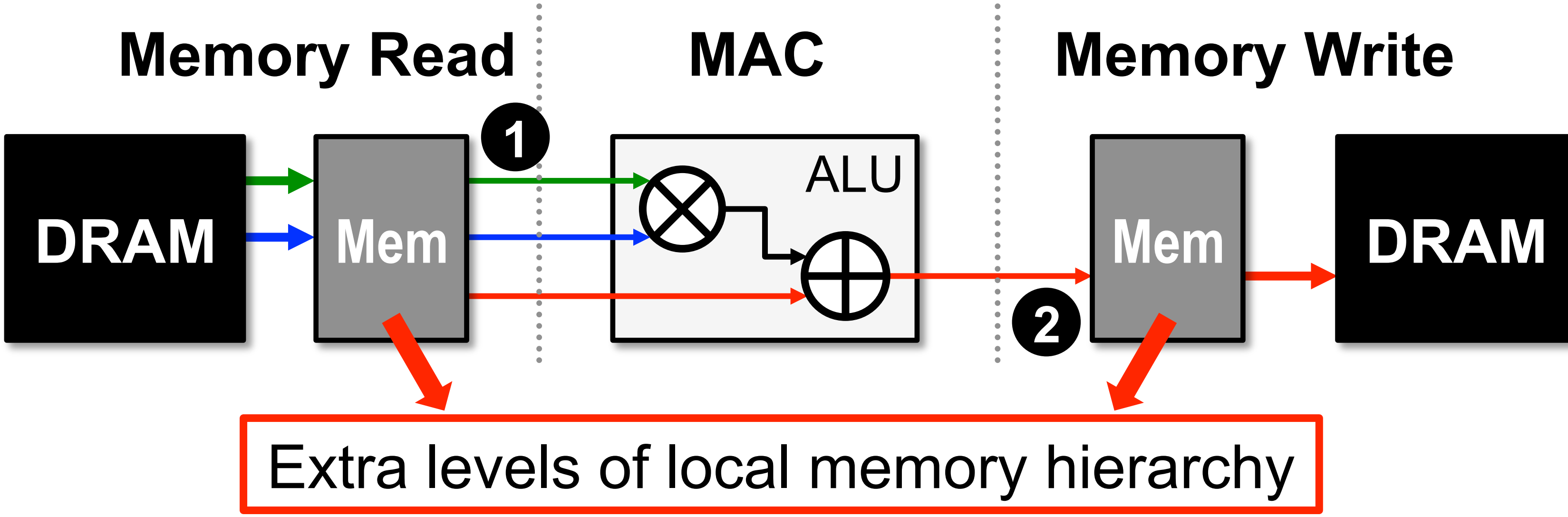
Memory Access is the Bottleneck



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

Memory Access is the Bottleneck



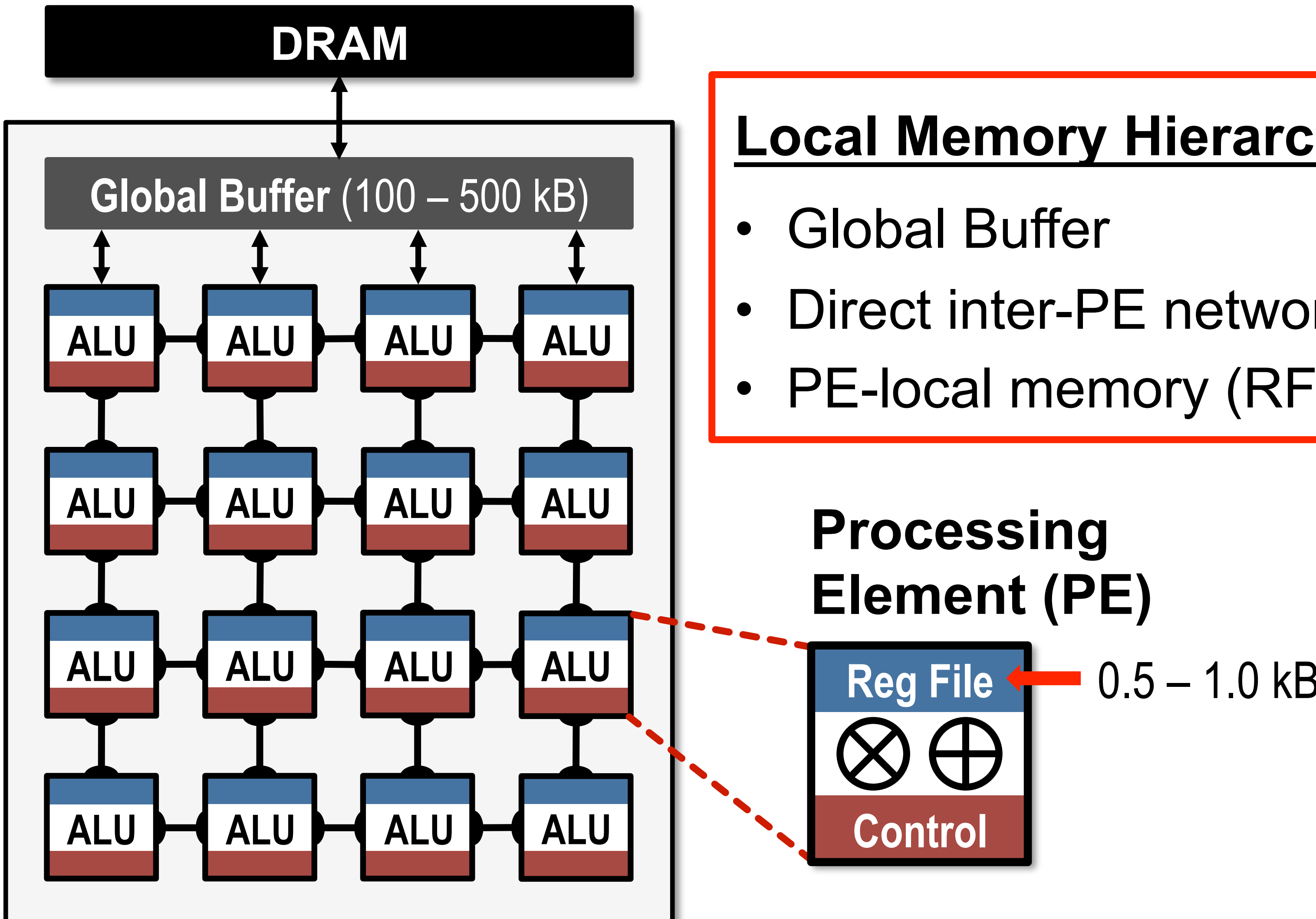
Opportunities: ❶ **data reuse** ❷ **local accumulation**

❶ Can reduce DRAM reads of filter/fmap 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)

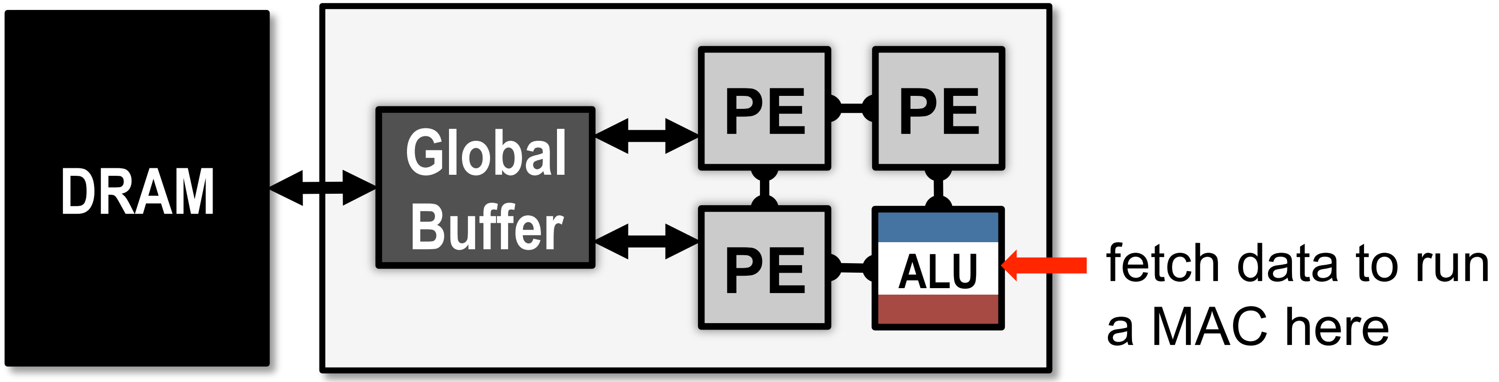
Spatial Architecture for DNN

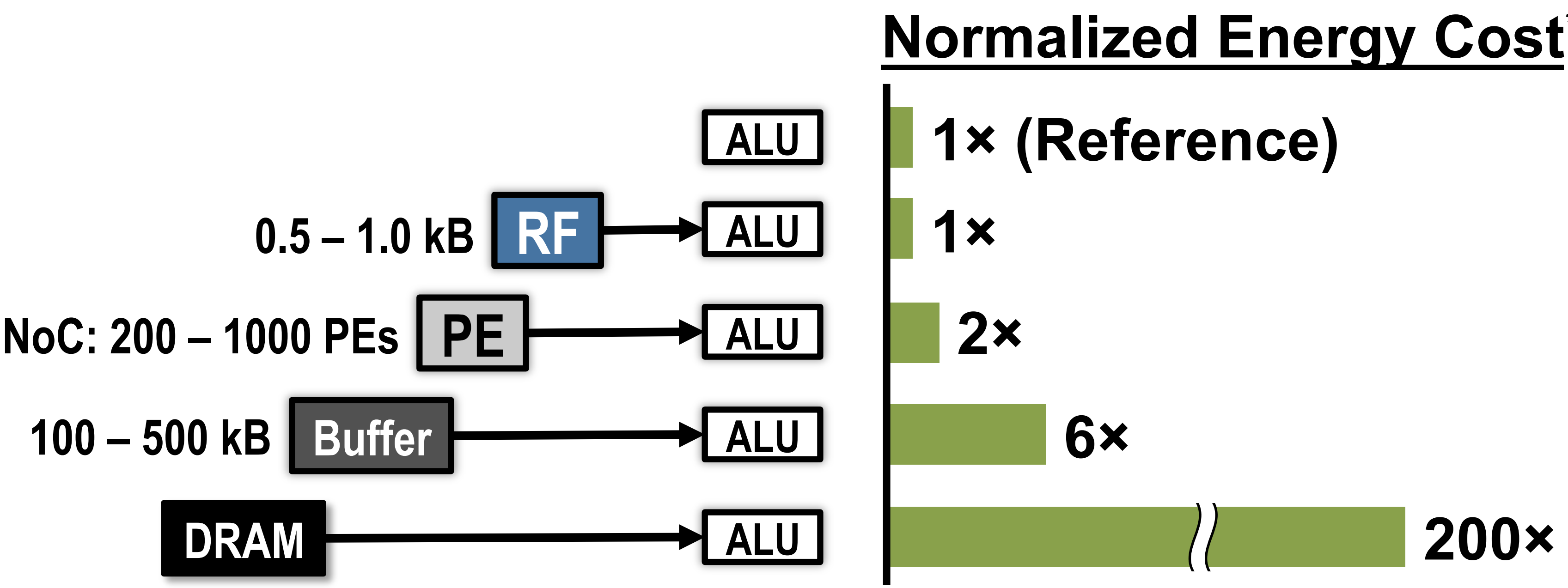


Local Memory Hierarchy

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

Low-Cost Local Data Access

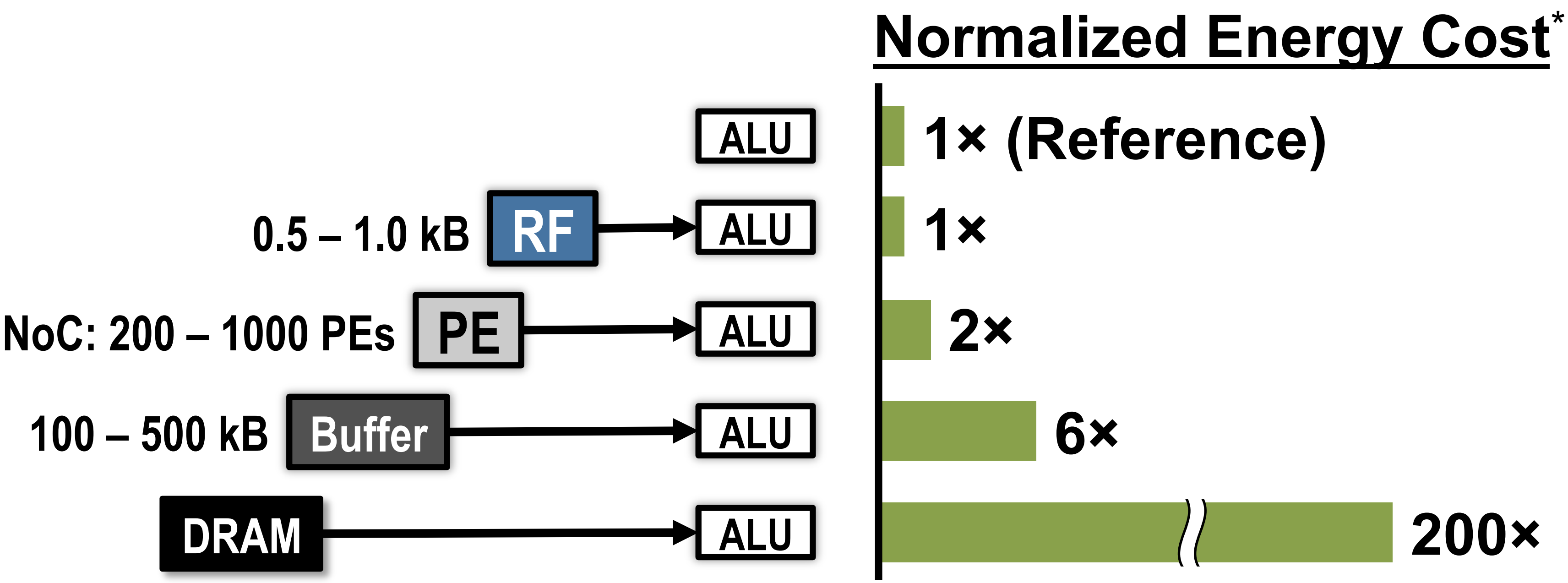




* measured from a commercial 65nm process

Low-Cost Local Data Access

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

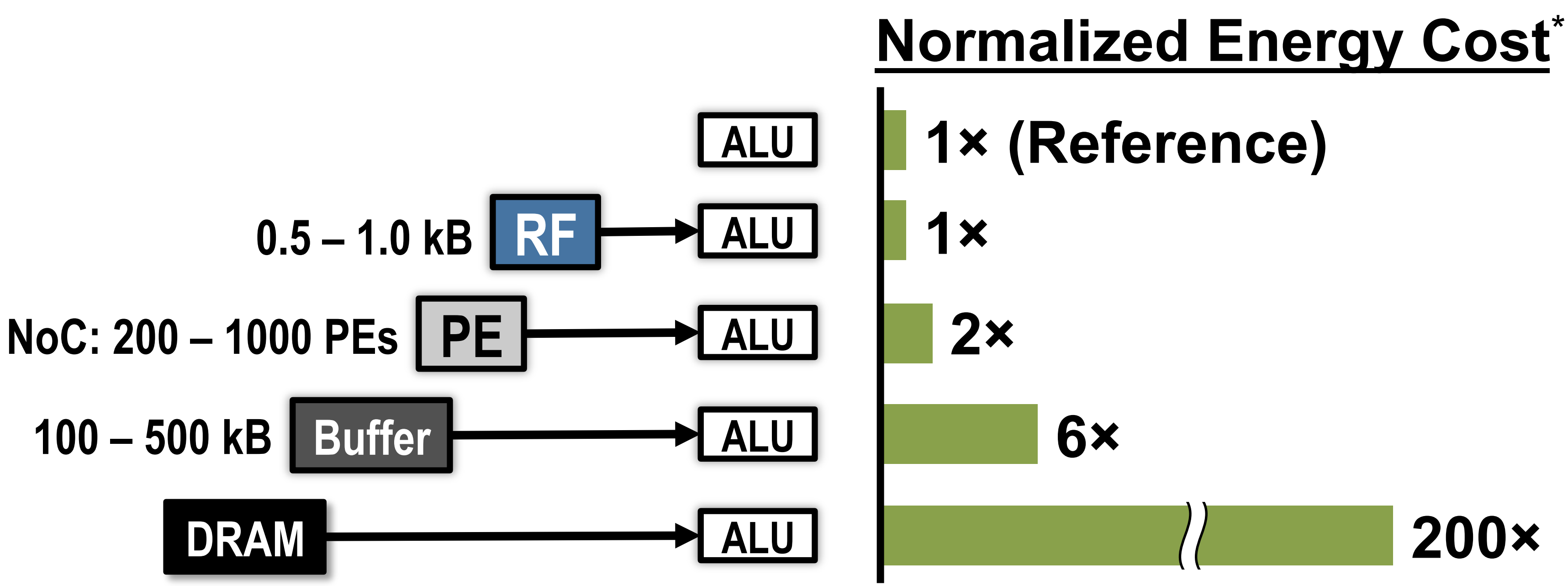


* measured from a commercial 65nm process

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!



