

The background of the slide features abstract, overlapping geometric shapes in various shades of blue, ranging from light sky blue to deep navy blue. These shapes are primarily located on the left and right sides, framing the central white area where the text is placed.

Lecture

Domain Specific Computing CASE Study: Eyeriss

Chung-Ho Chen
NCKU EE

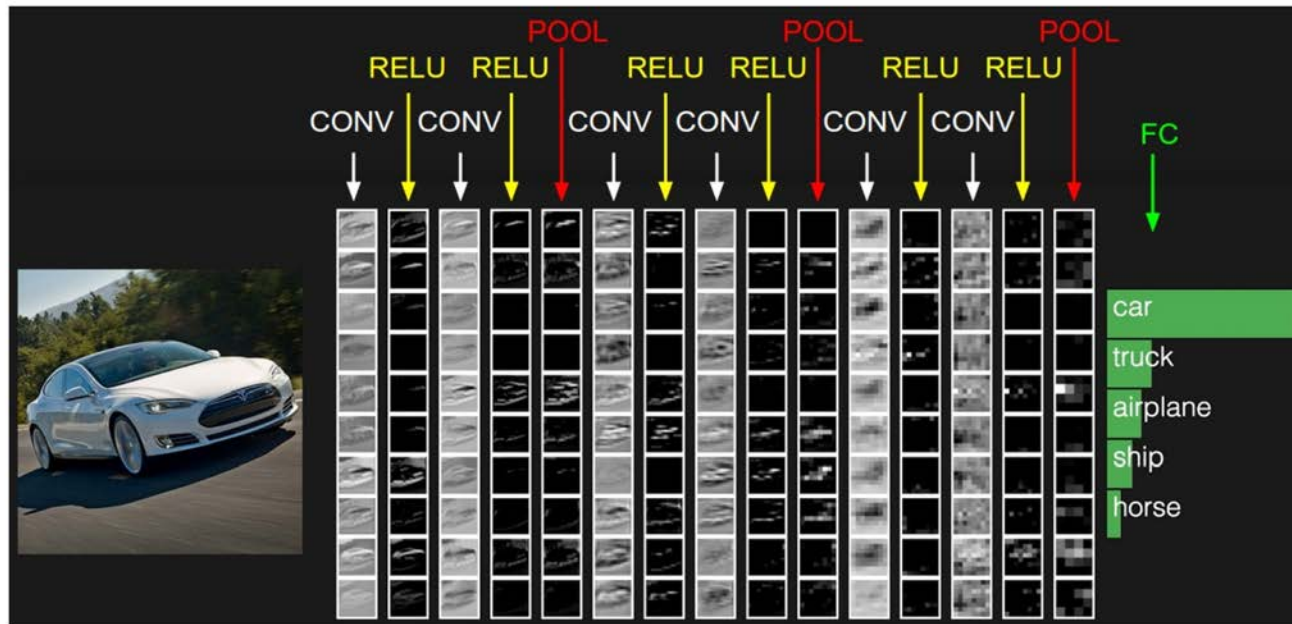
Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks

- ▶ A reconfigurable CNN processor
- ▶ Yu-Hsin Chen¹, Joel Emer^{1, 2}, Vivienne Sze¹
 - ▶ ¹ MIT ² NVIDIA
- ▶ 35 fps @ 278 mw, Alexnet CONV layers, ISSCC, 2016
- ▶ [Design for Highly Flexible and Energy-Efficient Deep Neural Network Accelerators \[Yu-Hsin Chen\]](#)
 - ▶ https://www.youtube.com/watch?v=brhOo-_7NS4