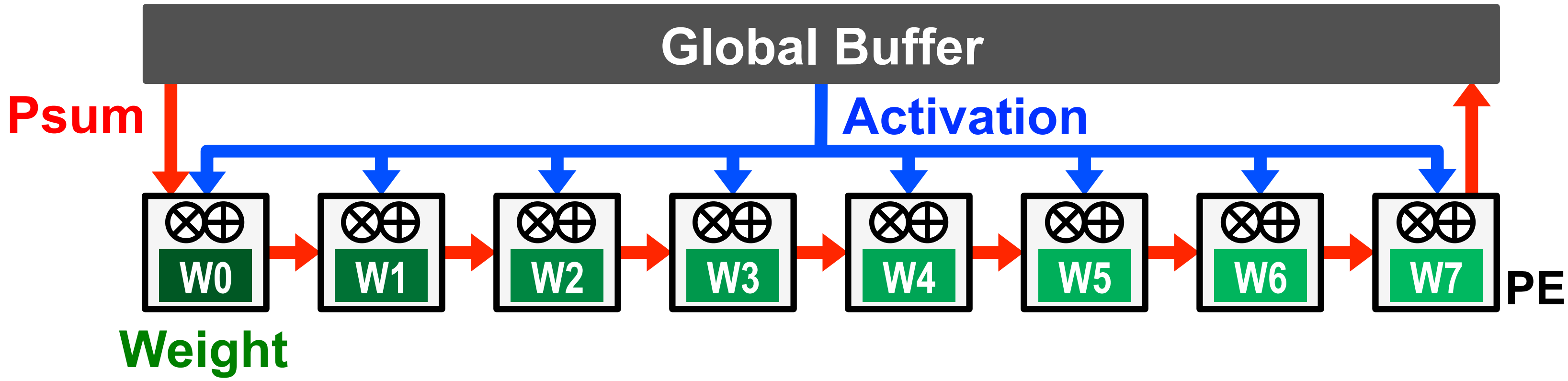
* measured from a commercial 65nm process

Dataflow Taxonomy

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

[Chen et al., ISCA 2016]

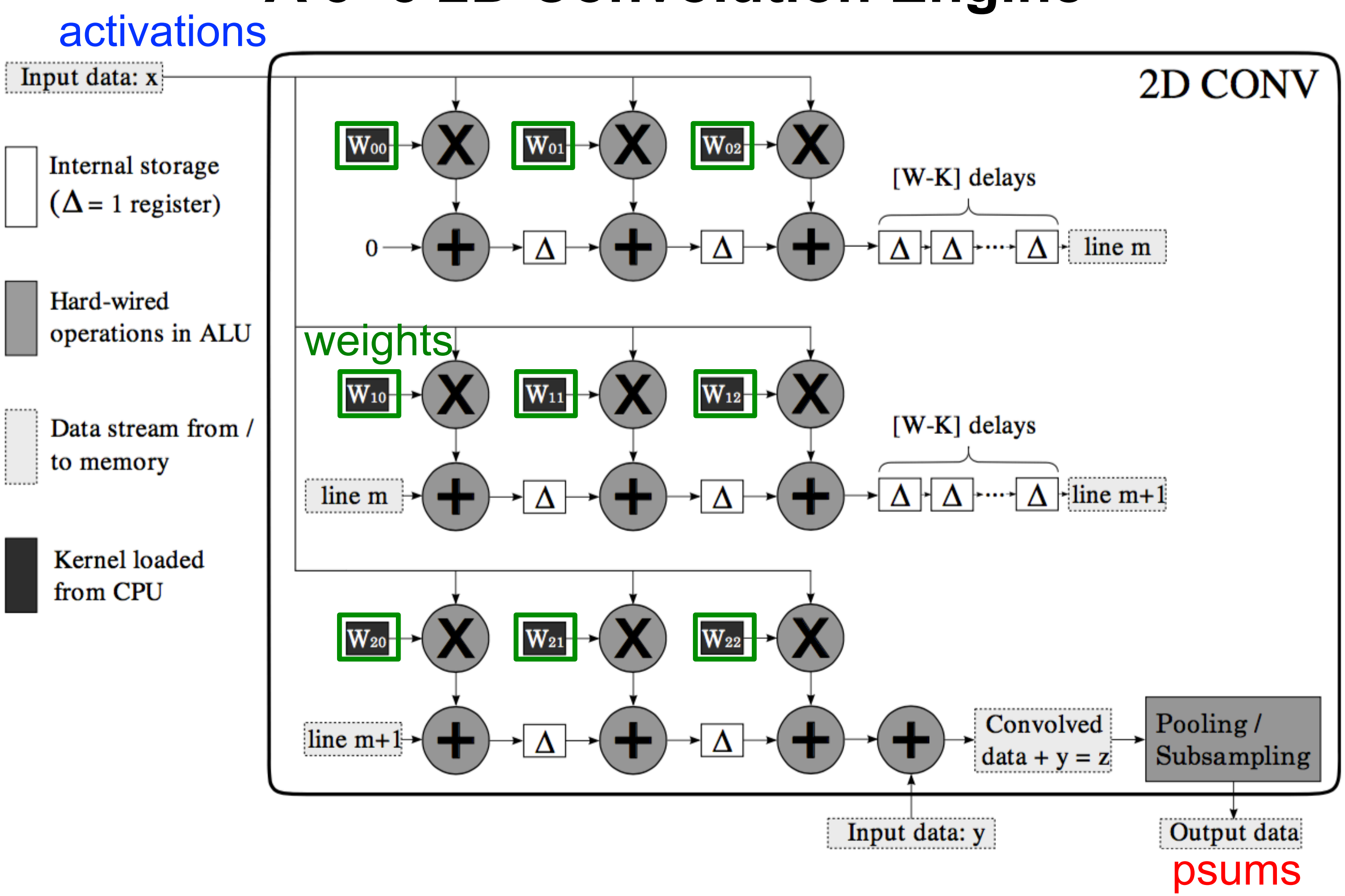
Weight Stationary (WS)



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

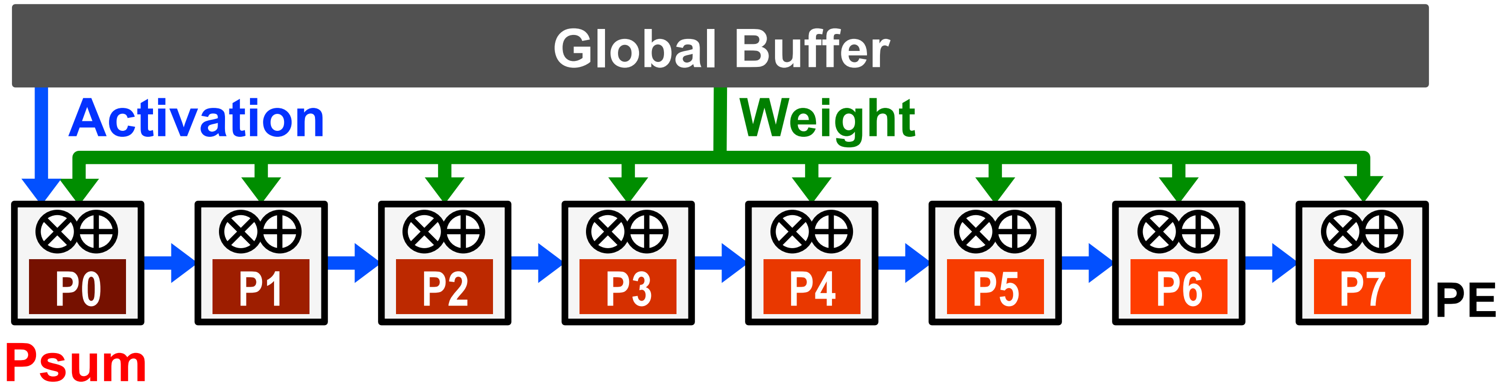
WS Example: nn-X (NeuFlow)

A 3×3 2D Convolution Engine



[Farabet et al., ICCV 2009]

Output Stationary (OS)

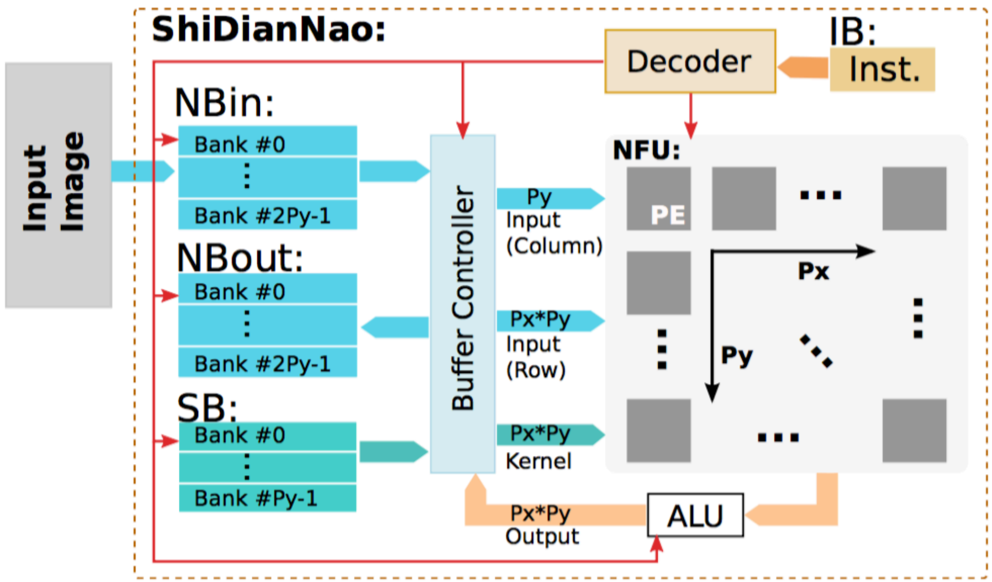


- **Minimize** **partial sum** R/W energy consumption
  - maximize local accumulation

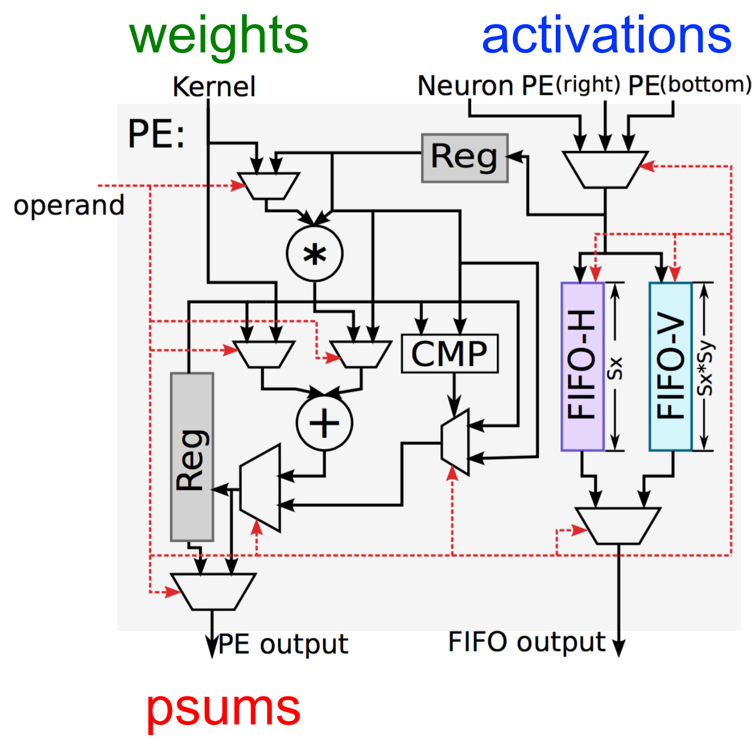
- **Broadcast/Multicast** **filter weights** and **reuse** **activations** **spatially** across the PE array

OS Example: ShiDianNao

Top-Level Architecture

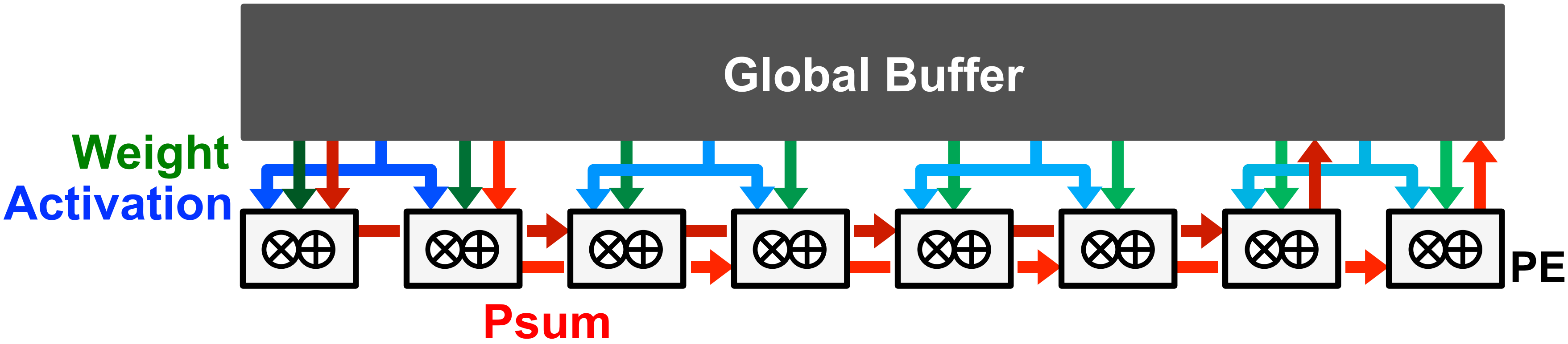


PE Architecture



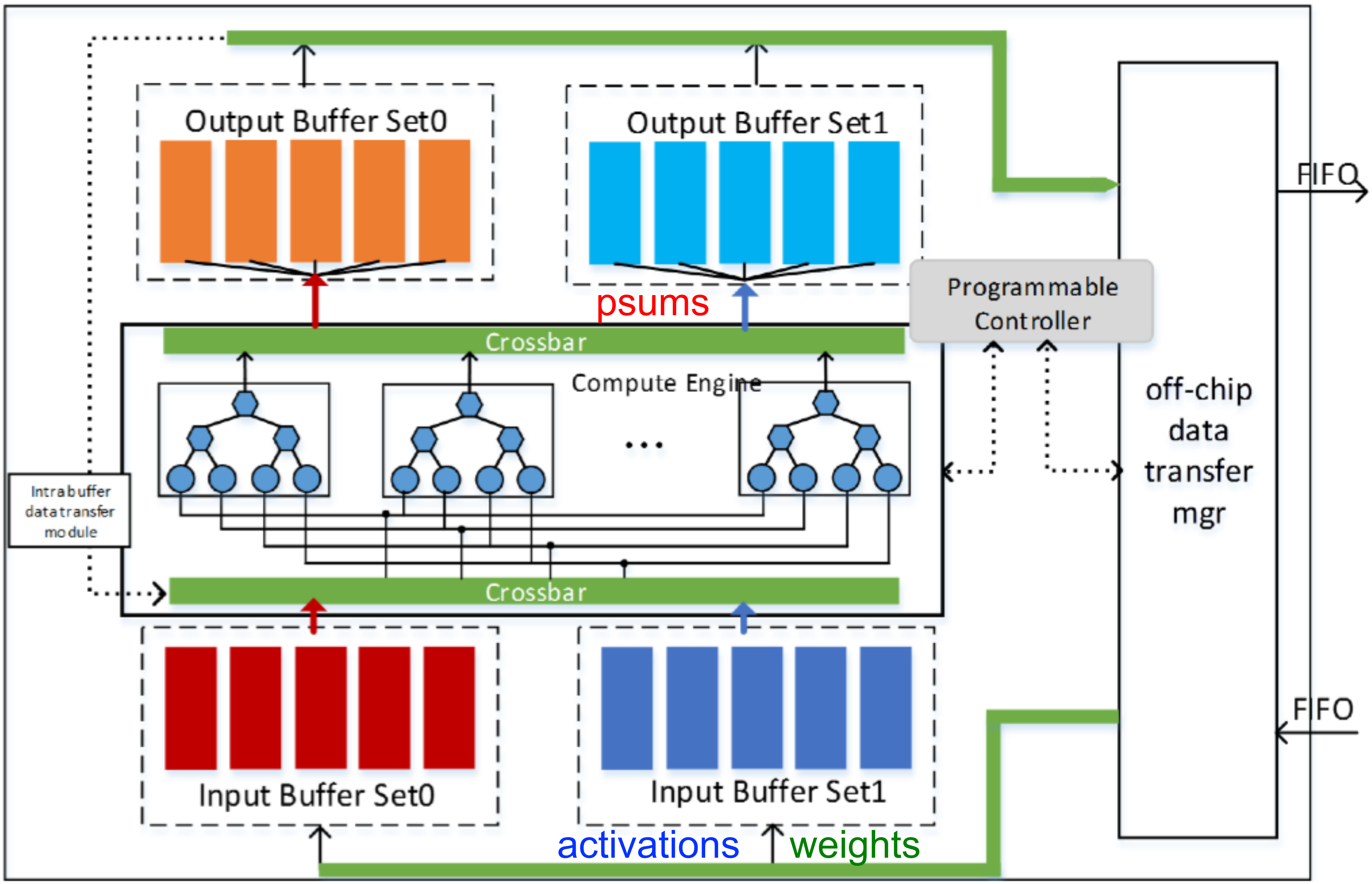
[Du et al., ISCA 2015]

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

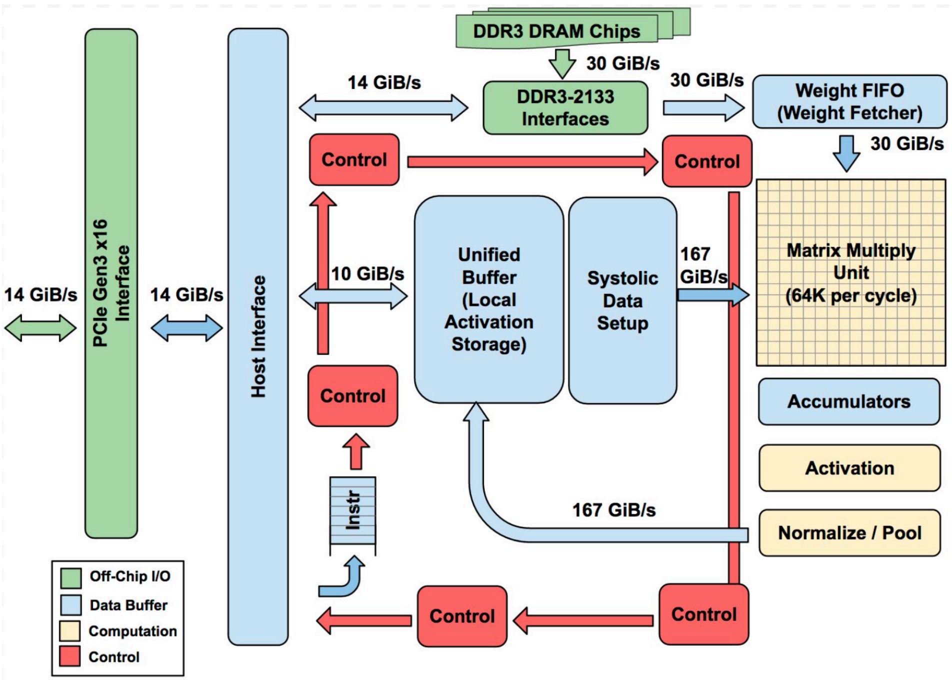
NLR Example: UCLA



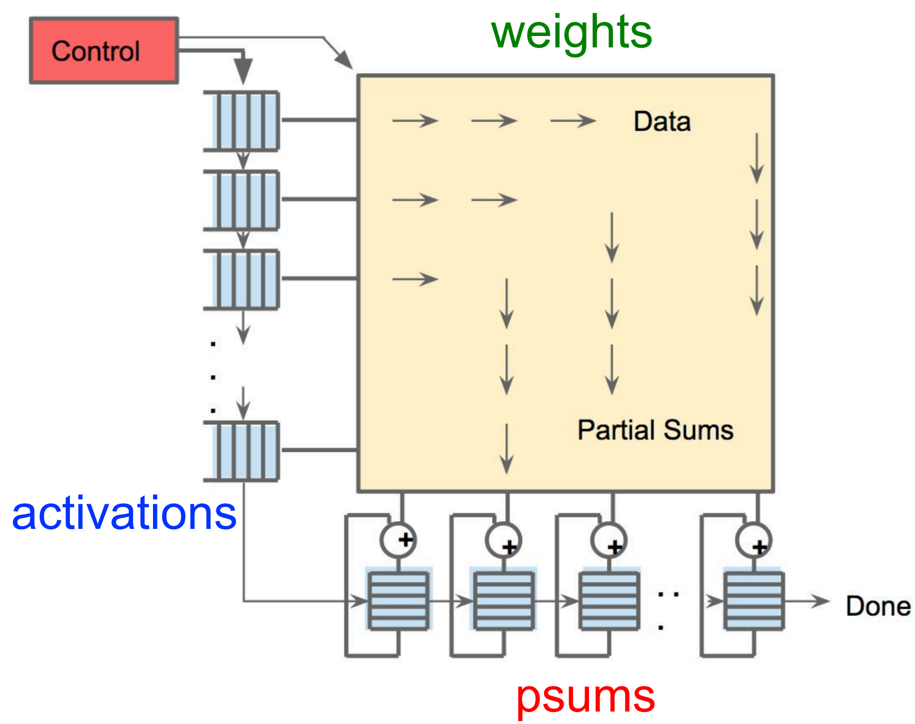
[Zhang et al., FPGA 2015]

NLR Example: TPU

Top-Level Architecture



Matrix Multiply Unit



[Jouppi et al., ISCA 2017]

Taxonomy: More Examples

- **Weight Stationary (WS)**

  [**Chakradhar**, *ISCA* 2010] [**nn-X (NeuFlow)**, *CVPRW* 2014]
  [**Park**, *ISSCC* 2015] [**ISAAC**, *ISCA* 2016] [**PRIME**, *ISCA* 2016]

- **Output Stationary (OS)**

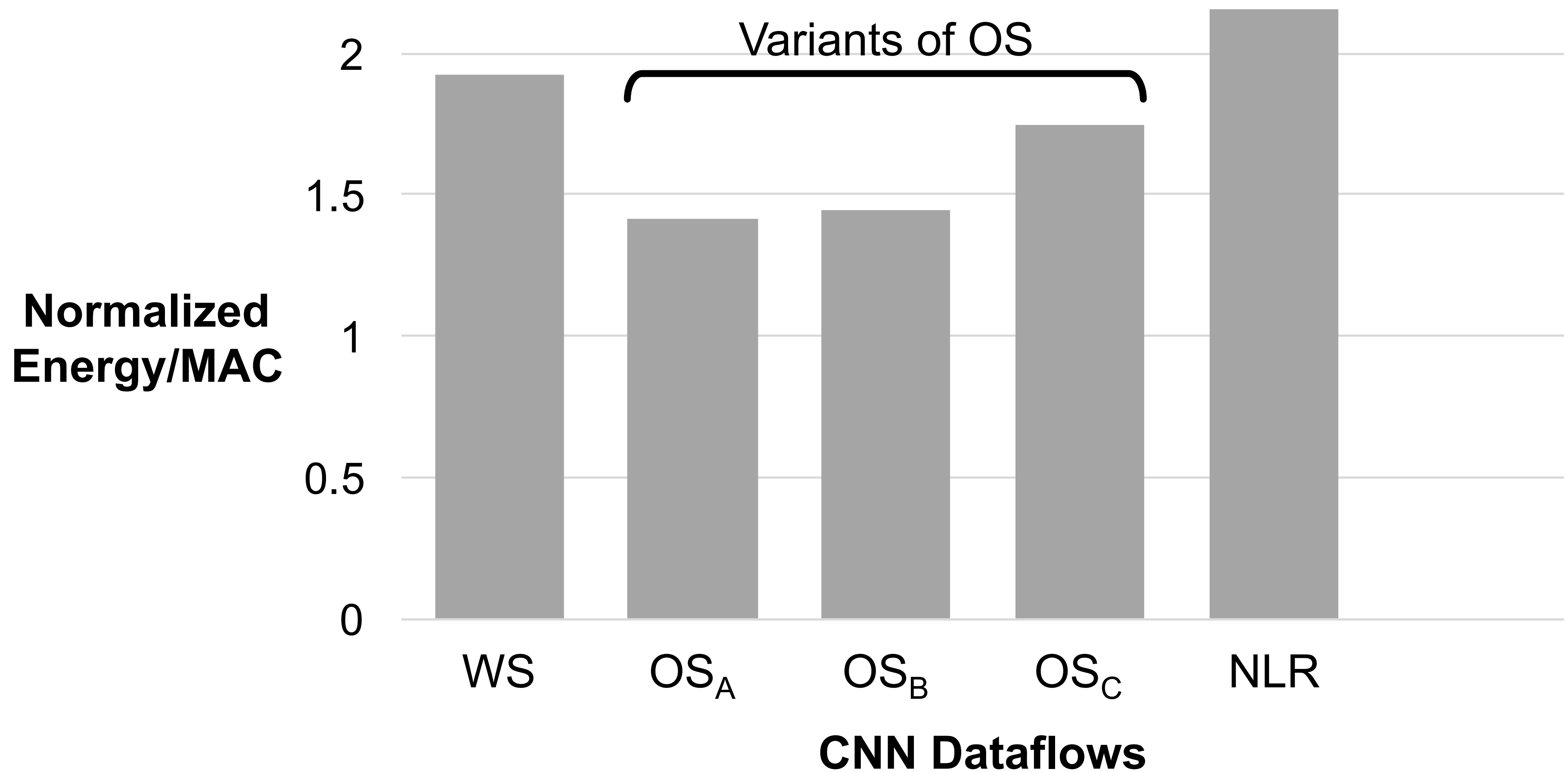
  [**Peemen**, *ICCD* 2013] [**ShiDianNao**, *ISCA* 2015]
  [**Gupta**, *ICML* 2015] [**Moons**, *VLSI* 2016]

- **No Local Reuse (NLR)**

  [**DianNao**, *ASPLOS* 2014] [**DaDianNao**, *MICRO* 2014]
  [**Zhang**, *FPGA* 2015] [**TPU**, *ISCA* 2017]

Energy Efficiency Comparison

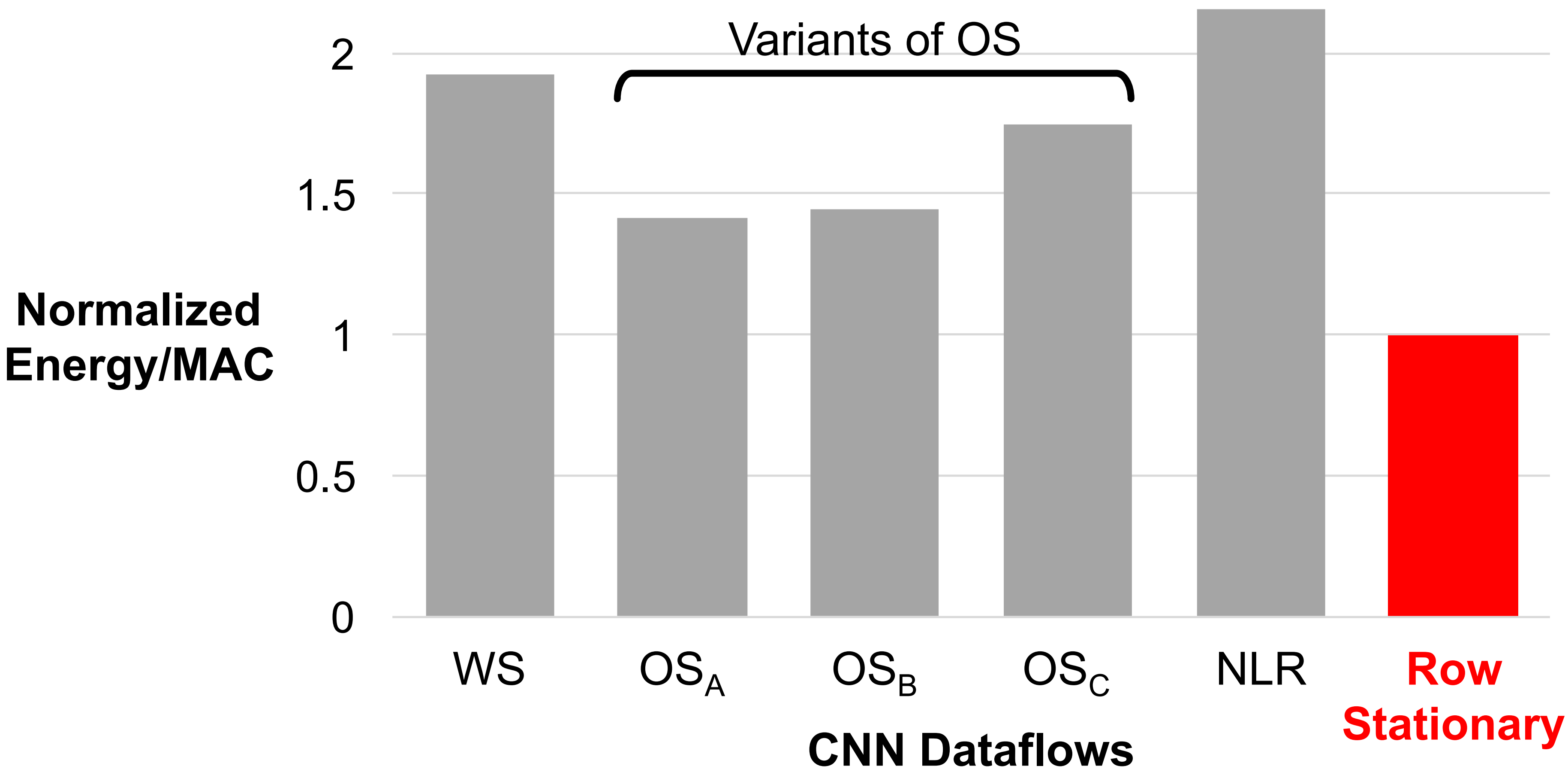
- Same total area
- AlexNet CONV layers
- 256 PEs
- Batch size = 16



[Chen et al., ISCA 2016]

Energy Efficiency Comparison

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



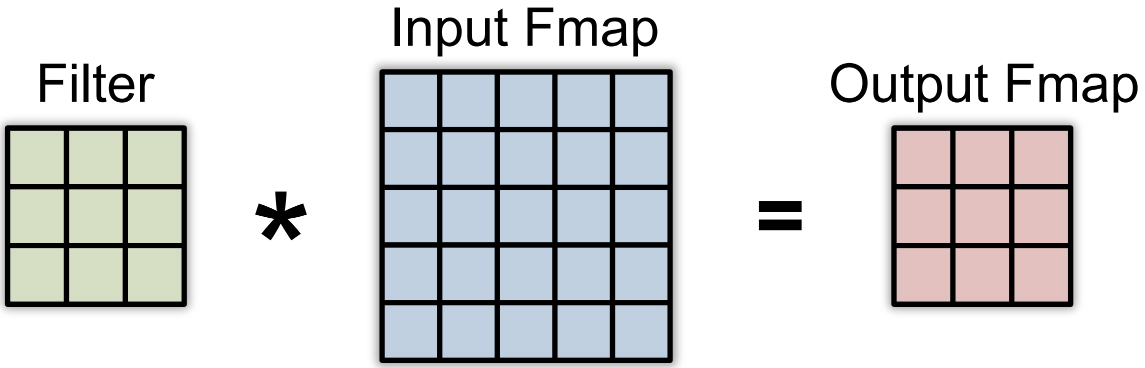
[Chen et al., ISCA 2016]

Energy-Efficient Dataflow: Row Stationary (RS)

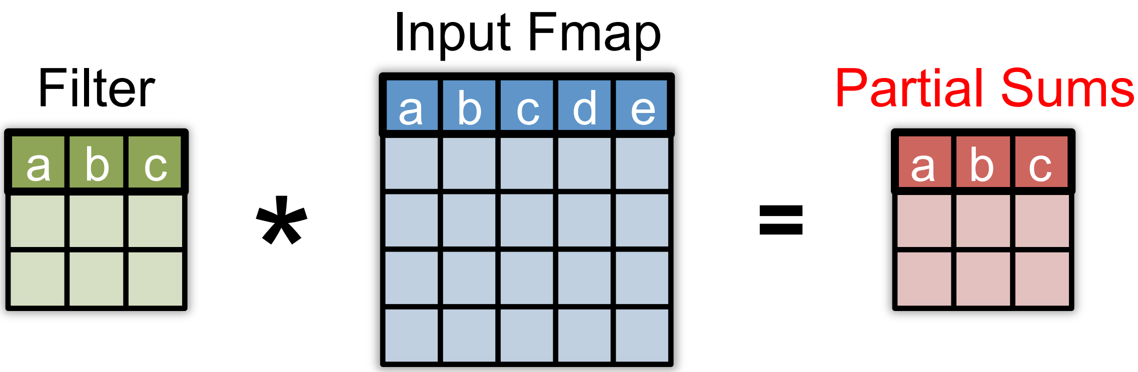
- **Maximize** reuse and accumulation at **RF**
- Optimize for **overall** energy efficiency instead for *only* a certain data type

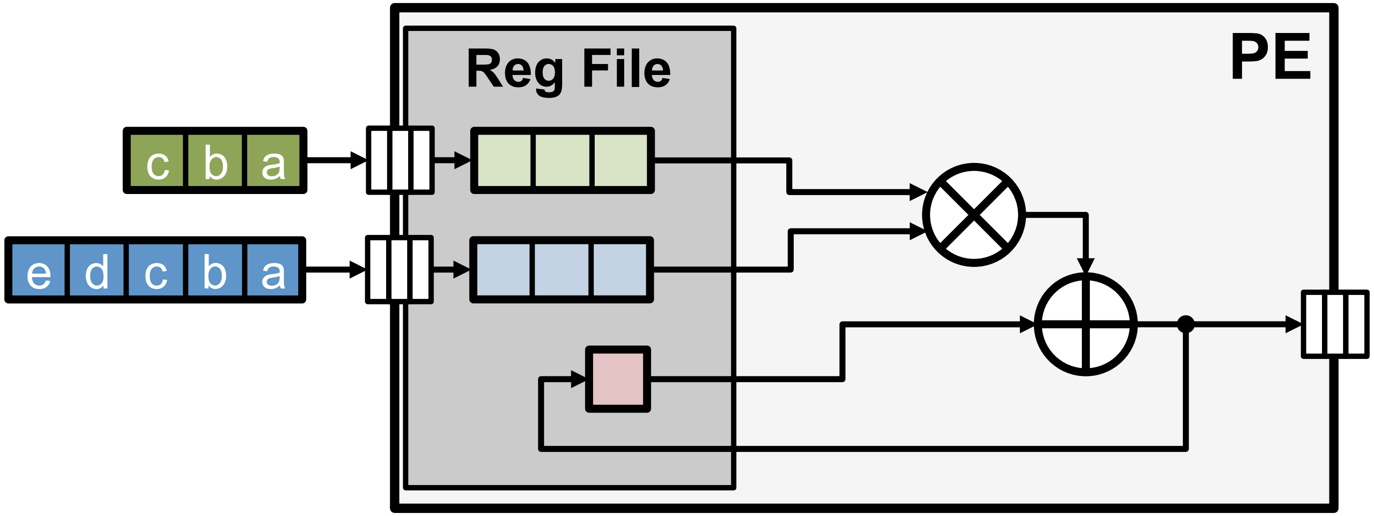
[Chen et al., ISCA 2016]