Deep Convolutional Neural Networks

- ▶ Deep CNN: up to 1000 CONV layers, 1-3 FC layers

<https://cs231n.github.io/convolutional-networks/>



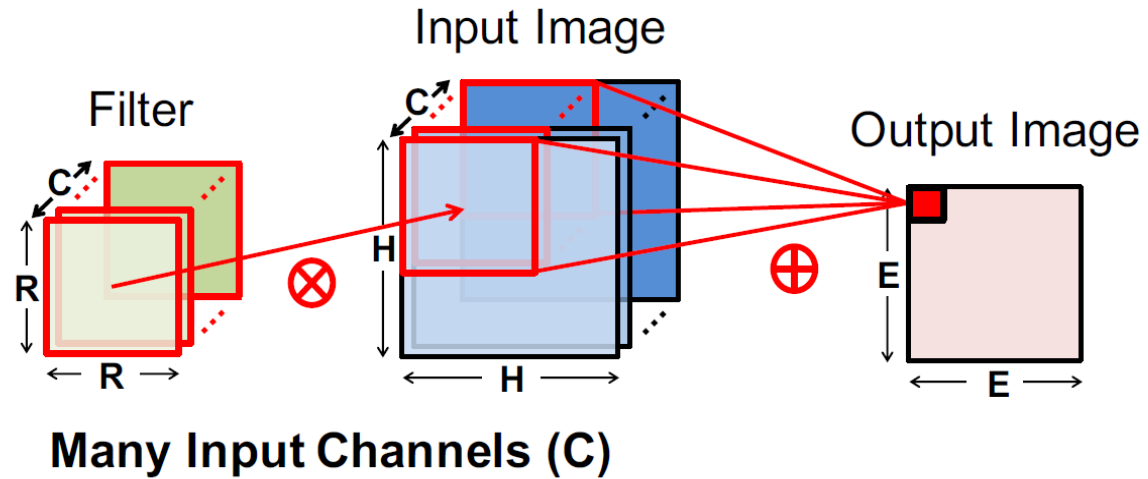
Low-level features

High-level features

Focus on Volume Convolution Computing

- ▶ Many input channels: C

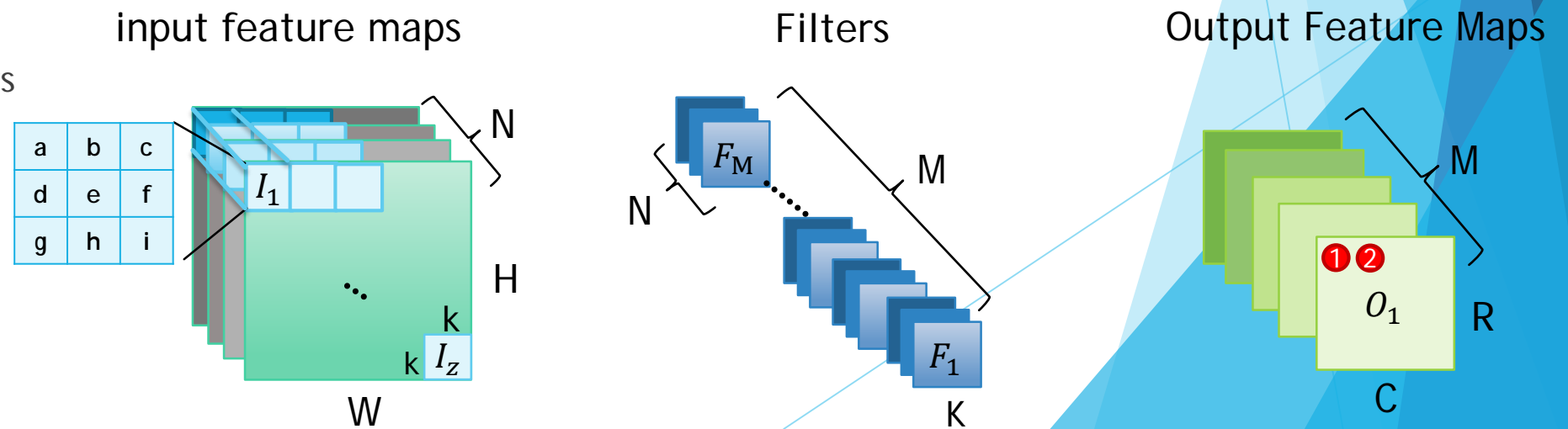
- ▶ One filter



- ▶ Many output channels: M

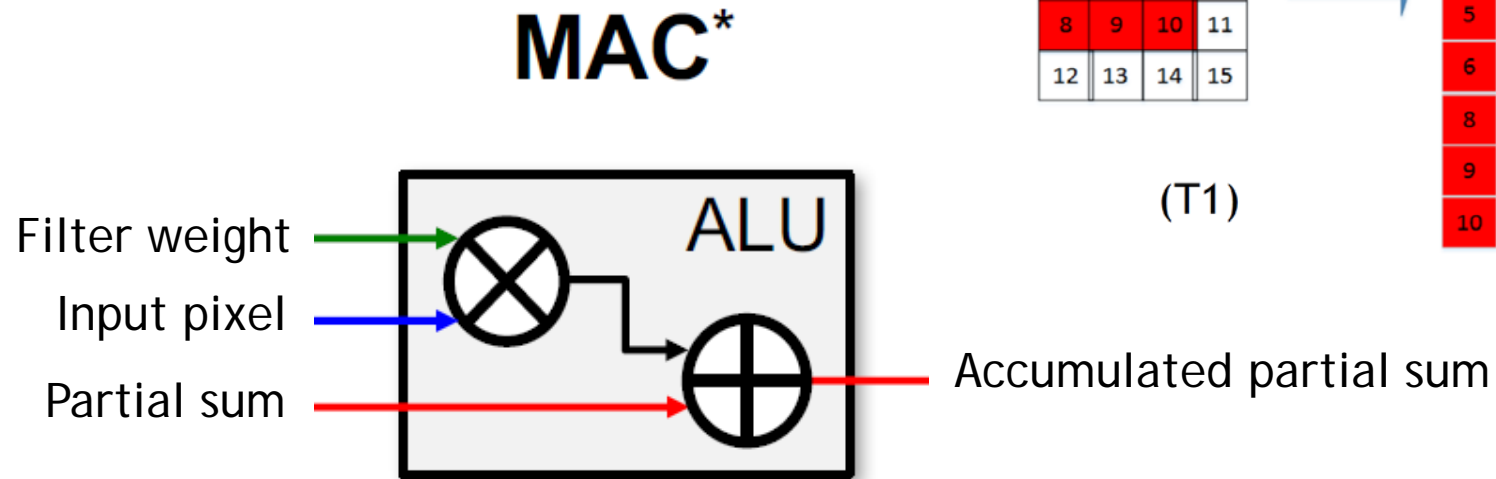
- ▶ Multiple filters

- ▶ M is the # of filters



Multiply-and-Accumulate

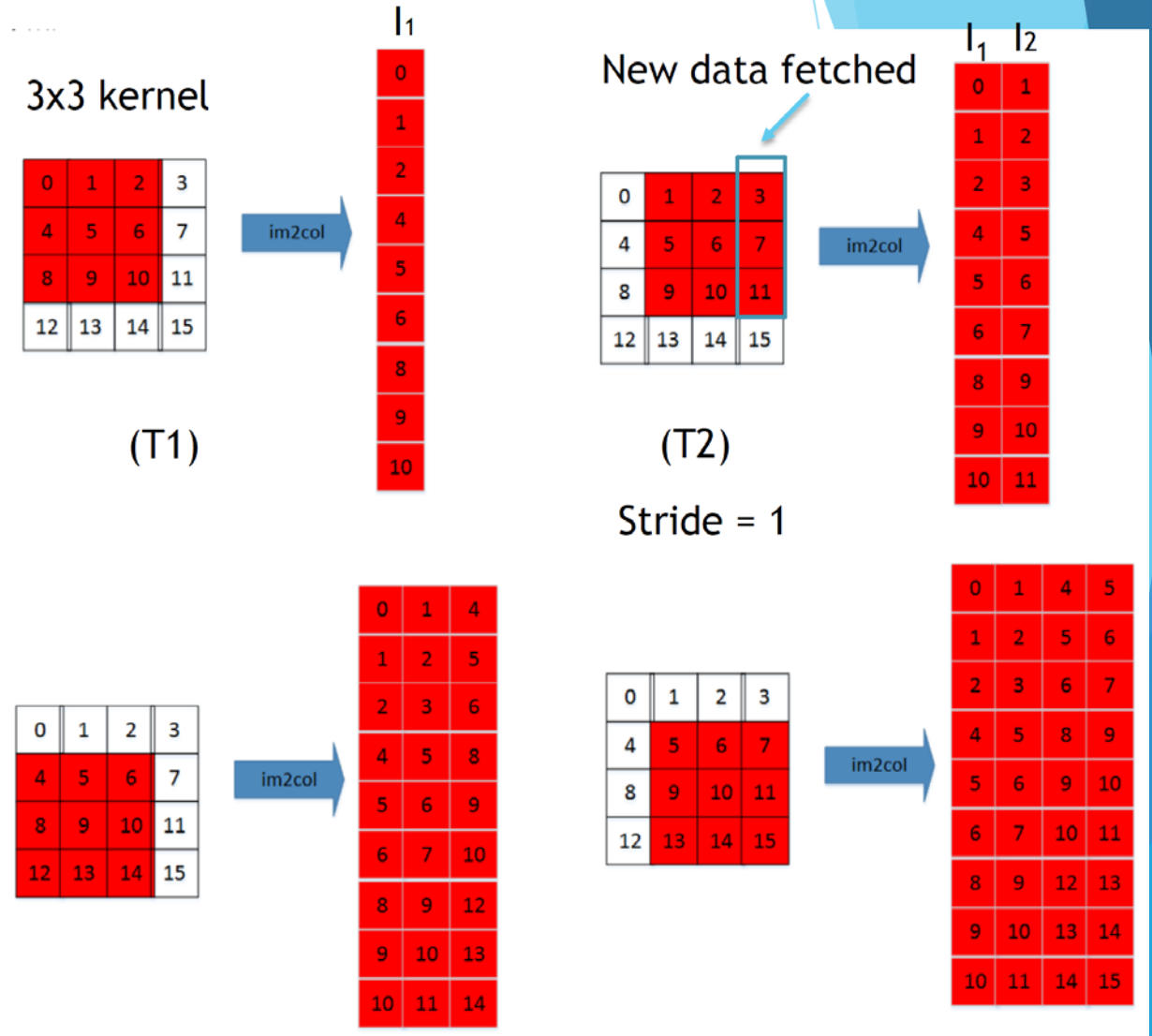
- ▶ Element-wise operation



Accumulate
partial sum from
cell 0, 1, 2, 4....
to 8, 9, 10

Data Reuse

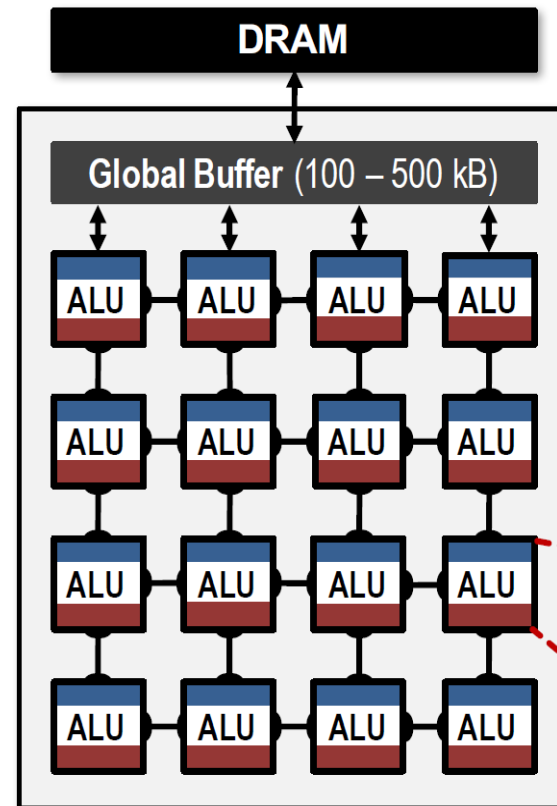
- ▶ Reuse filter weights
 - ▶ the same kernel is used while sliding the convolution block over the input
- ▶ Reuse input image pixels
 - ▶ Some of the same image pixels are reused while sliding the convolution block. E.g., 1, 2, 5, 6, 9, 10 at T2.



Spatial Architecture

► Features

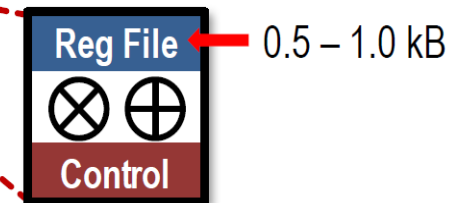
- **Row Stationary (RS)** dataflow model
 - Data flow: computation initiated upon the arrival of data
- Employ 168 PE and four level memory hierarchy: DRAM, global buffer, buffer of neighboring PEs, PE registers
- NoC: features **multicast and point-to-point single cycle data delivery** to support RS dataflow
- Employ **Run-Length Compression (RLC)** and PE gating that exploits data sparsity for energy saving



Local Memory Hierarchy

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

Processing Element (PE)

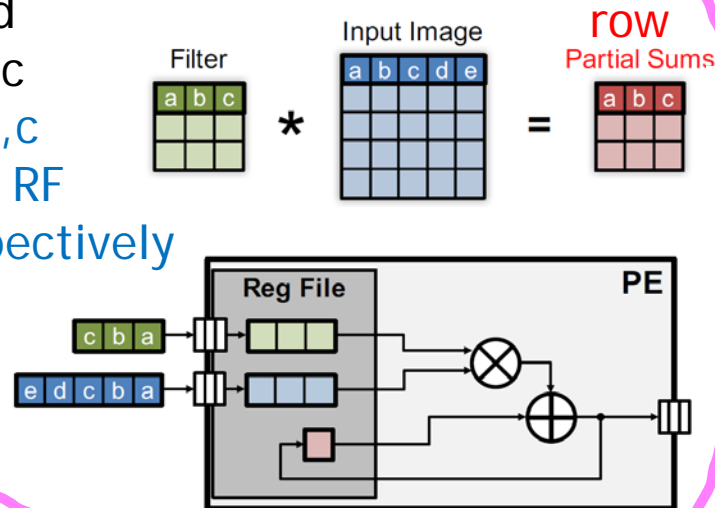


Row Stationary Model: One Row

- ▶ Map 2D convolution to 1D vector multiplication
- ▶ A PE = REG file + one MAC
- ▶ Filter register: right rotate
- ▶ Image register:
 - ▶ First pixel of a convolution block is right shifted
 - ▶ 2nd, 3th pixel rotate right to be used for stride = 1

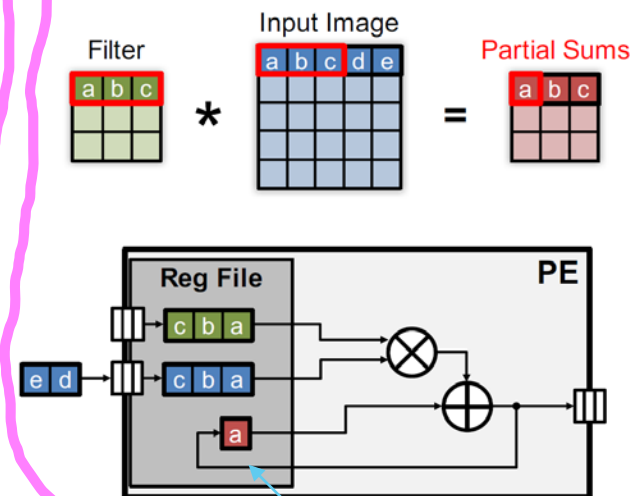
Start in filter row

Load
a, b, c
a, b, c
Into RF
respectively



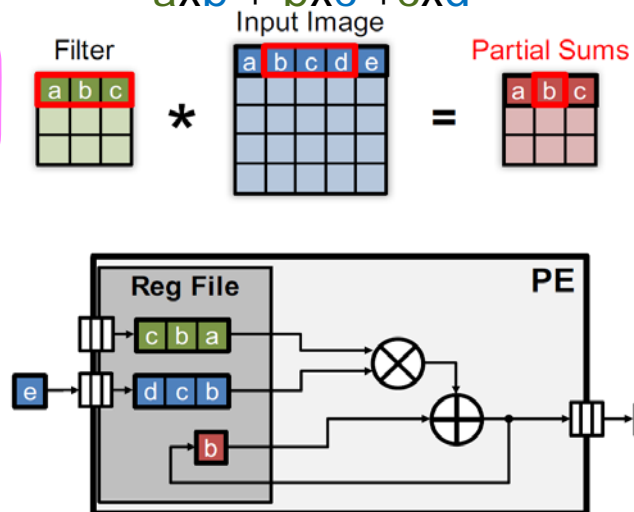
C1-2-3: row partial sum accumulated

$$a = axa + bxb + cxc$$



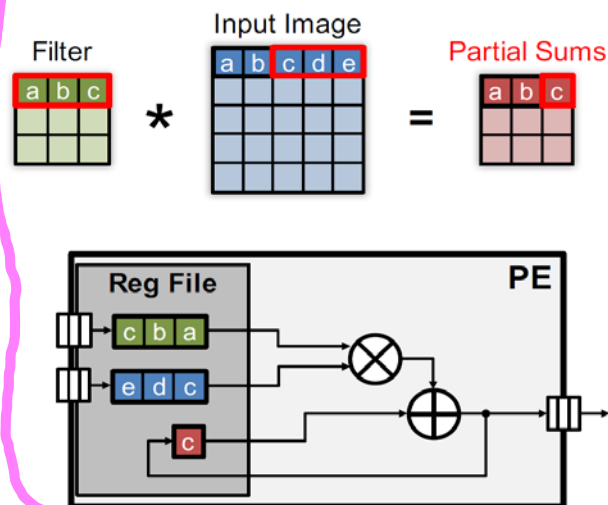
C4-5-6: row partial sum **b** =

$$axb + bxc + cxd$$



C7-8-9: row partial sum **c** =

$$axc + bxd + cx e$$



At C3

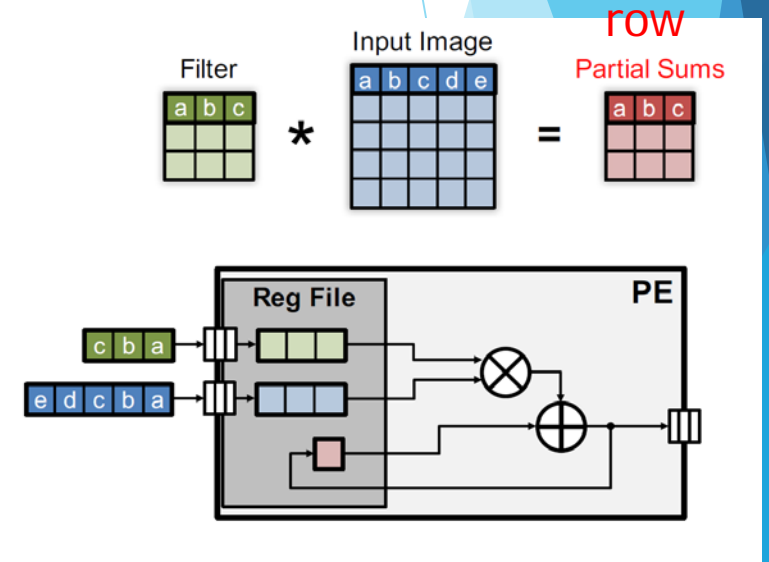
Note: stride = 1

the same filter row is used as window sliding over input and b, c is reused after row partial sum **a** is computed.