Row Stationary: Energy-efficient Dataflow

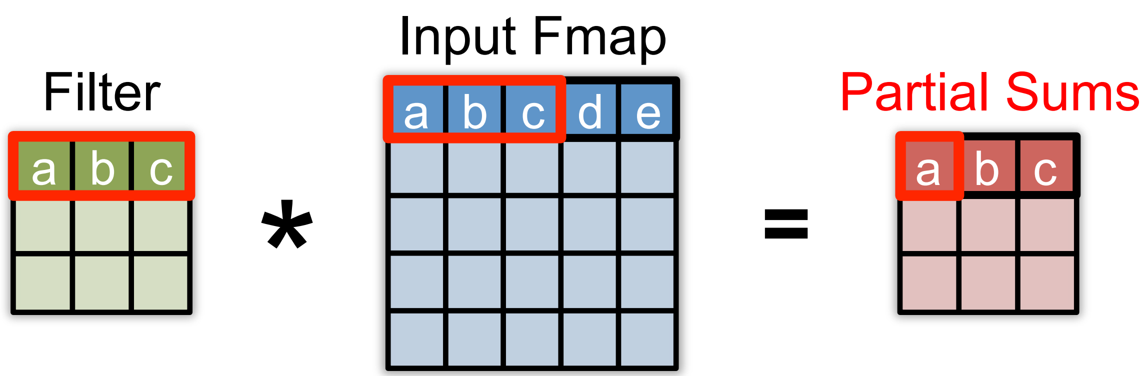


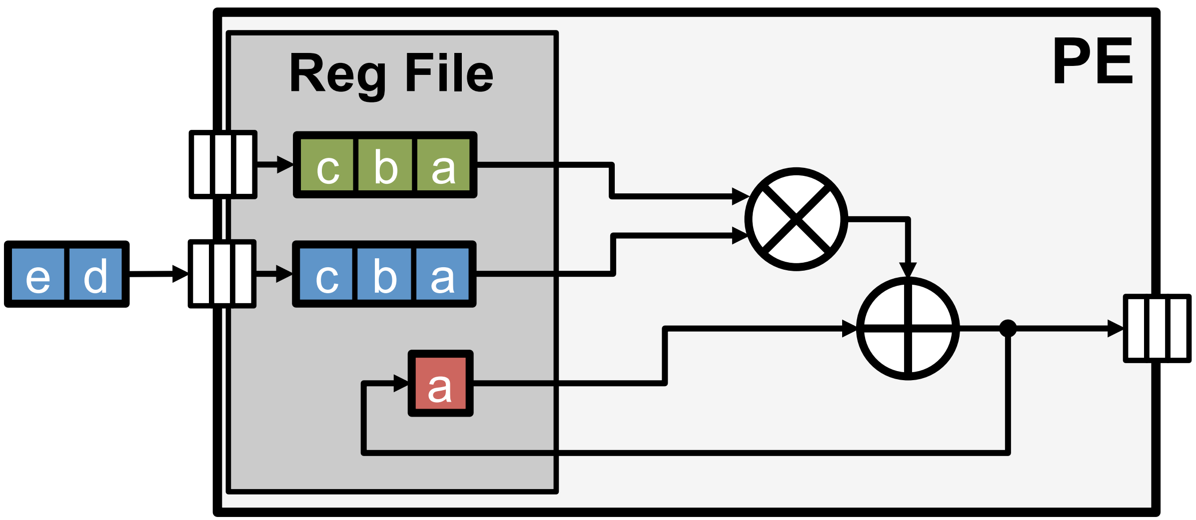
1D Row Convolution in PE



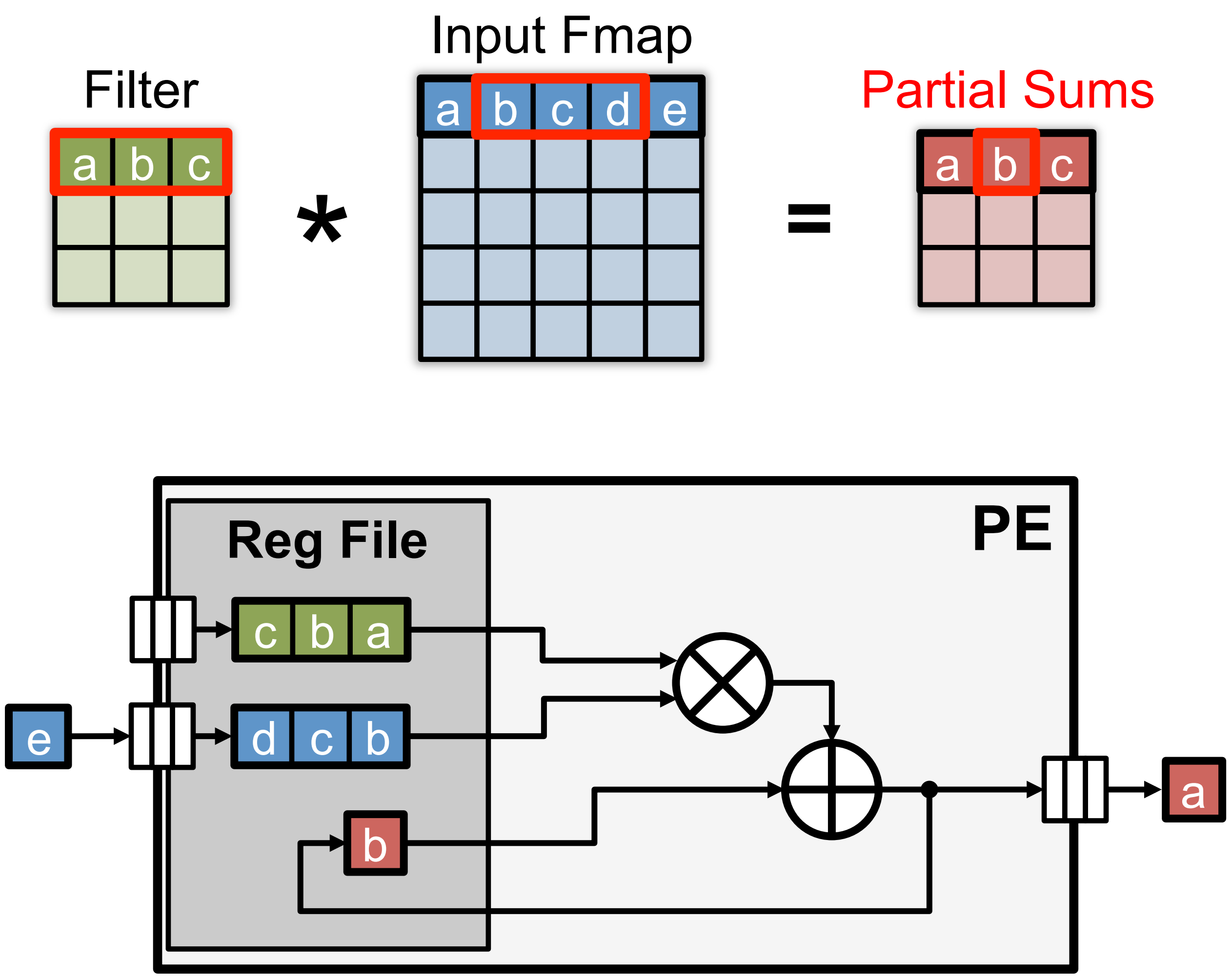


1D Row Convolution in PE

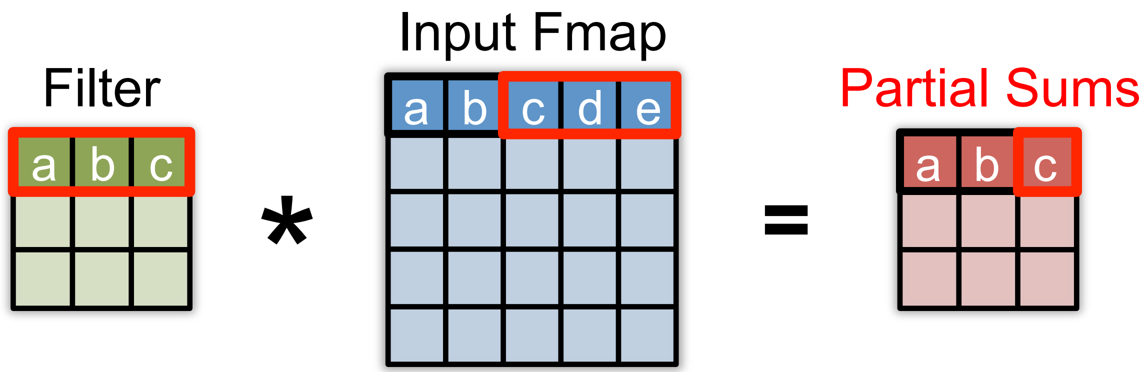


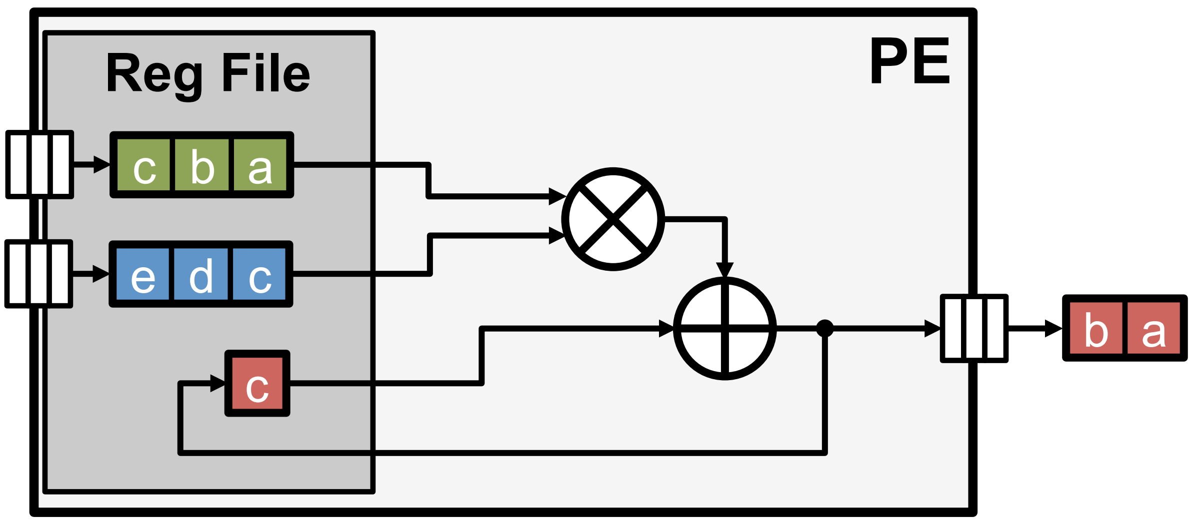


1D Row Convolution in PE



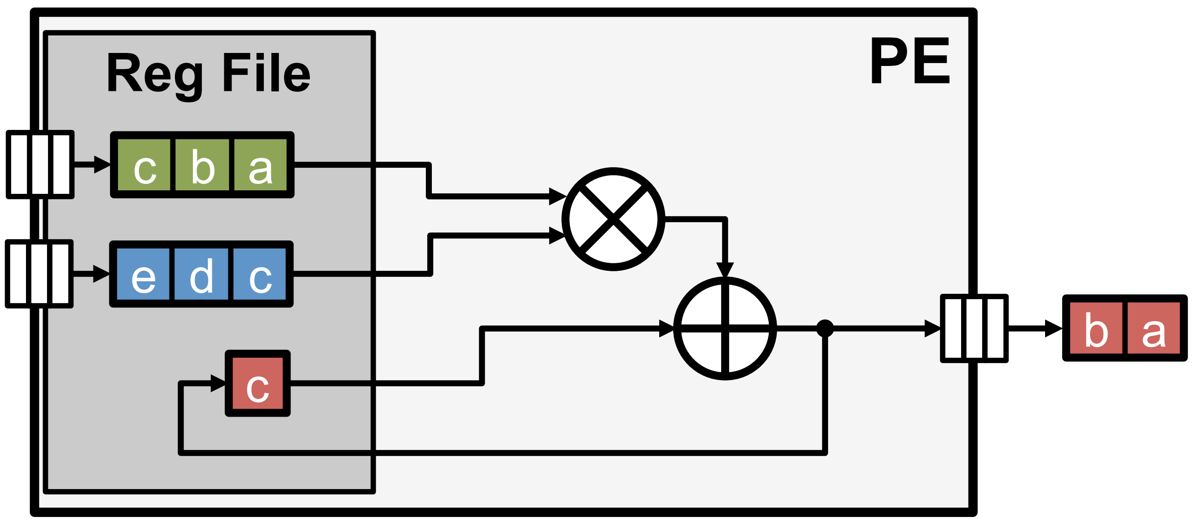
1D Row Convolution in PE



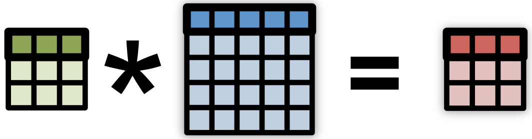


1D Row Convolution in PE

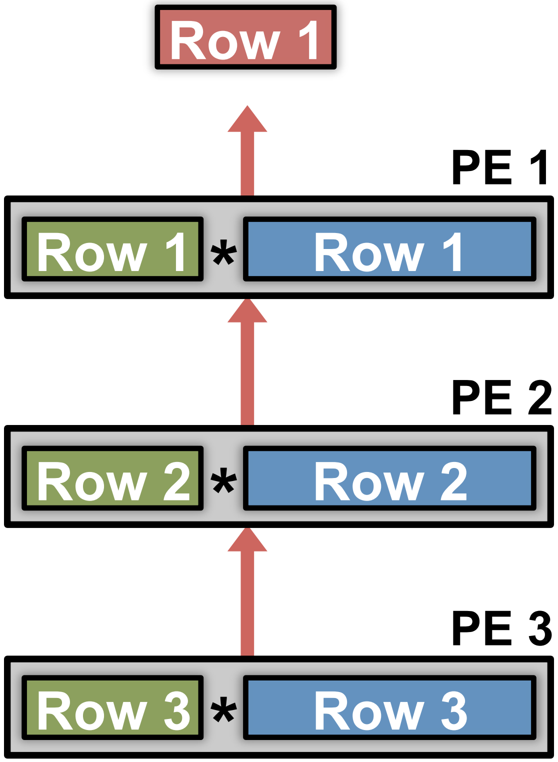
- Maximize row **convolutional reuse** in RF
  - Keep a **filter** row and **fmap** sliding window in RF
- Maximize row **psum** **accumulation** in RF