Note that the row partial sums here (**a, b, c**) are only for **row1** of the input image for their row convolutions only, not the result of the entire 3x3 block.

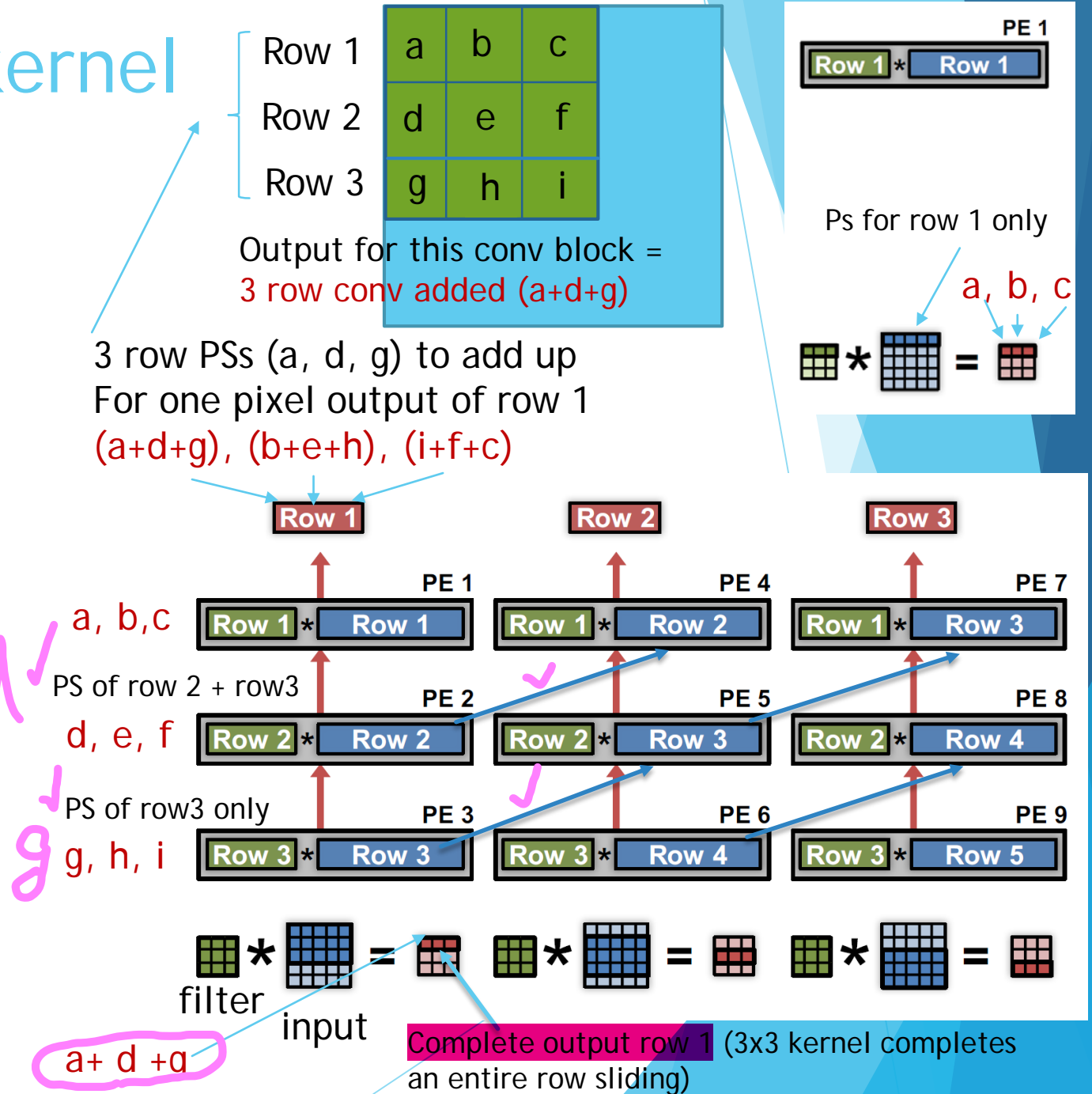
Row Stationary

- ▶ Observation
 - ▶ Row filter weights are reused while moving the convolution block right, striding
 - ▶ Row input pixels are reused while moving the convolution block right
 - ▶ The partial sums computed here are just a **row partial sum, that is, row convolution.**
 - ▶ Register file size: maximum filter row width x pixel size x 2 (for input) + ps result latching
- ▶ Hence row stationary (row data and row filter reused)



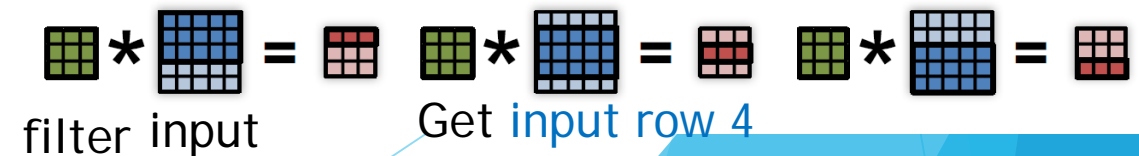
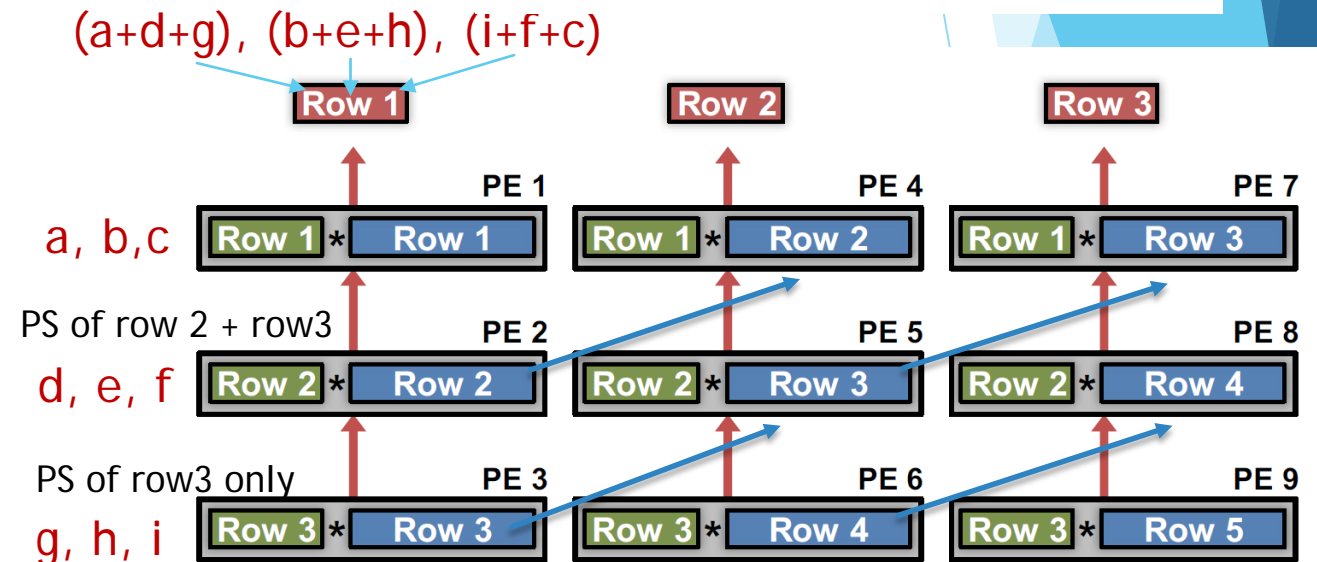
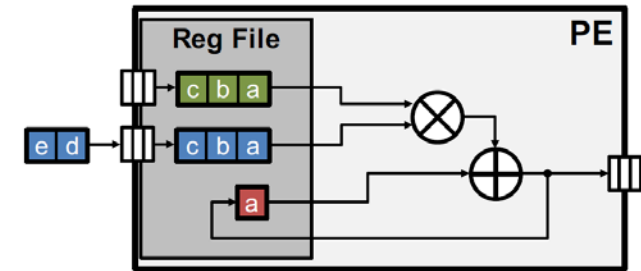
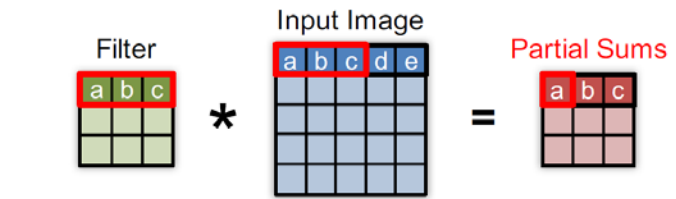
2D Convolution, 3x3 kernel

- ▶ 9 PEs to add up 9 partial sums for an output (3x3 kernels, k x k)
- ▶ A pixel value in an output row is accumulated across k PEs **vertically** with their respective row partial sums.
 - ▶ Need interconnection for PEs.
- ▶ **Filter rows are reused across PEs horizontally**
- ▶ Image rows (2,3) are delivered **diagonally** to designated PEs (PE 4,5) when computing the next output row (row 2). Only get one new input image row (row 4)
- ▶ **Need NoC to deliver the already fetched data to the diagonally neighboring PEs.**
- ▶ **Need sophisticated data routing/timing to feed the destination PEs for diagonally transferred data and the new input rows.**



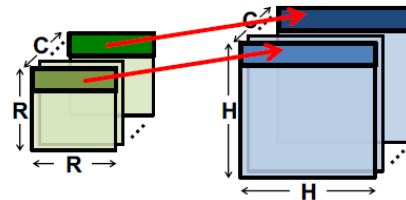
Greater details

- Pixel value in an output row is accumulated across PEs vertically with their row partial sums.
 - a, d, g, are generated within k cycles where **k is the kernel row width** (, i.e., k x k)
 - Send g, after k cycles, and add with d in PE2. However, g and d are generated at the same k cycles
 - Meaning, accumulating of g, d, a at PE2 and PE1, takes k-1 more cycles to add up.
 - On the other hand at PE3, it may latch g, h, i, and then send up at a time, but this requires PE2 to add h and e, f and i. So this is probably not the case.
- Pixel values in output **row 2** are generated after **row 1**, etc.



Multiple Channels Extension

- ▶ Compute a row output by
 - ▶ interleaving the same filter rows of all channels, say row1.
 - ▶ Interleaving the same input rows of all channels, say row1.
- ▶ So that an output pixel value of that row, ie, the multiply results of the interleaved channels can be accumulated in row convolution



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

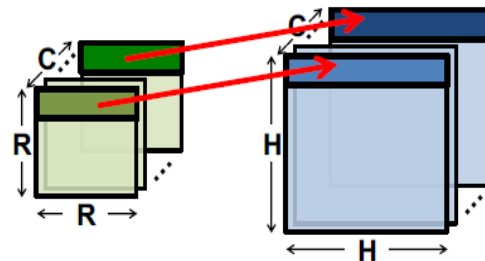
accumulate psums

Processing in PE: interleave channels

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

Multiple Channel, alternative

- ▶ One filter with multiple channels
- ▶ Spatial allocation of channel weights to a new set of PEs, e.g., Channel 2,
- ▶ Input image rows are reused in a channel, but not across channel since each channel has their respective data.



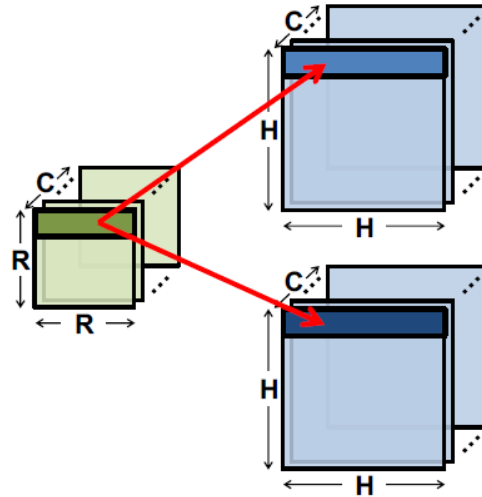
$$\begin{array}{lcl} \text{Channel 1} & \text{Filter 1} & \text{Image 1} \\ & \boxed{\text{Row 1}} & * \boxed{\text{Row 1}} = \boxed{\text{Row 1}} \\ & & \text{Psum 1} \\ \\ \text{Channel 2} & \text{Filter 1} & \text{Image 1} \\ & \boxed{\text{Row 1}} & * \boxed{\text{Row 1}} = \boxed{\text{Row 1}} \\ & & \text{Psum 1} \end{array}$$

accumulate psums

$$\boxed{\text{Row 1}} + \boxed{\text{Row 1}} = \boxed{\text{Row 1}}$$

Multiple Image Extension

- ▶ Dimensions beyond 2D
- ▶ For the same channel, concatenate image rows so that the same filter row can be reused for different images.



$$\begin{array}{lcl} \text{Channel 1} & \begin{array}{c} \text{Filter 1} \\ \text{Row 1} \end{array} * & \begin{array}{c} \text{Image 1} \\ \text{Row 1} \end{array} = \begin{array}{c} \text{Psum 1} \\ \text{Row 1} \end{array} \\ \text{Channel 1} & \begin{array}{c} \text{Filter 1} \\ \text{Row 1} \end{array} * & \begin{array}{c} \text{Image 2} \\ \text{Row 1} \end{array} = \begin{array}{c} \text{Psum 2} \\ \text{Row 1} \end{array} \end{array}$$

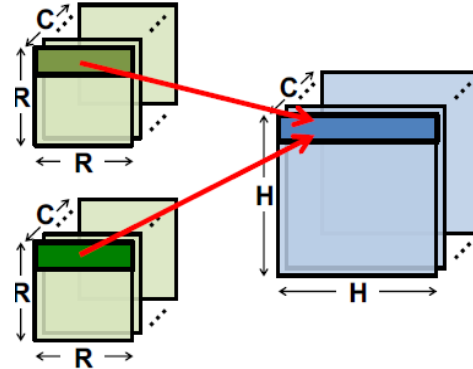
share the same filter row

concatenate image rows

$$\begin{array}{lcl} \text{Channel 1} & \begin{array}{c} \text{Filter 1} \\ \text{Row 1} \end{array} * & \begin{array}{c} \text{Image 1 \& 2} \\ \text{Row 1} \quad \text{Row 1} \end{array} = \begin{array}{c} \text{Psum 1 \& 2} \\ \text{Row 1} \quad \text{Row 1} \end{array} \end{array}$$

Multiple Filter Extension



- ▶ Dimensions beyond 2D
- ▶ In multiple filters, for the same channel, interleave the filter rows so that all filter rows can MAC on the same input image row.