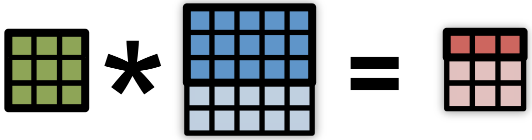


2D Convolution in PE Array

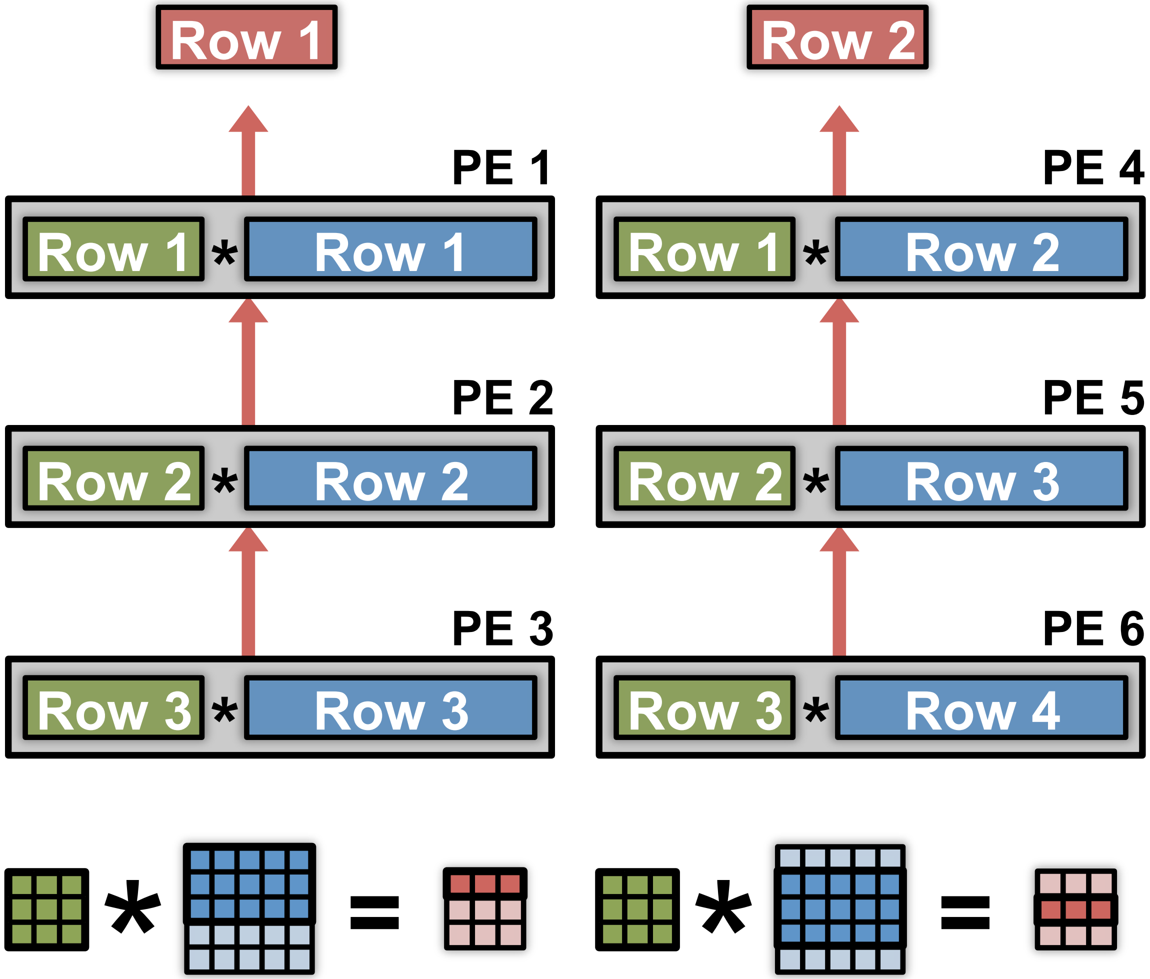


2D Convolution in PE Array

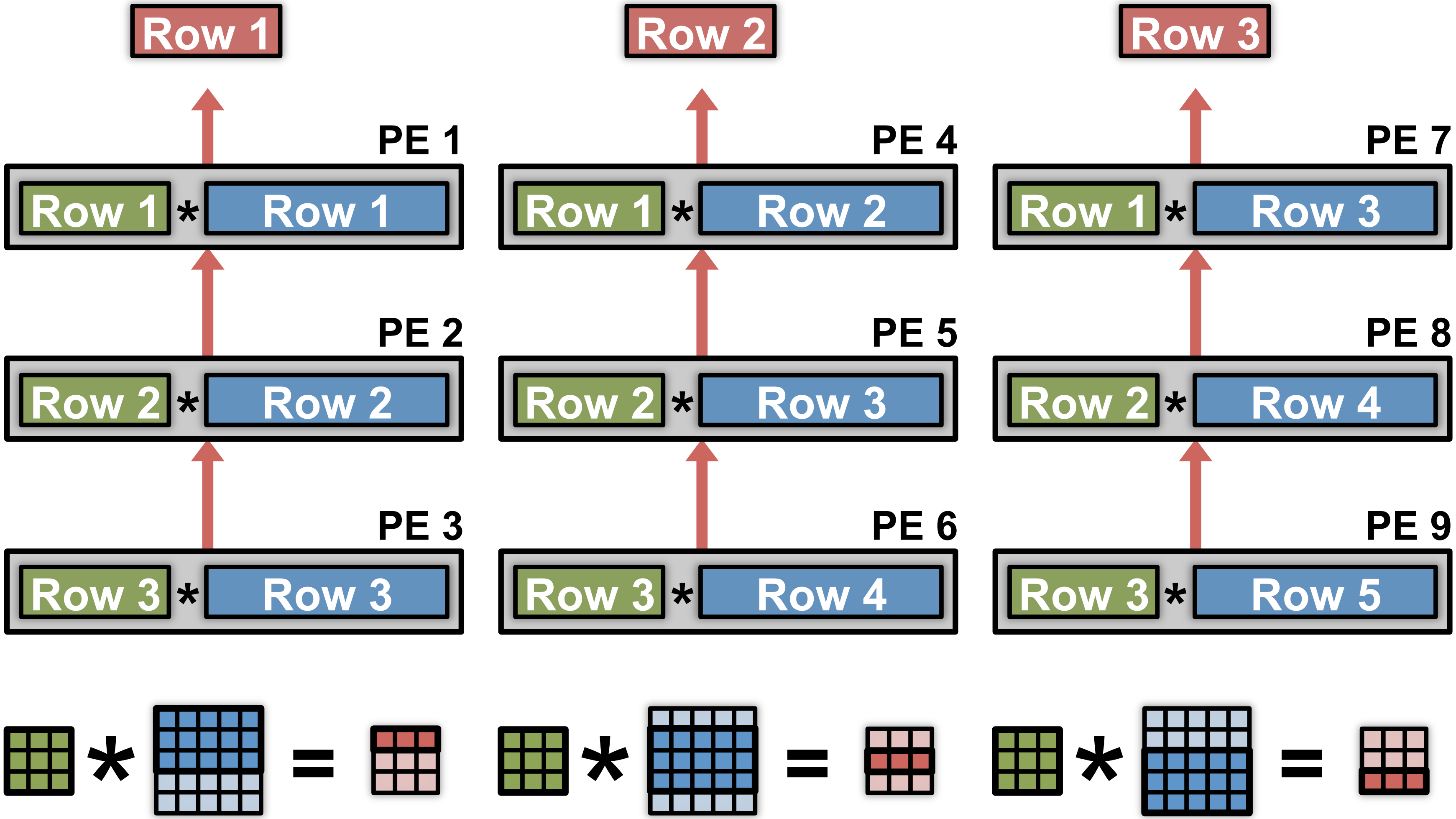




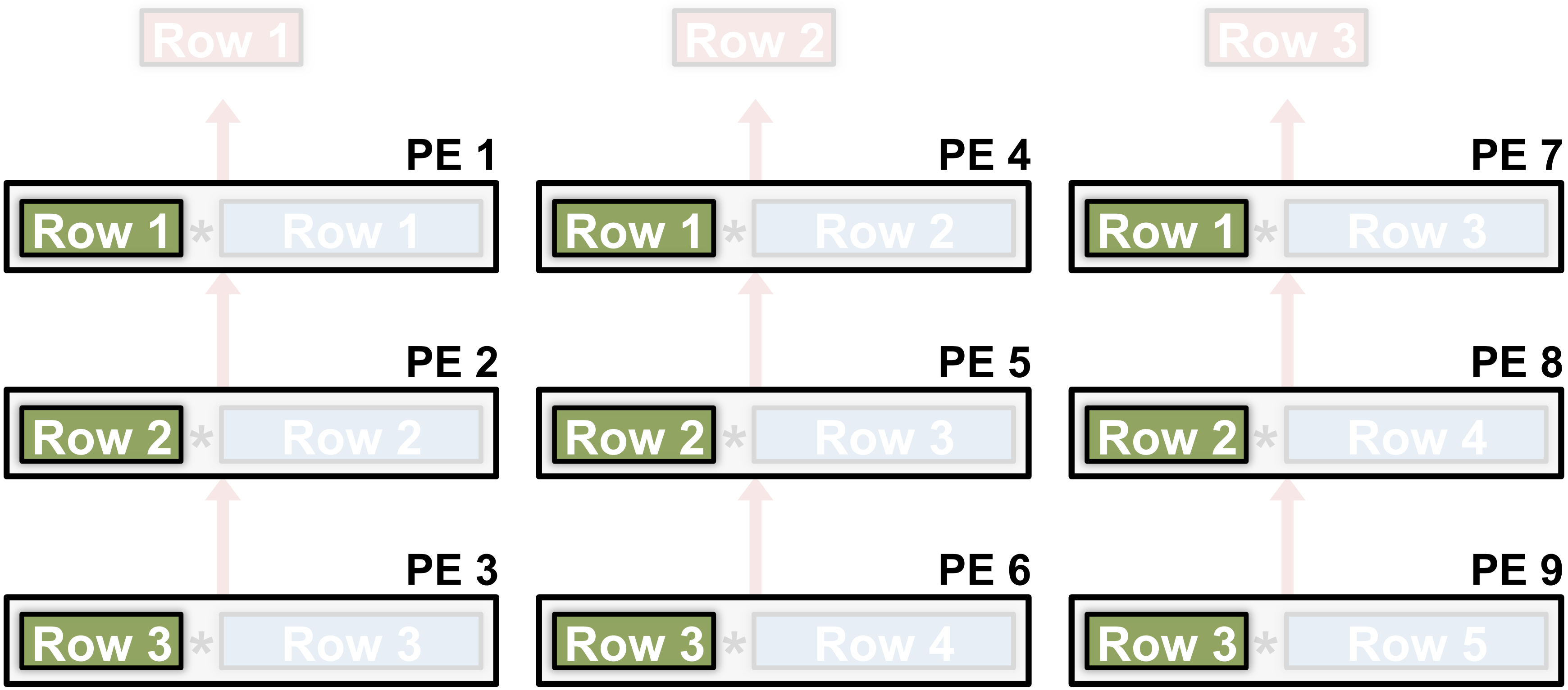
2D Convolution in PE Array



2D Convolution in PE Array

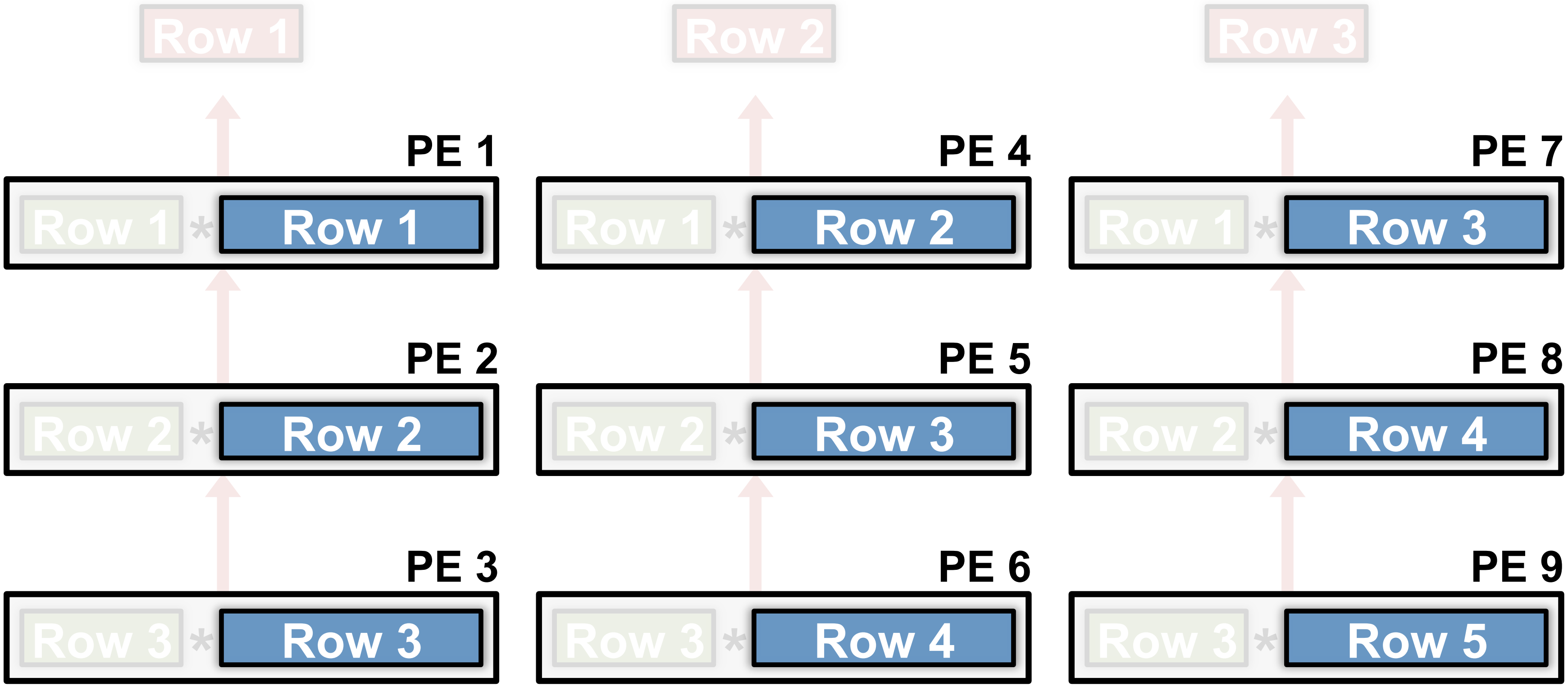


Convolutional Reuse Maximized



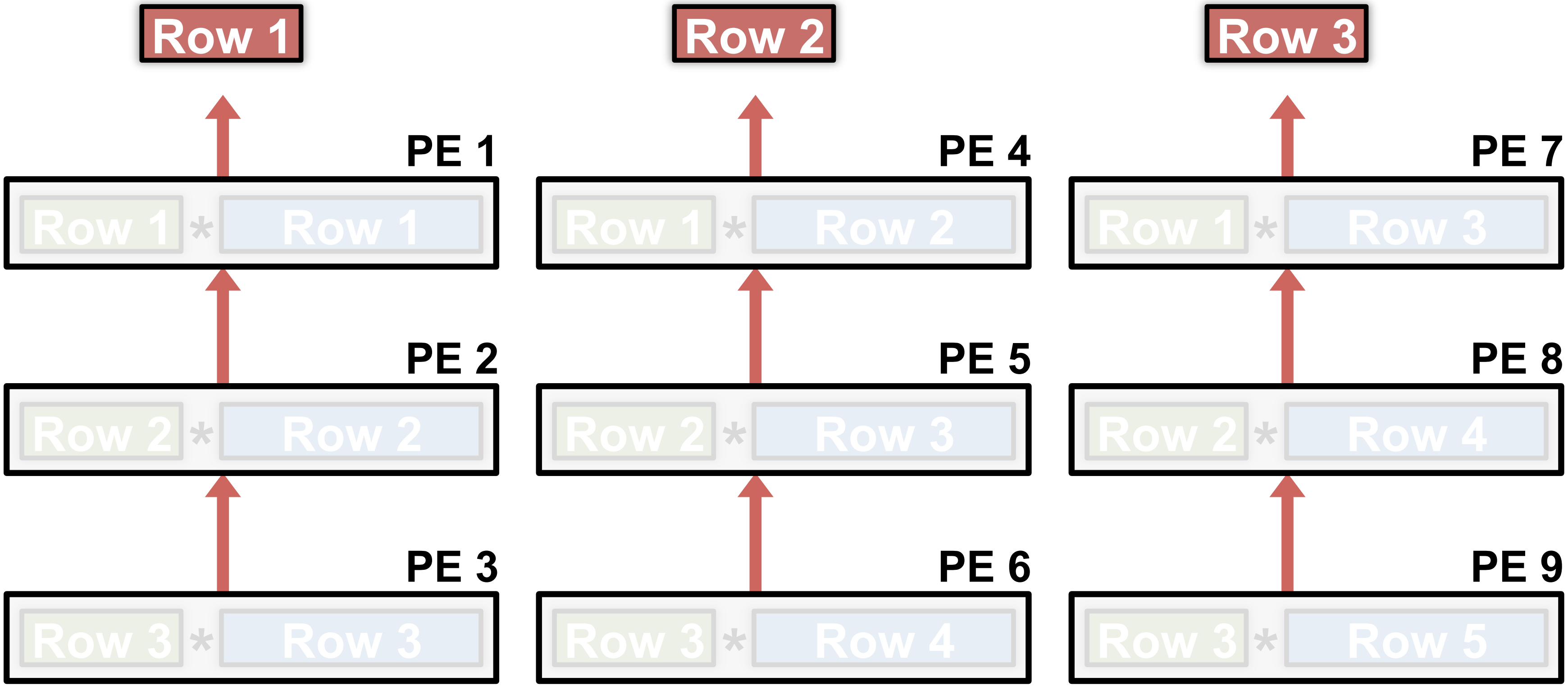
Filter rows are reused across PEs **horizontally**

Convolutional Reuse Maximized



Fmap rows are reused across PEs **diagonally**

Maximize 2D Accumulation in PE Array



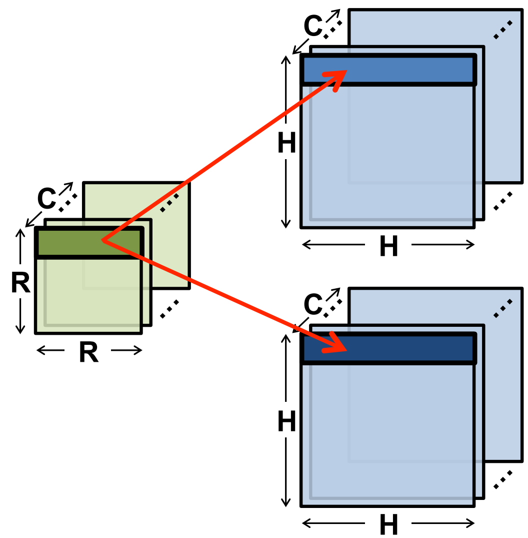
Partial sums accumulate across PEs **vertically**

Dimensions Beyond 2D Convolution

1 **Multiple Fmaps** 2 **Multiple Filters** 3 **Multiple Channels**

Filter Reuse in PE

1 **Multiple Fmaps** 2 Multiple Filters 3 Multiple Channels

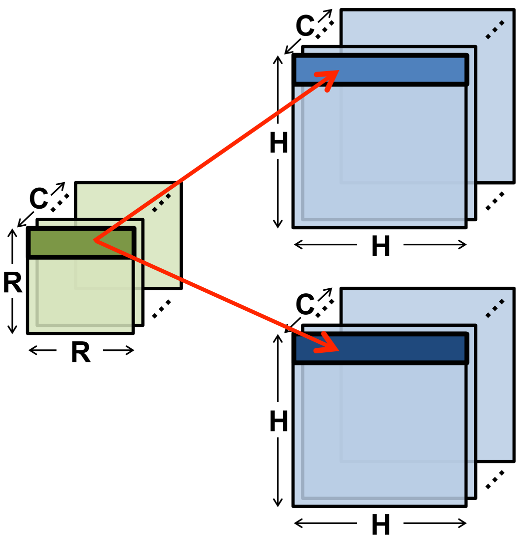


	Filter 1		Fmap 1		Psum 1
Channel 1	Row 1	*	Row 1	=	Row 1

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

Filter Reuse in PE

1 Multiple Fmaps   2 Multiple Filters   3 Multiple Channels



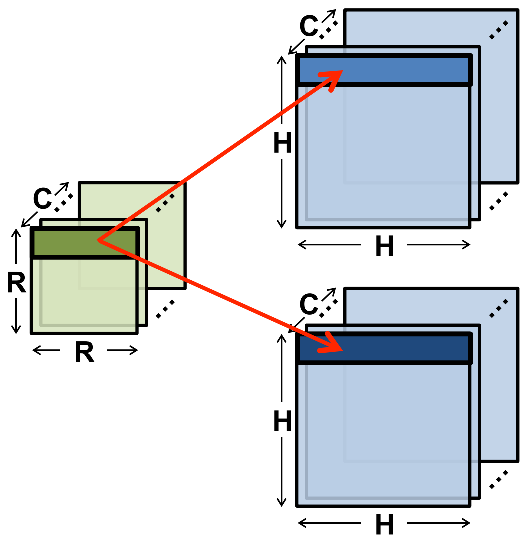
	Filter 1		Fmap 1		Psum 1
Channel 1	Row 1	*	Row 1	=	Row 1

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

share the same filter row

Filter Reuse in PE

1 Multiple Fmaps 2 Multiple Filters 3 Multiple Channels



	Filter 1		Fmap 1		Psum 1
Channel 1	Row 1	*	Row 1	=	Row 1

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

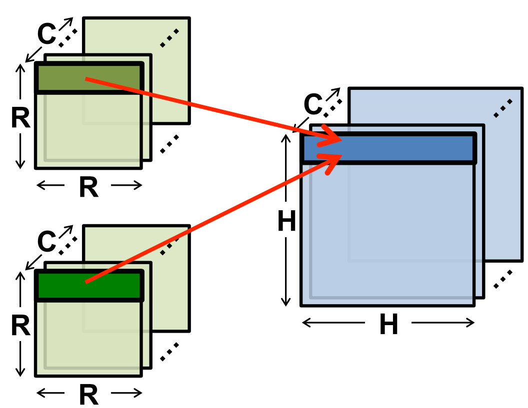
share the same filter row

<u>Processing in PE</u>: **concatenate fmap rows**

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

Fmap Reuse in PE

1 Multiple Fmaps **2 Multiple Filters** 3 Multiple Channels

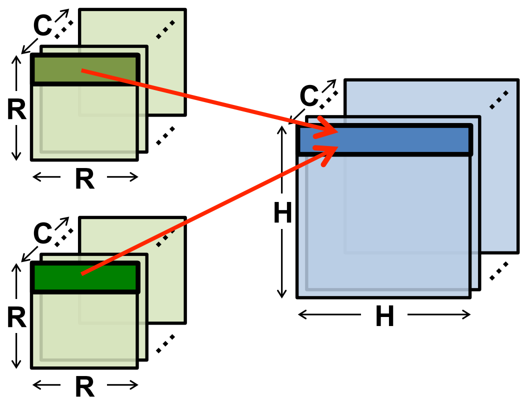


	Filter 1		Fmap 1		Psum 1
Channel 1	Row 1	*	Row 1	=	Row 1

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

Fmap Reuse in PE

1 Multiple Fmaps　**2 Multiple Filters**　3 Multiple Channels

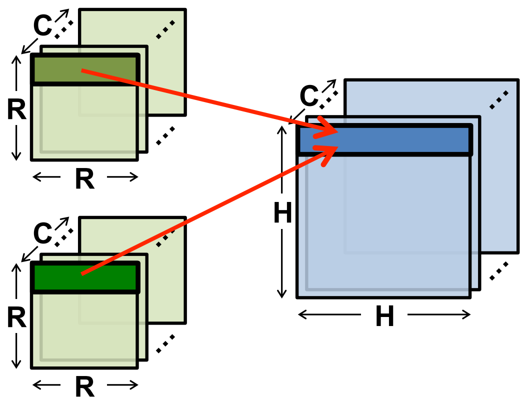


	Filter 1		Fmap 1		Psum 1
Channel 1	Row 1	*	Row 1	=	Row 1
	Filter 2		**Fmap 1**		**Psum 2**
Channel 1	Row 1	*	Row 1	=	Row 1

share the same fmap row

Fmap Reuse in PE

1 Multiple Fmaps **2 Multiple Filters** 3 Multiple Channels



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

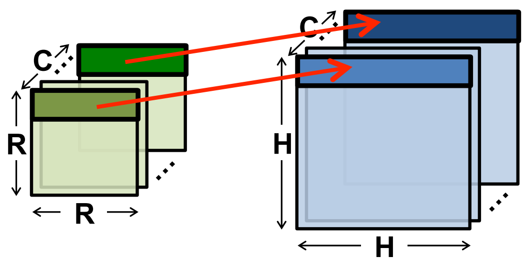
share the same fmap row

Processing in PE: **interleave filter rows**

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

Channel Accumulation in PE

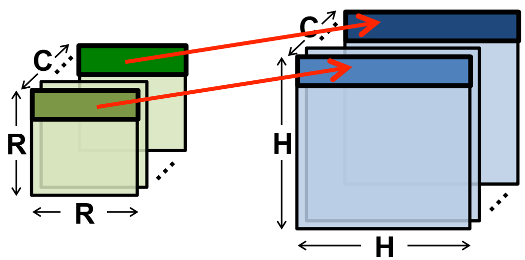
1 Multiple Fmaps 2 Multiple Filters **3 Multiple Channels**



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

Channel Accumulation in PE

1 Multiple Fmaps   2 Multiple Filters   **3 Multiple Channels**



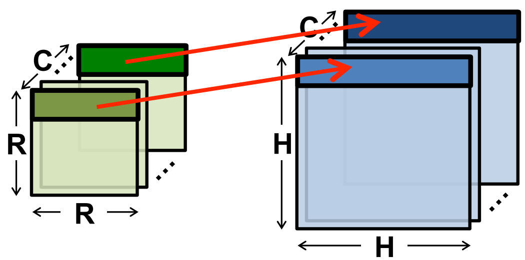
	Filter 1		Fmap 1		Psum 1
Channel 1	Row 1	*	Row 1	=	Row 1
	Filter 1		**Fmap 1**		**Psum 1**
Channel 2	Row 1	*	Row 1	=	Row 1

accumulate psums

Row 1 + Row 1 = Row 1

Channel Accumulation in PE

① Multiple Fmaps ② Multiple Filters **③ Multiple Channels**



	Filter 1		Fmap 1		Psum 1
Channel 1	Row 1	*	Row 1	=	Row 1

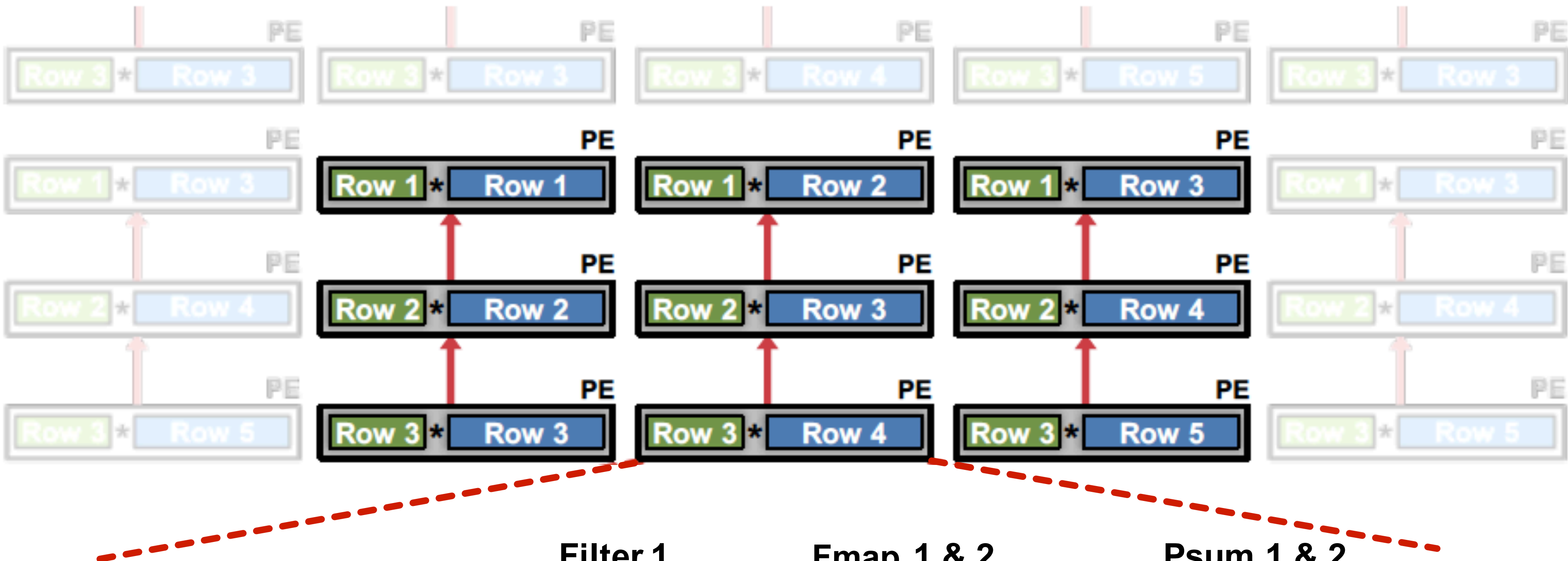
	Filter 1		Fmap 1		Psum 1
Channel 2	Row 1	*	Row 1	=	Row 1