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

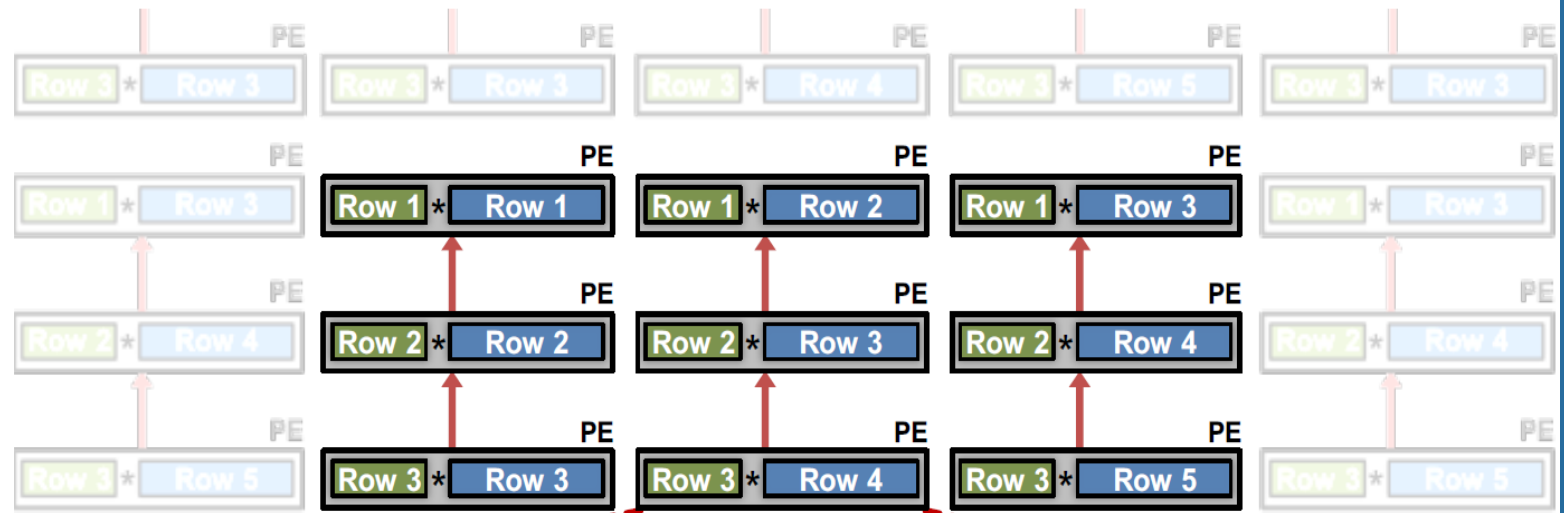
share the same image row

interleave filter rows

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

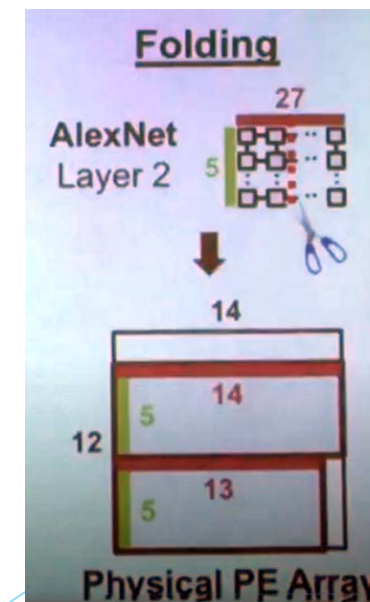
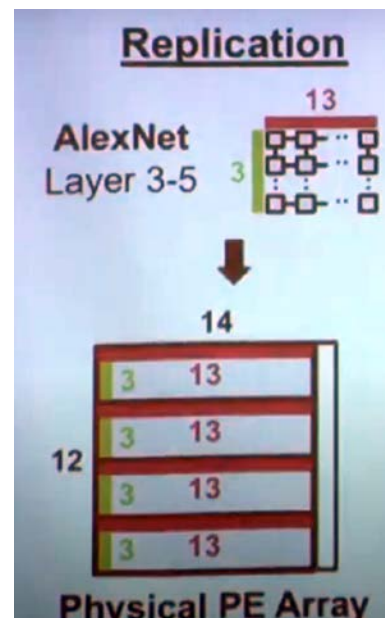
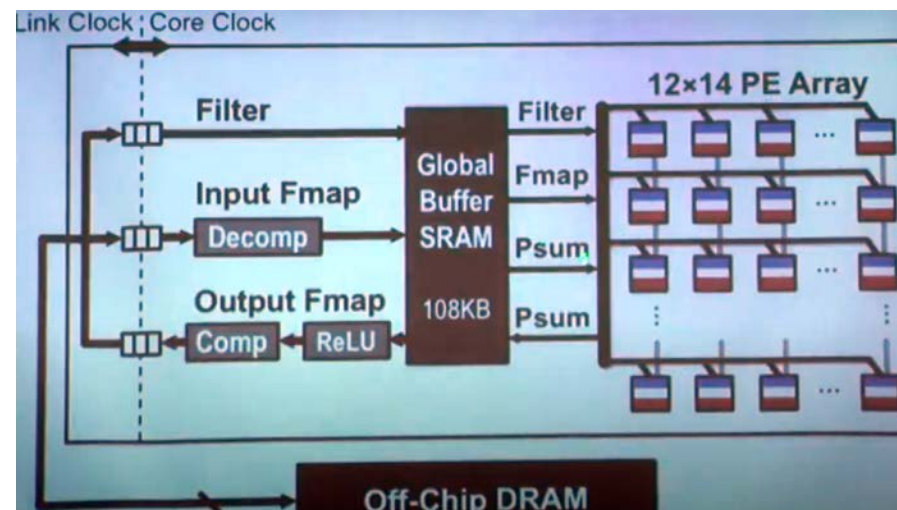
The Full Picture

- ▶ Filter, image row stationary
- ▶ Pixel output is accumulated vertically across k PEs.
- ▶ Filter rows are reused in PEs that participate in the convolution row operations
- ▶ Image rows are reused in PEs in the next row convolution
- ▶ New image row is fetched when changing rows, fetched rows are reused by diagonal PEs.



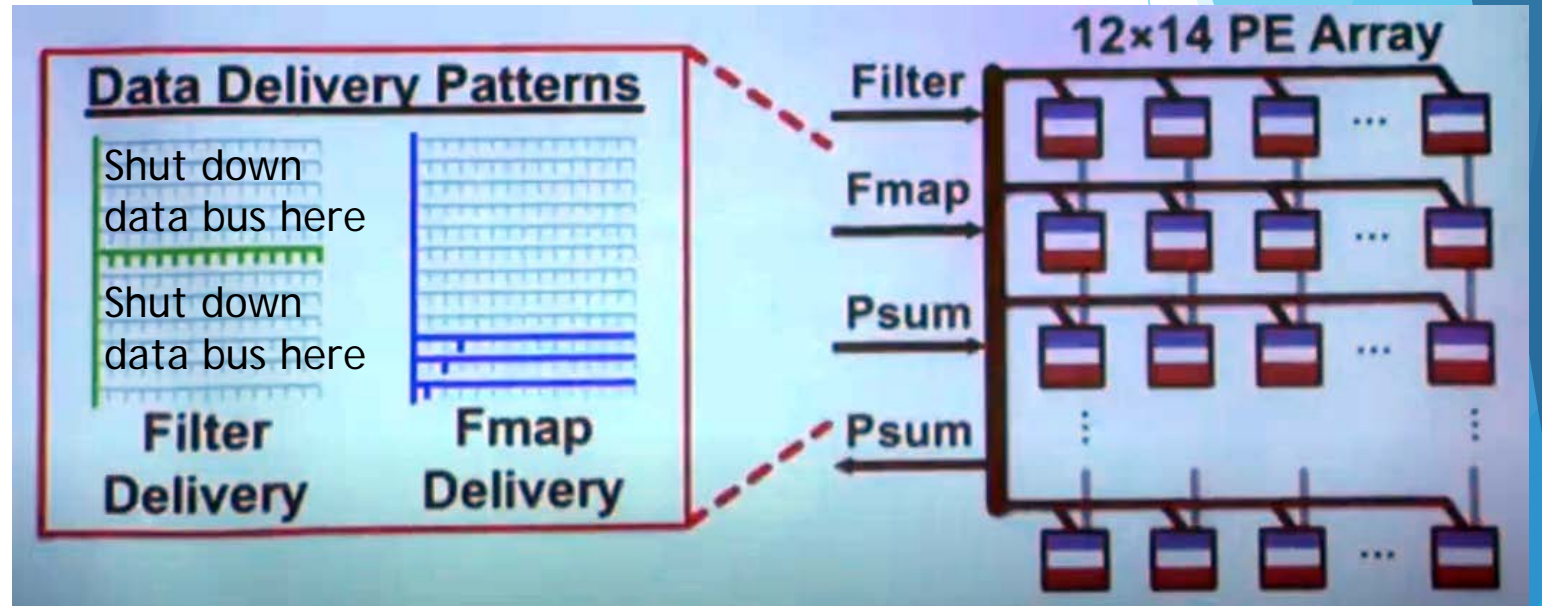
Mapping RS dataflow to PE array

- ▶ No. of PE rows = filter row # (filter size)
- ▶ No. of PE columns = output feature map size
- ▶ Fixed physical PE array 12x14
 - ▶ Alexnet layer 3-5, 3 x 13
 - ▶ Map different filter or feature map to the 3x13 PEs blocks
 - ▶ Replication if output is small than 12x14
 - ▶ Otherwise folding (best fit and repeat in time)



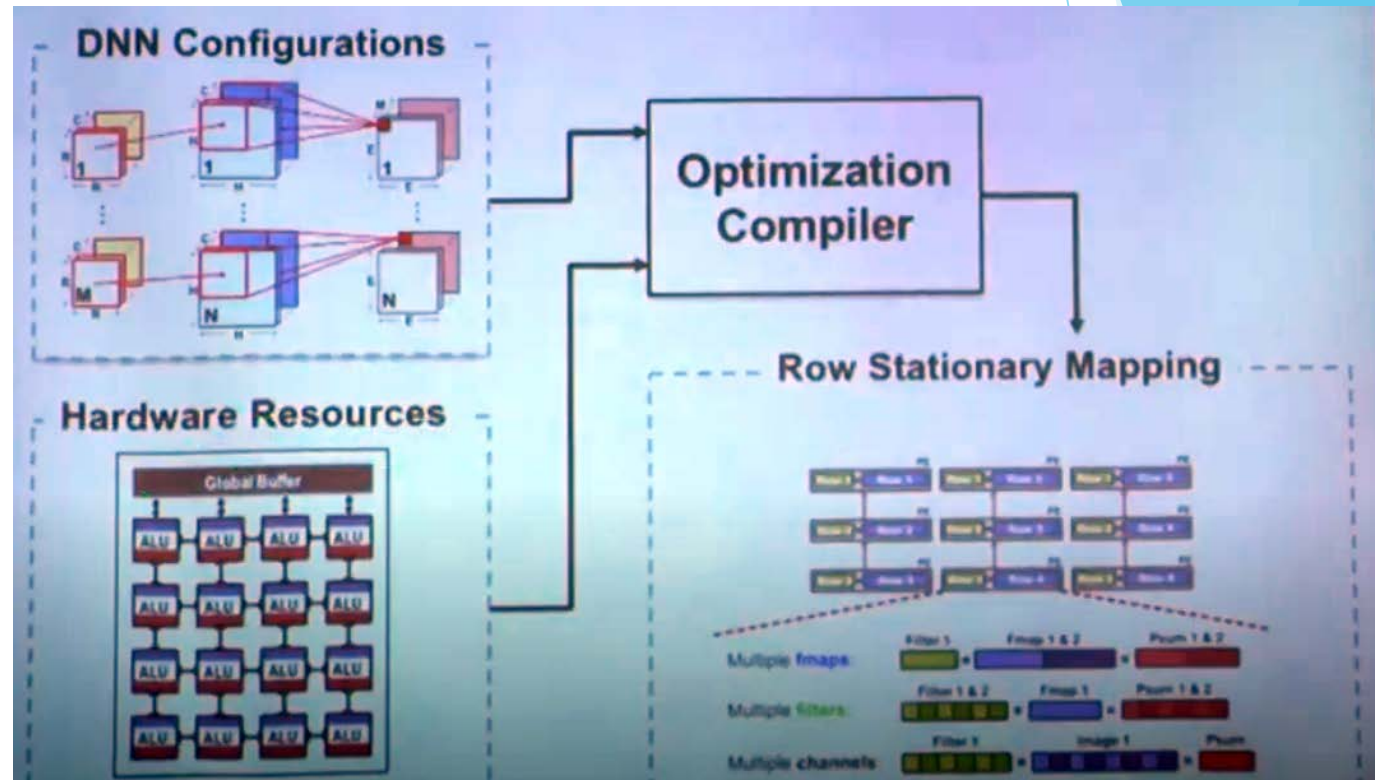
Data Delivery

- ▶ Multicast network instead of full network broadcast
- ▶ Deliver filters horizontally
- ▶ Deliver data diagonally
- ▶ Deliver partial sum vertically



Optimization

- ▶ Given a specific layer in a DNN, the physical hardware (PEs, buffer), how to generate the mapping/control for RS operation?
 - ▶ An optimization compiler
- ▶ PE gating for zero input/weight
- ▶ Run length coding for data compression to/from DRAM