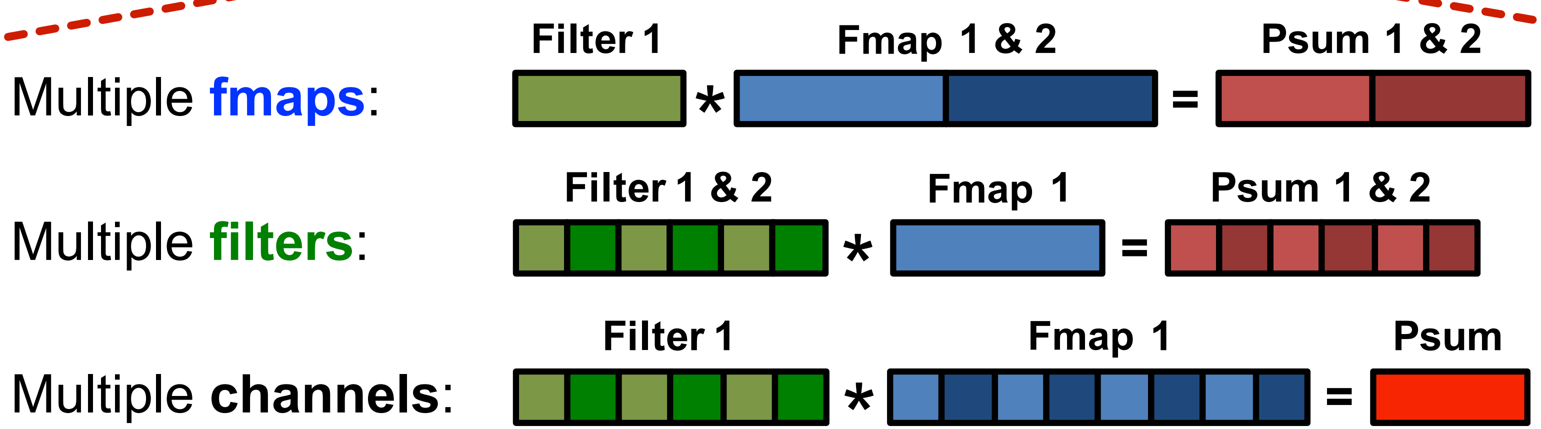
accumulate psums

Processing in PE: **interleave channels**

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

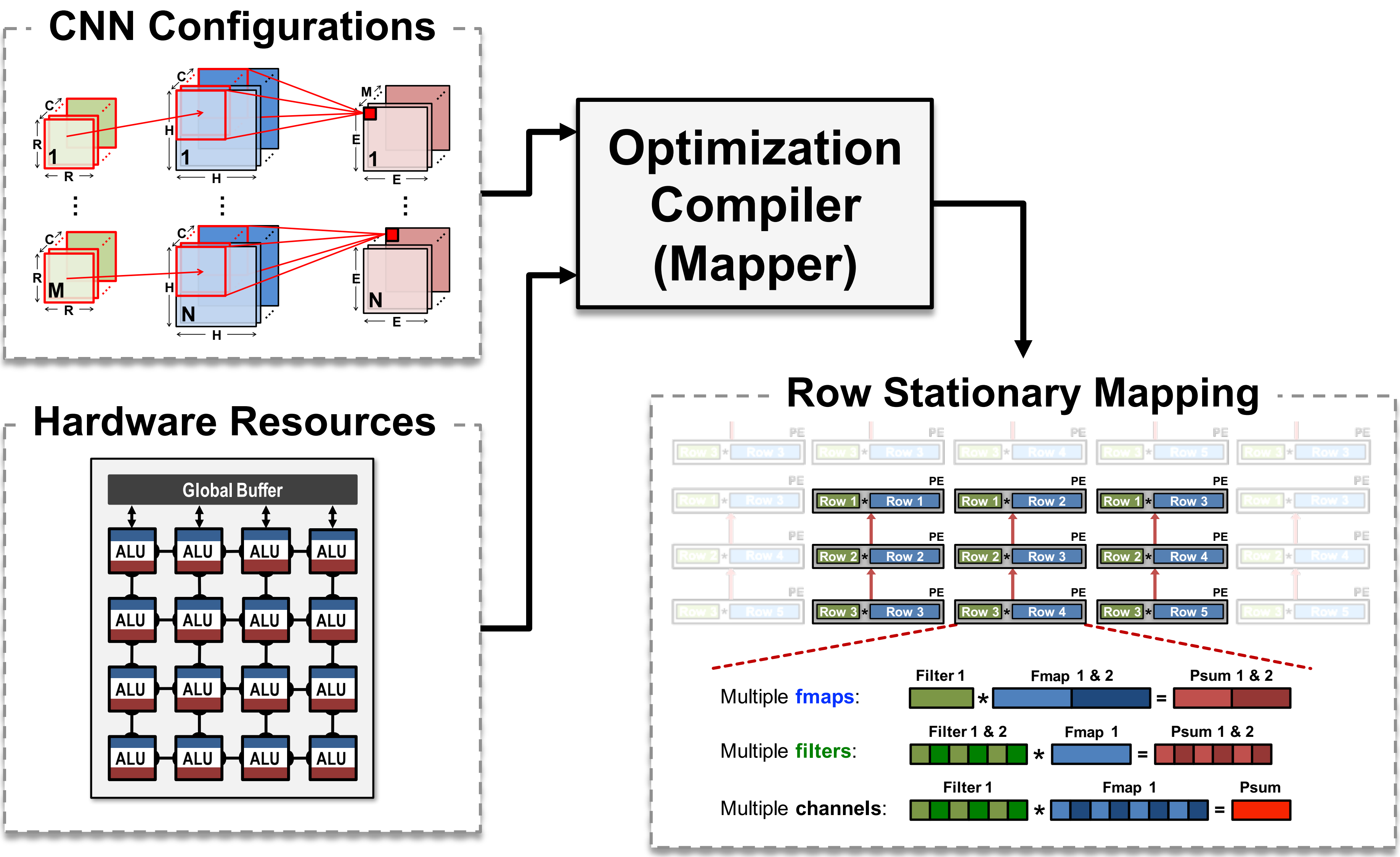
DNN Processing – The Full Picture





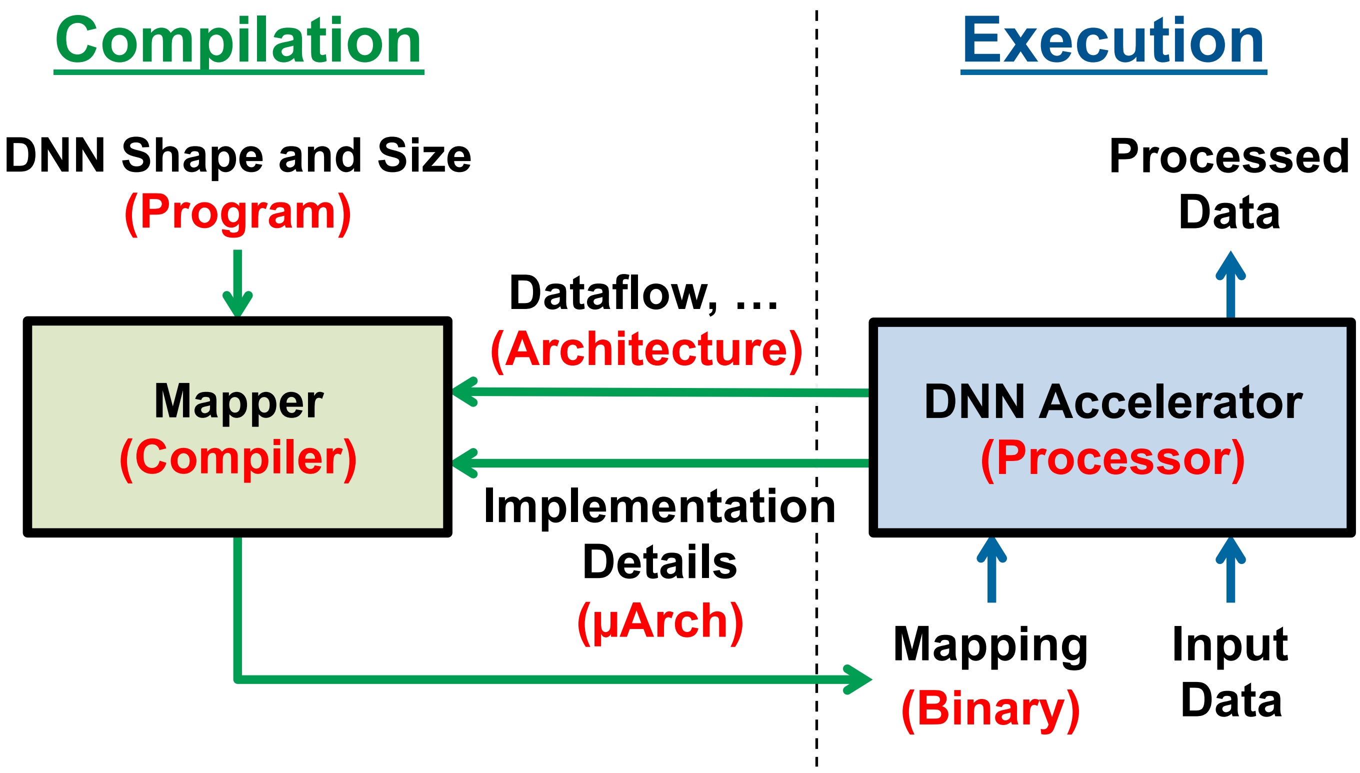
Map rows from **multiple fmaps**, **filters** and **channels** to same PE to exploit other forms of reuse and local accumulation

Optimal Mapping in Row Stationary



[Chen et al., ISCA 2016]

Computer Architecture Analogy



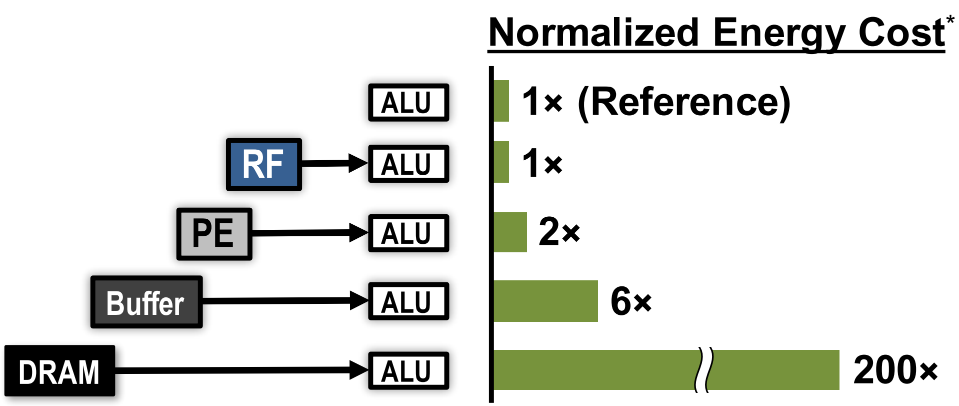
[Chen et al., Micro Top-Picks 2017]

Dataflow Simulation Results

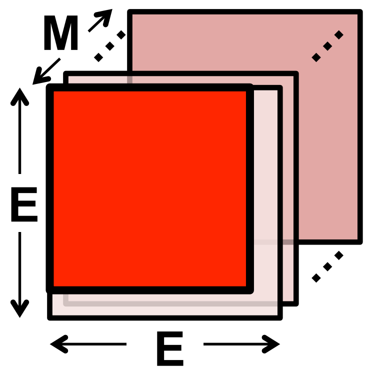
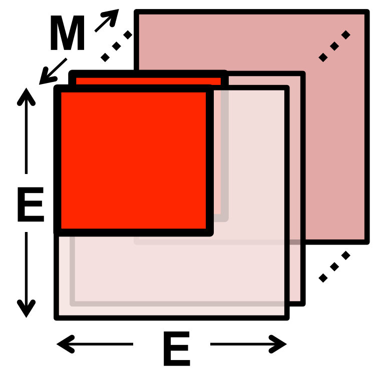
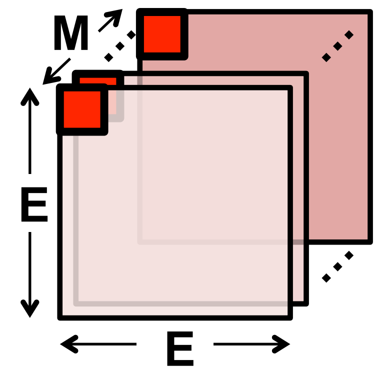
Evaluate Reuse in Different Dataflows

- **Weight Stationary**
  - Minimize movement of filter weights
- **Output Stationary**
  - Minimize movement of partial sums
- **No Local Reuse**
  - No PE local storage. Maximize global buffer size.
- **Row Stationary**

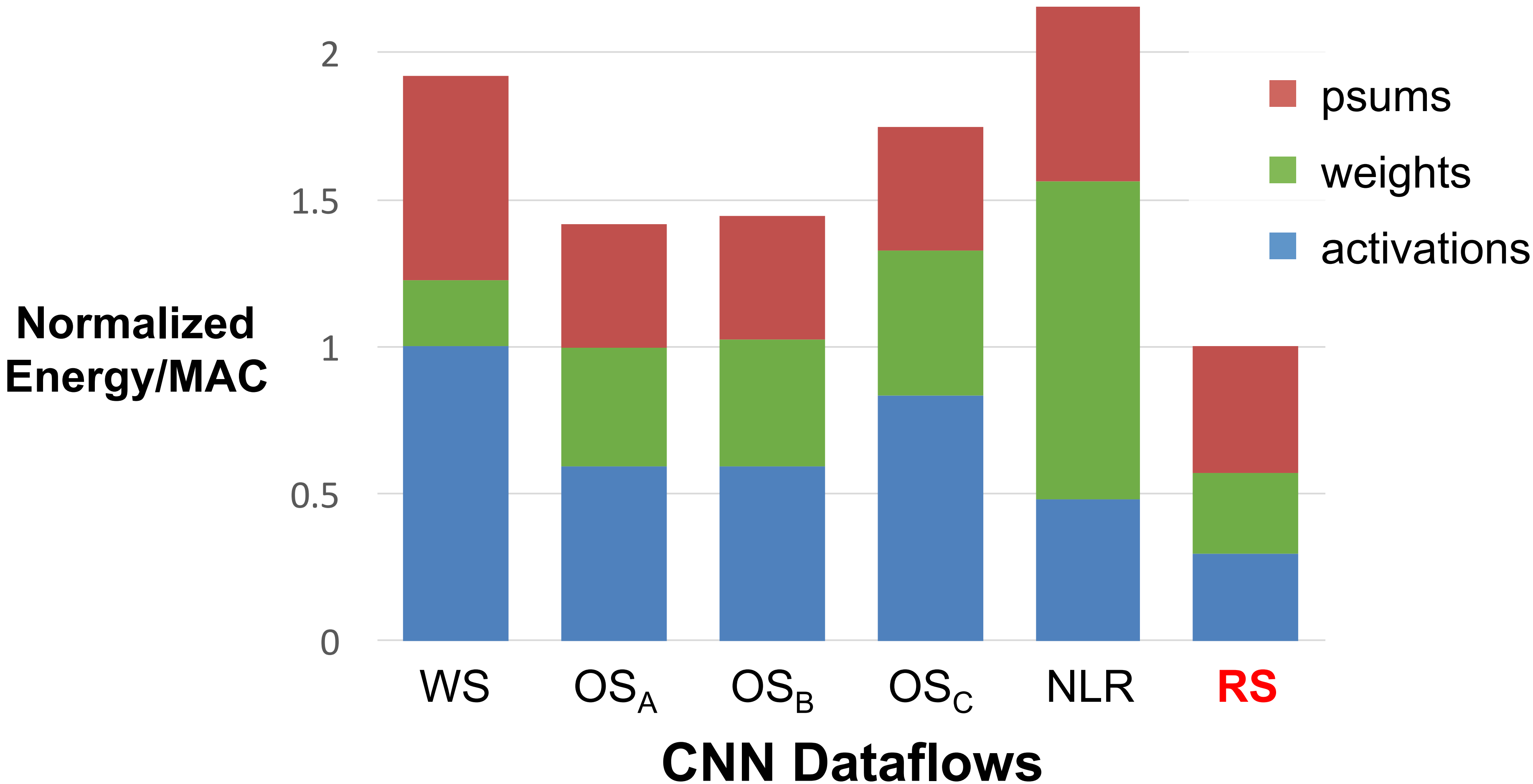
Evaluation Setup
- same total area
- 256 PEs
- AlexNet
- batch size = 16



Variants of Output Stationary

	OS_A	OS_B	OS_C
Parallel Output Region			
# Output Channels	Single	Multiple	Multiple
# Output Activations	Multiple	Multiple	Single
Notes	Targeting **CONV** layers		Targeting **FC** layers

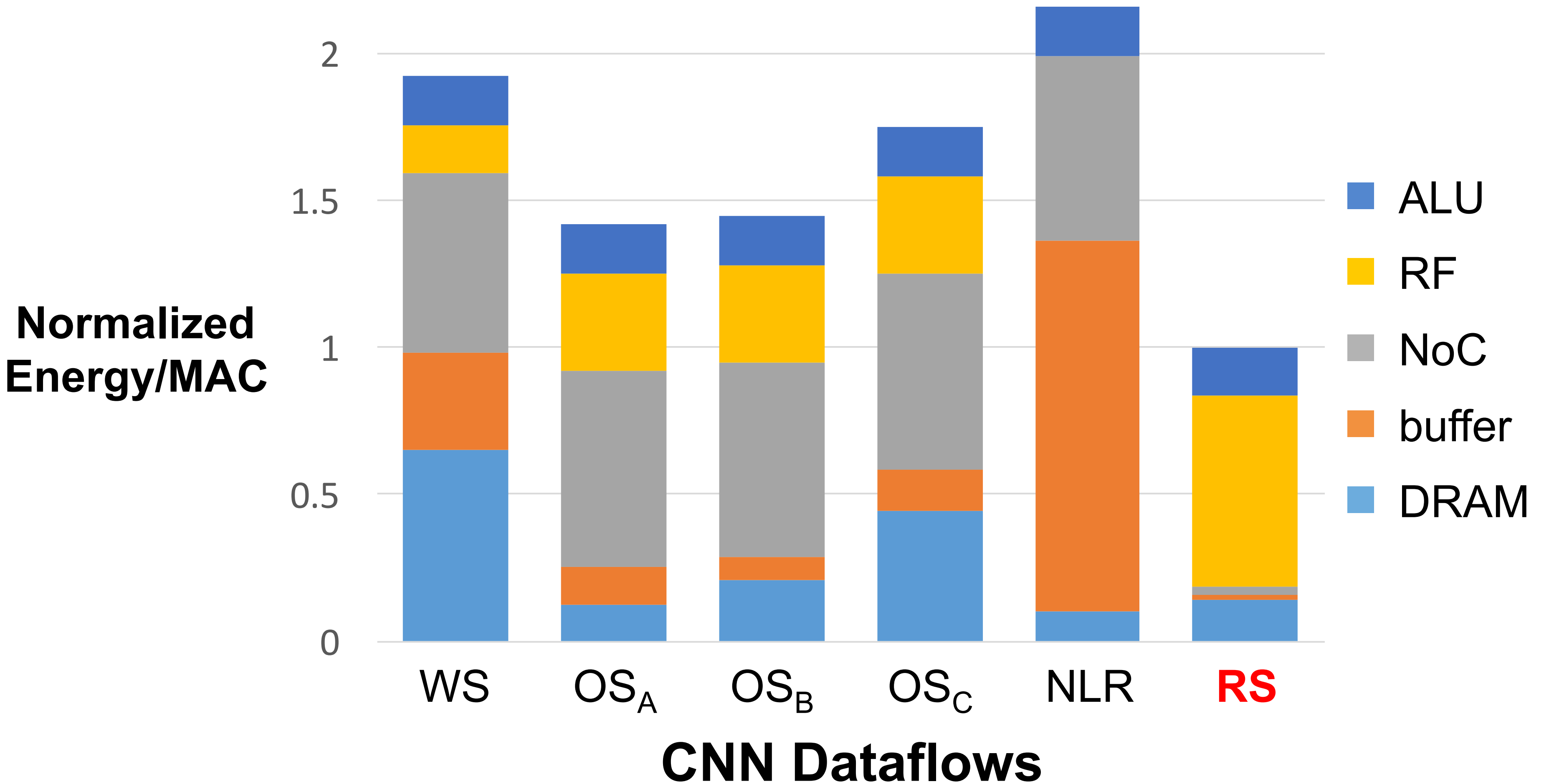
Dataflow Comparison: CONV Layers



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

[Chen et al., ISCA 2016]

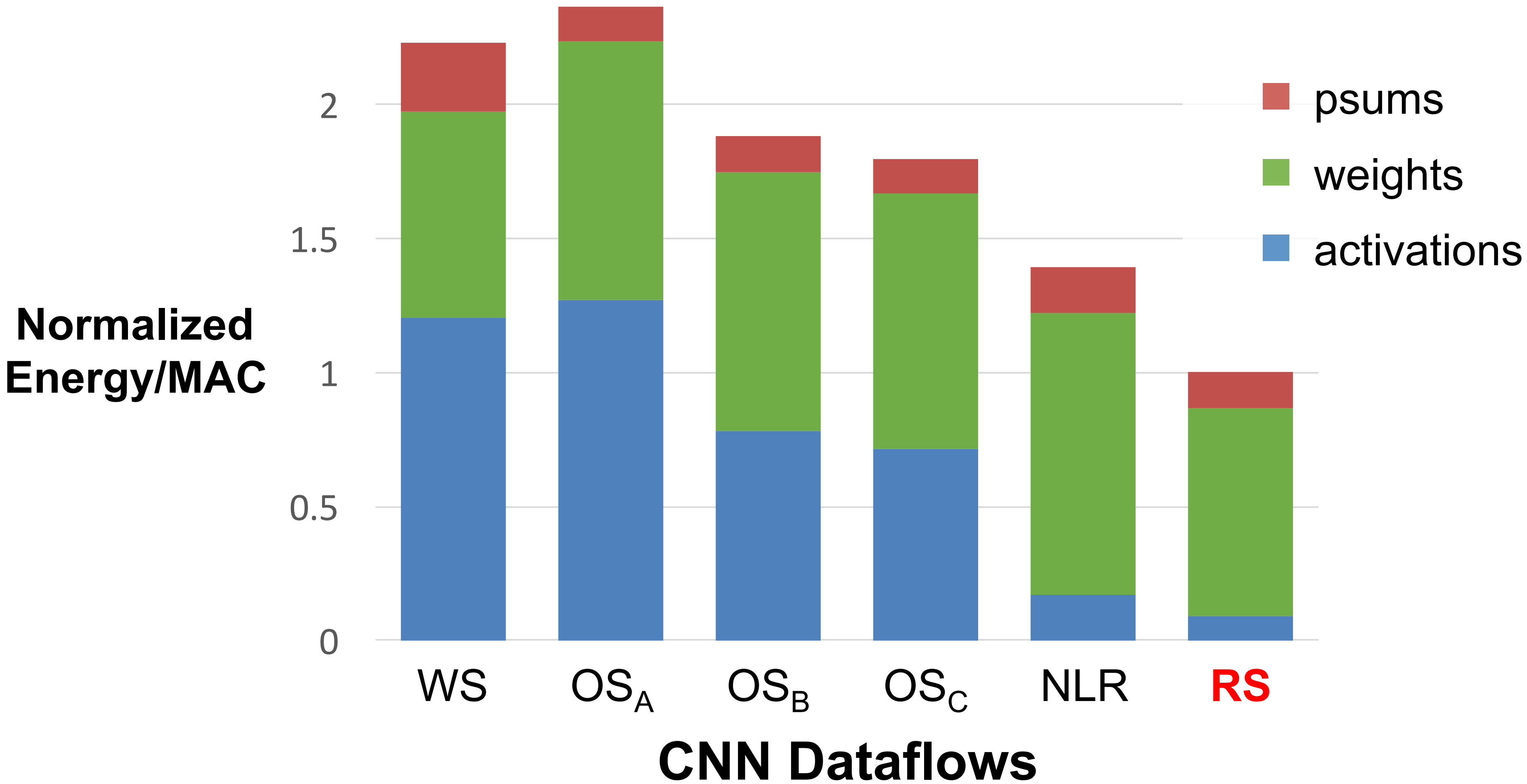
Dataflow Comparison: CONV Layers



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

[Chen et al., ISCA 2016]

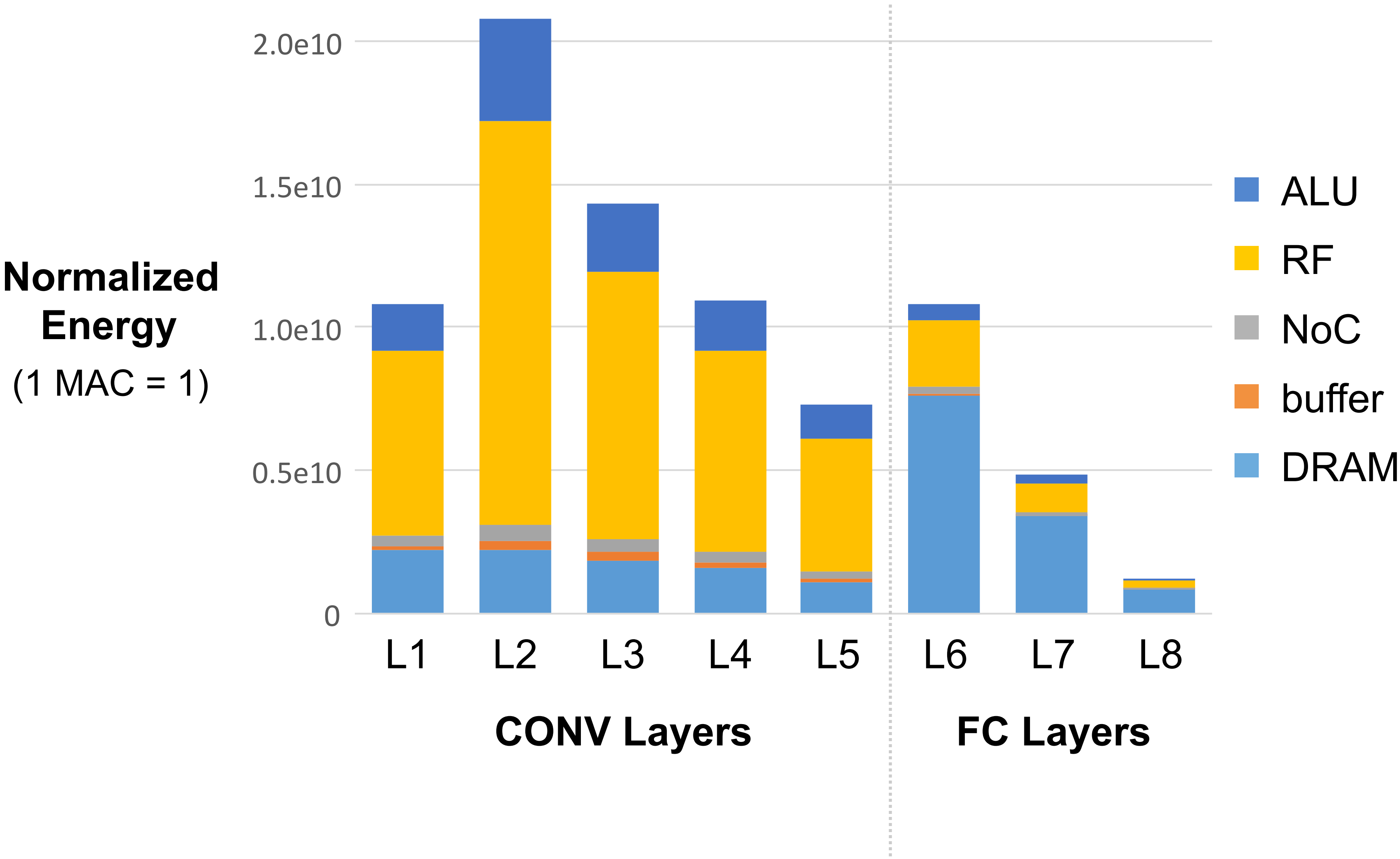
Dataflow Comparison: FC Layers



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

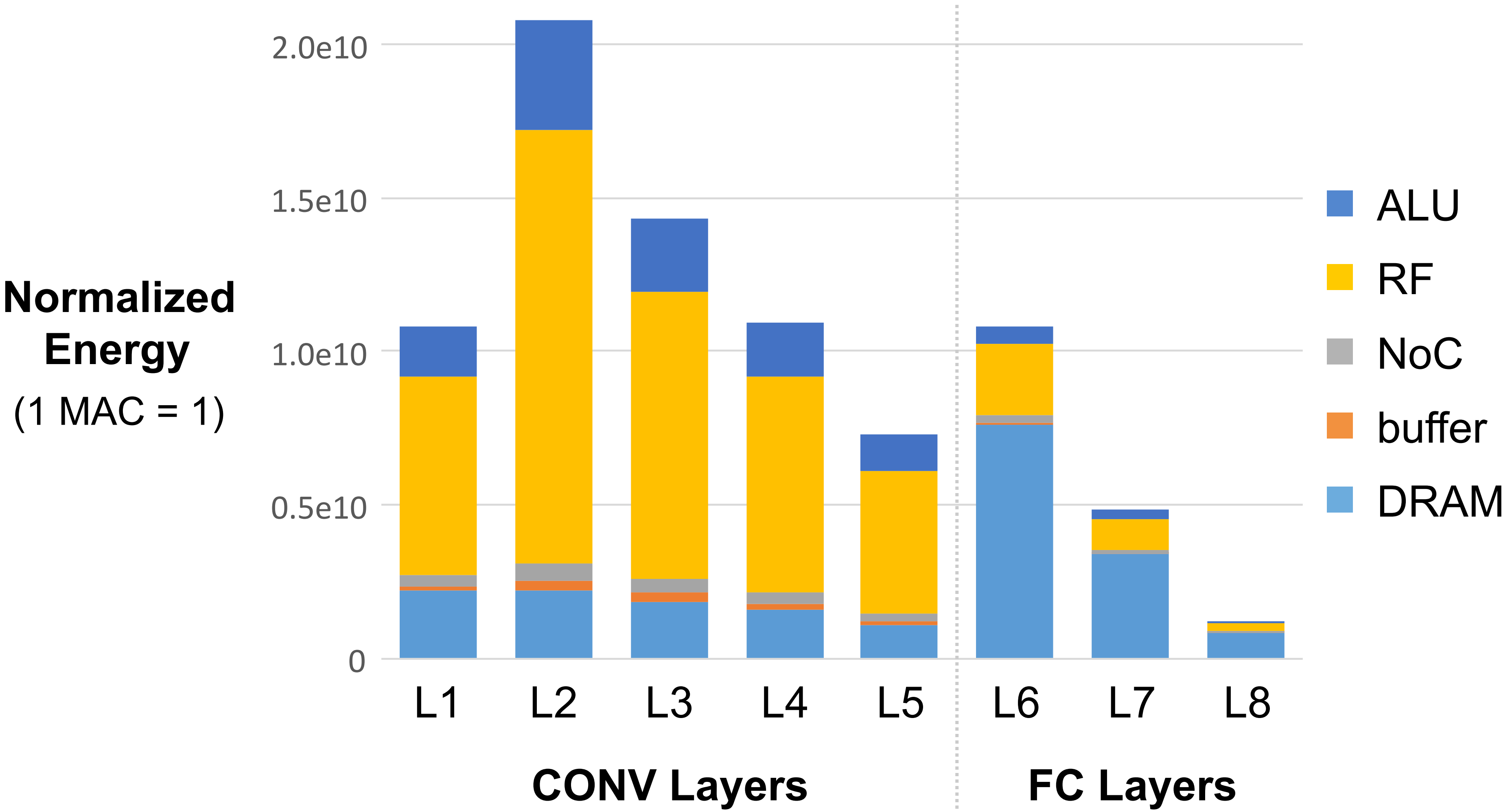
[Chen et al., ISCA 2016]

Row Stationary: Layer Breakdown



[Chen et al., ISCA 2016]

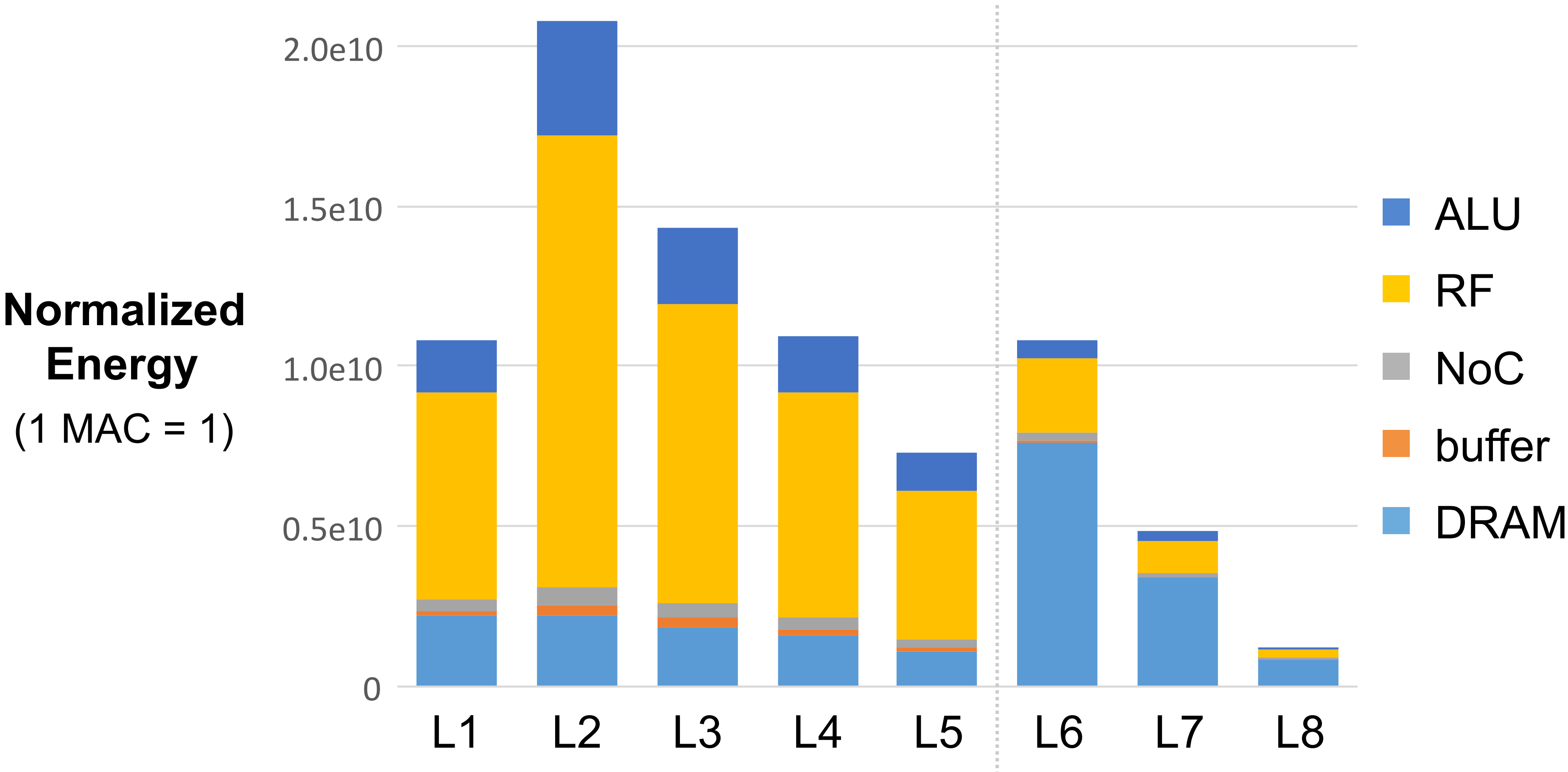
Row Stationary: Layer Breakdown



RF dominates

[Chen et al., ISCA 2016]

Row Stationary: Layer Breakdown

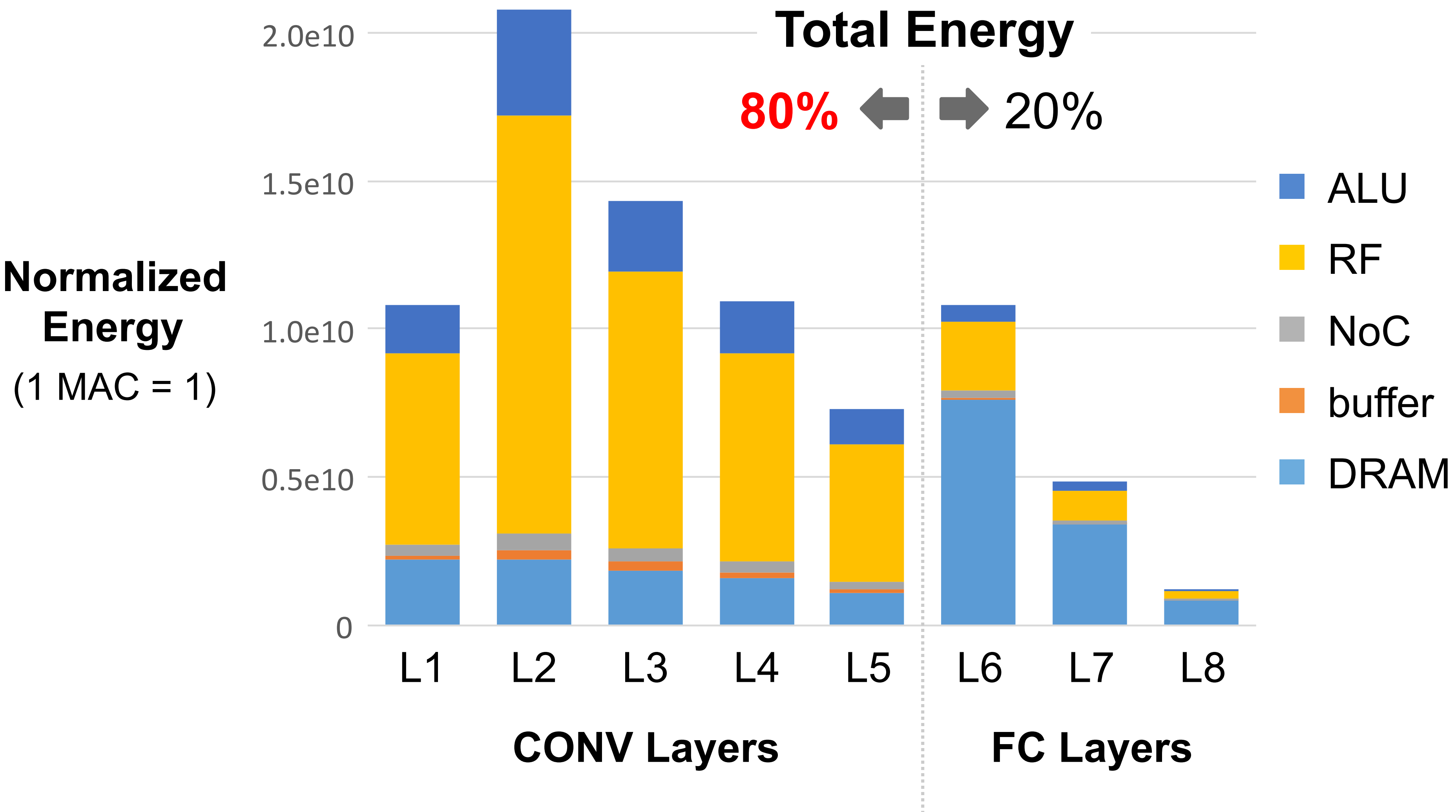


RF dominates

DRAM dominates

[Chen et al., ISCA 2016]

Row Stationary: Layer Breakdown

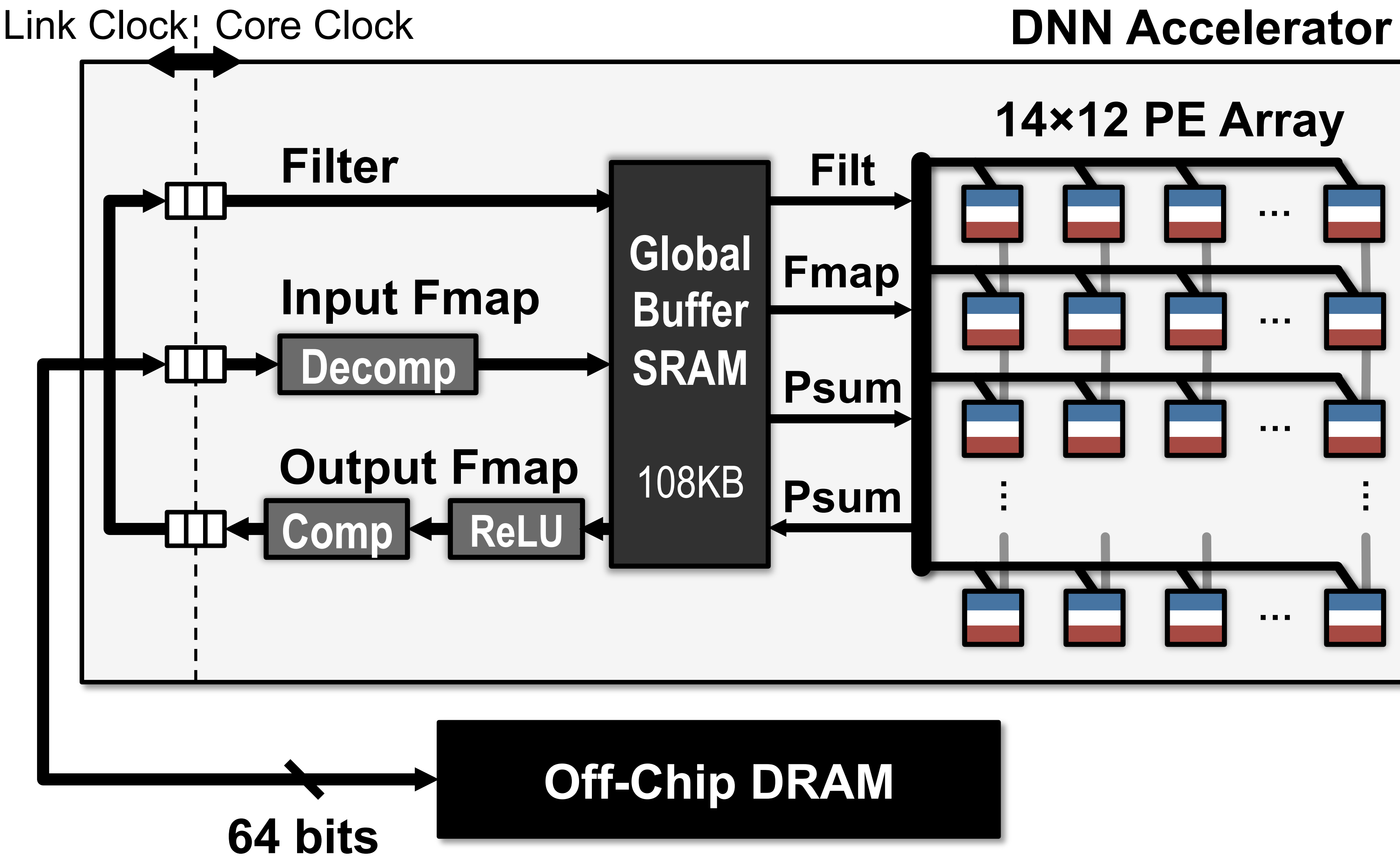


CONV layers dominate energy consumption!

Hardware Architecture for RS Dataflow

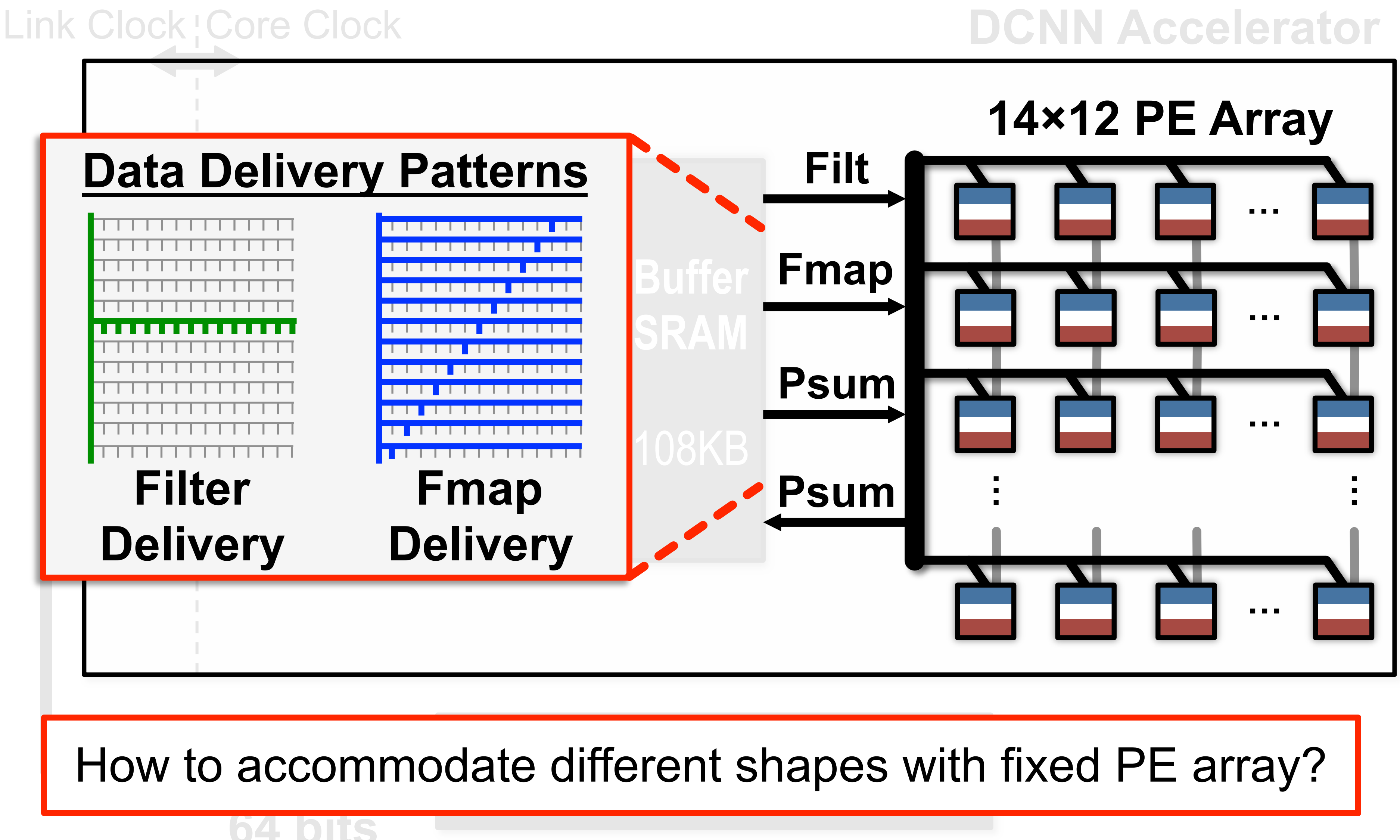
[Chen et al., ISSCC 2016]

Eyeriss DNN Accelerator



[Chen et al., ISSCC 2016]

Data Delivery with On-Chip Network

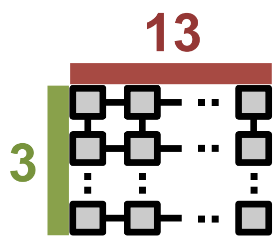


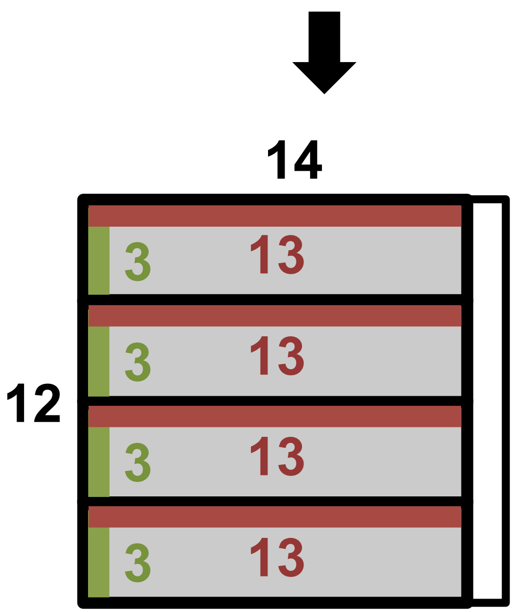
How to accommodate different shapes with fixed PE array?

Logical to Physical Mappings

Replication

AlexNet
Layer 3-5

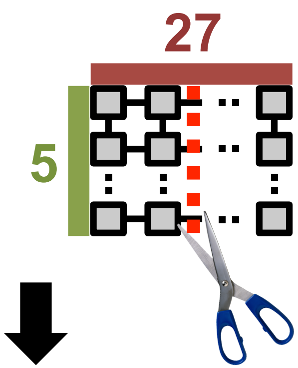


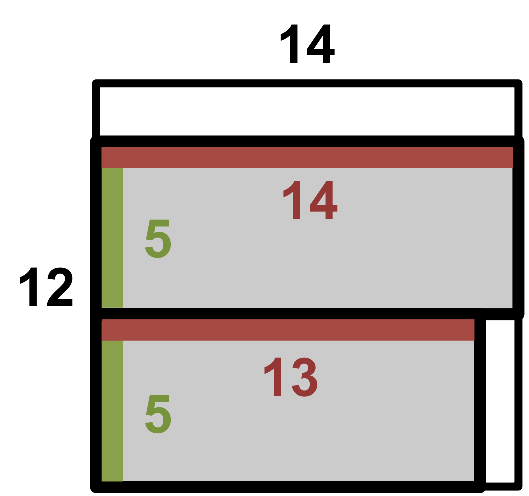


Physical PE Array

Folding

AlexNet
Layer 2

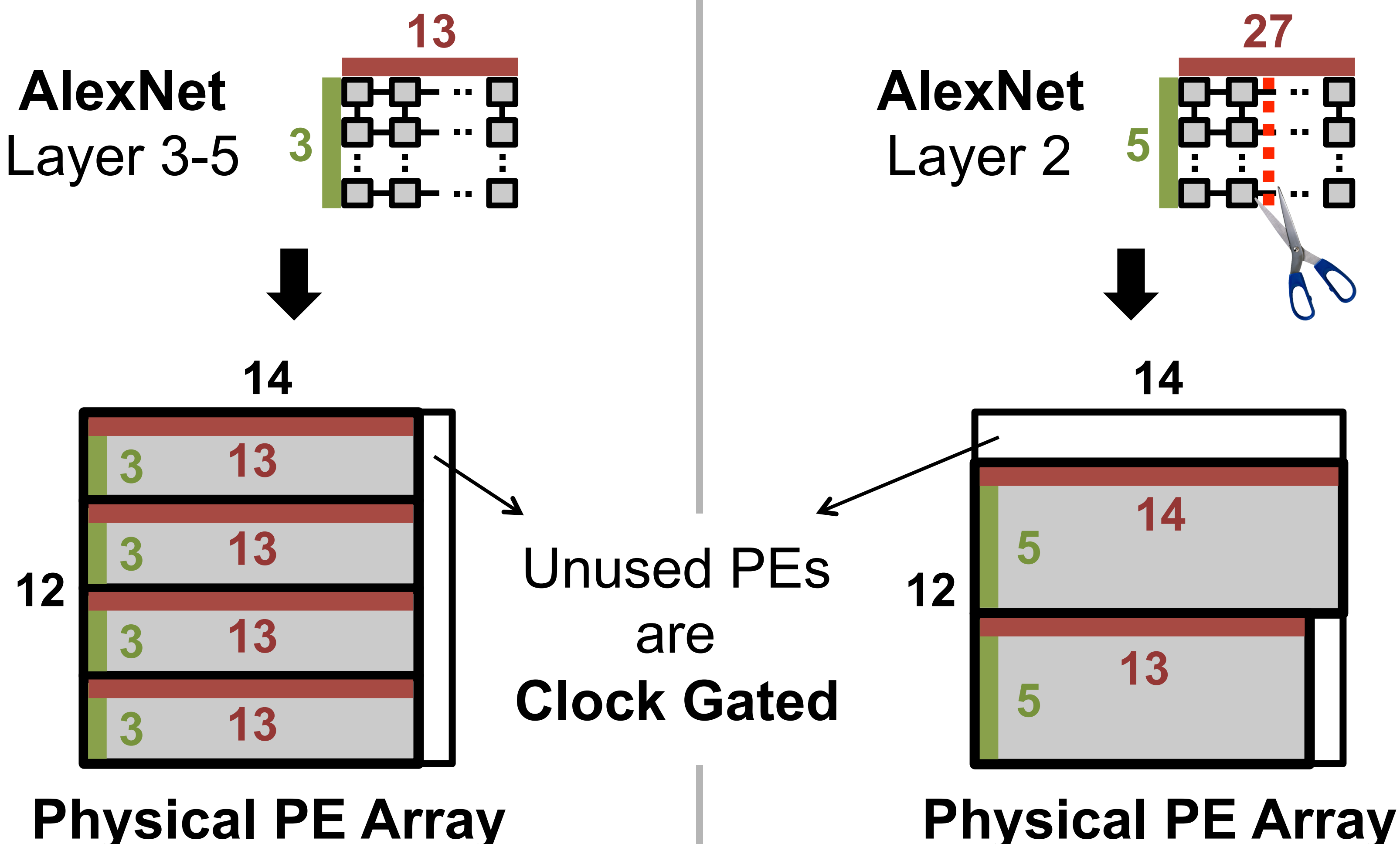




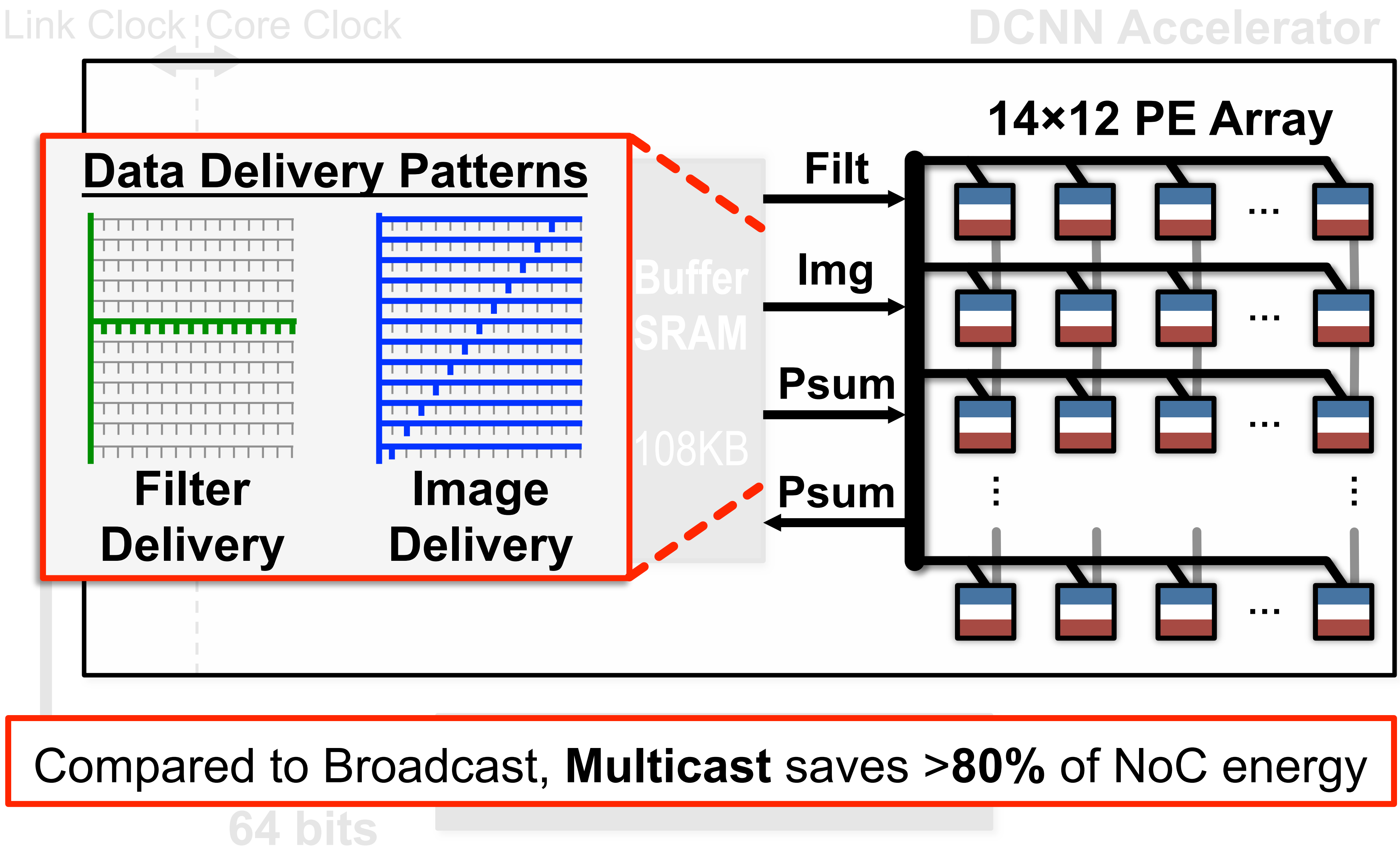
Physical PE Array

Logical to Physical Mappings

Replication | Folding



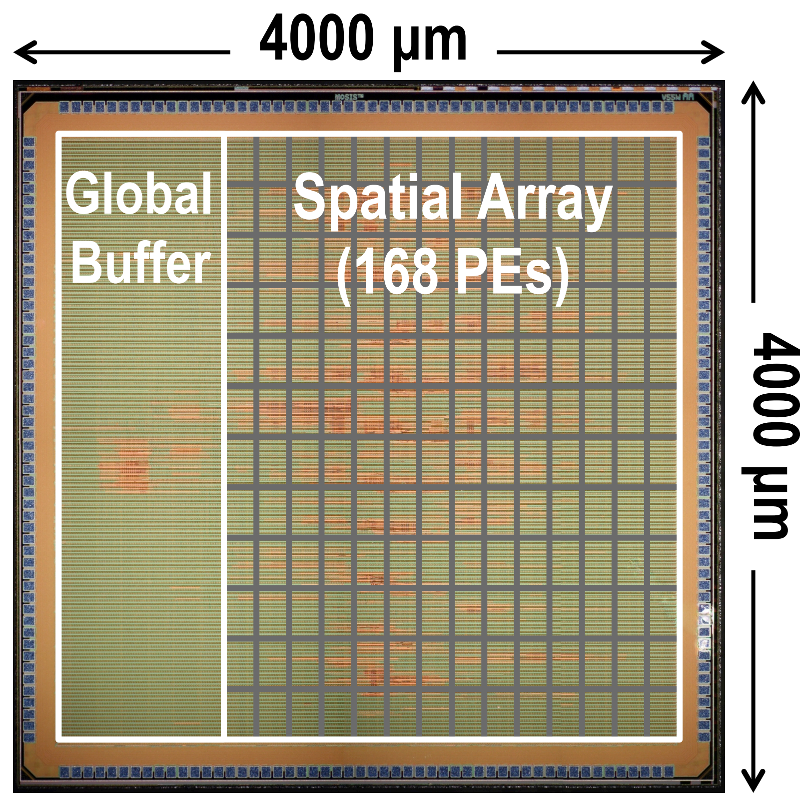
Data Delivery with On-Chip Network



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)

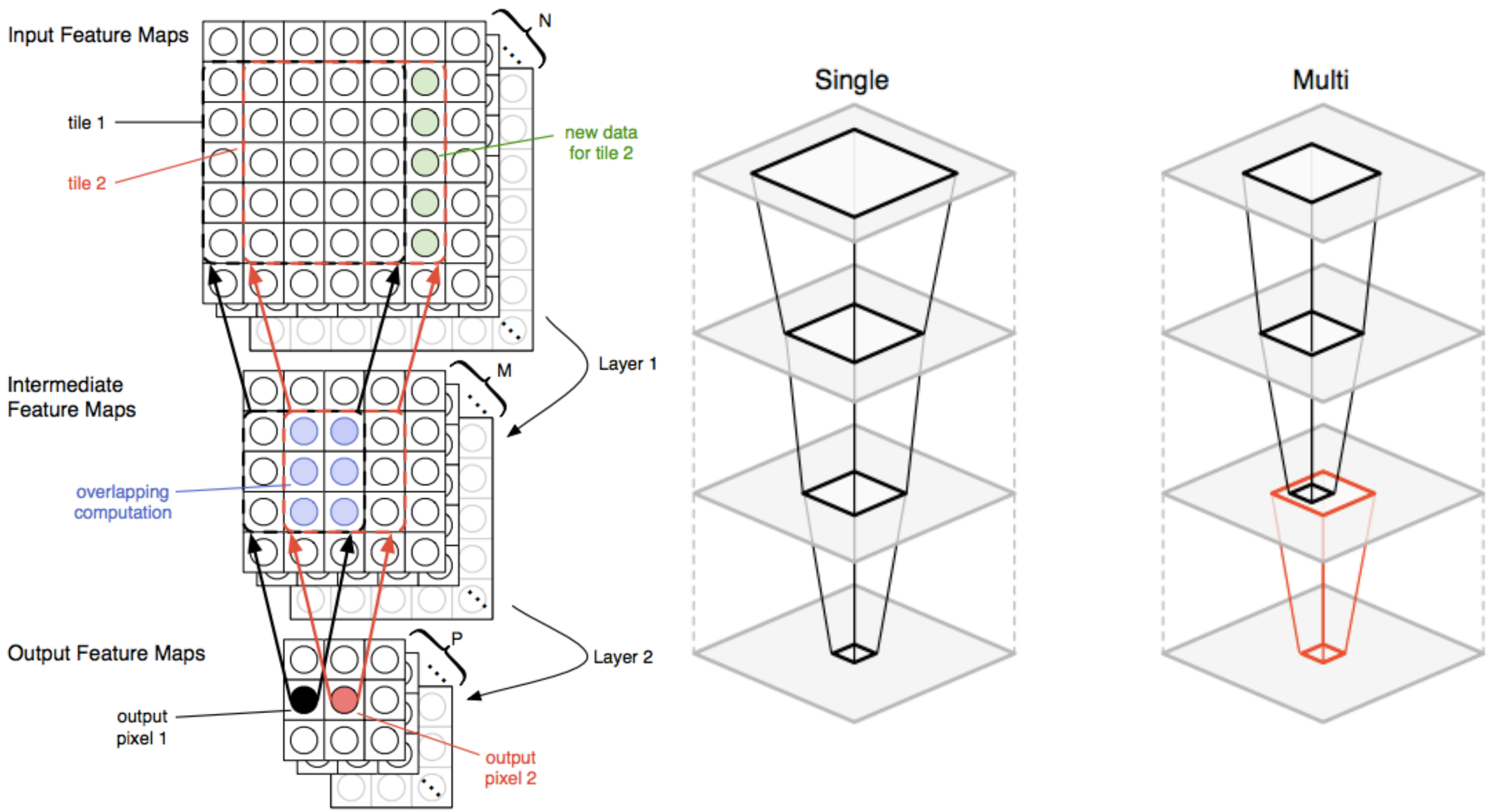
[Chen et al., ISSCC 2016]

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**

Fused Layer

- **Dataflow across multiple layers**



[Alwani et al., MICRO 2016]