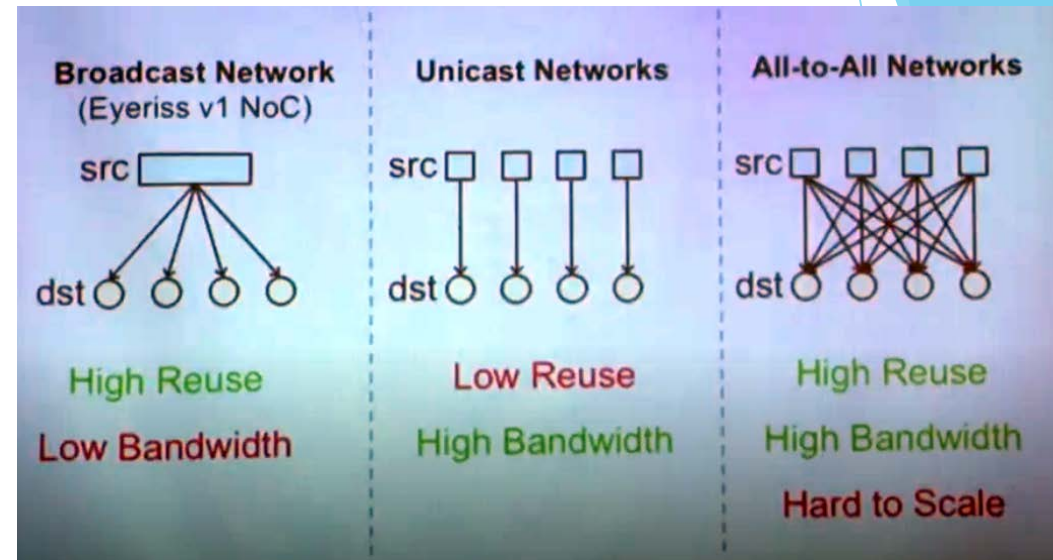
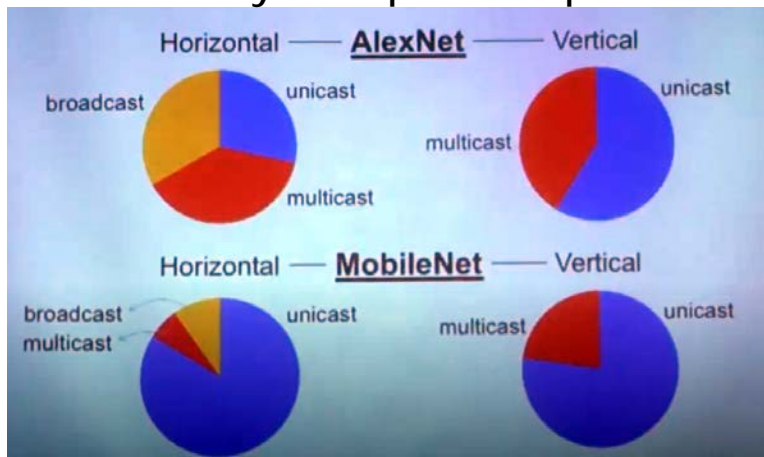
Eyeriss v2: Design Goals

- ▶ Support a wide range of data reuse
 - ▶ A good dataflow that reduces idle PEs or achieves good reuse
- ▶ Support a wide range of data sparsity both in feature maps and weights

Eyeriss v2: RS+

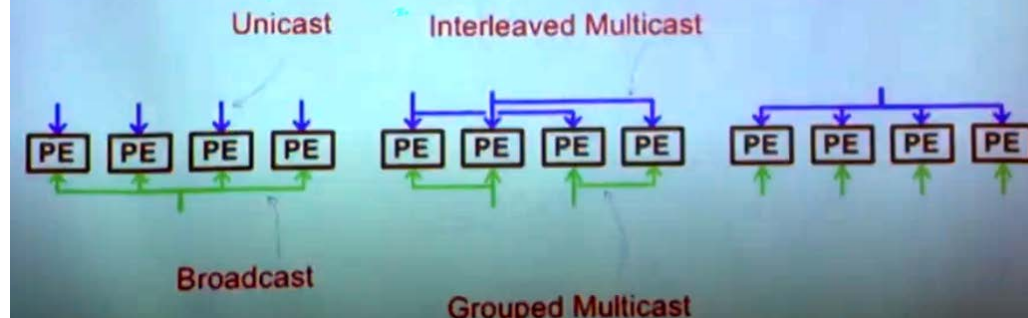
- ▶ Enhanced dataflow, RS+
 - ▶ Minimize the # of idle PEs
- ▶ Mesh NoC to support various type of communication traffic and data reuse requirement
 - ▶ Unicast
 - ▶ Interleaved group multicast
 - ▶ Broadcast
 - ▶ Support high data reuse

Delivery of Input Fmaps in RS



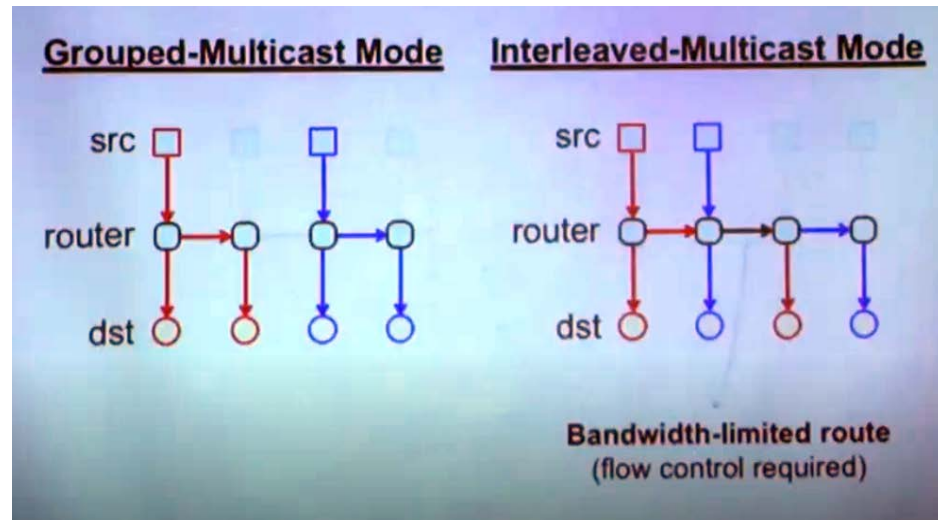
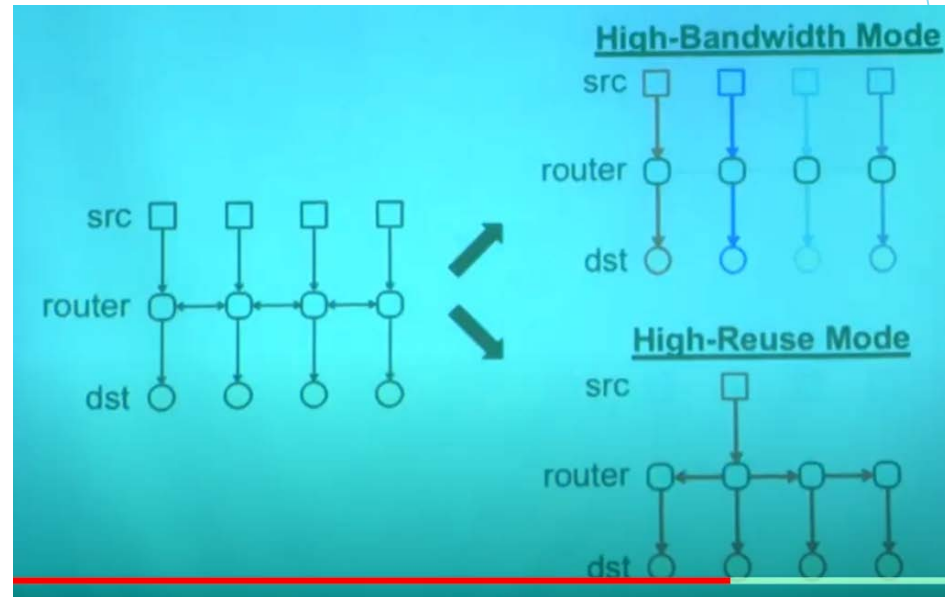
- **Dataflow:** always keep as many PEs busy as possible
- **NoC:** adapt to the bandwidth and reuse requirements

4 Data Delivery Patterns



Mesh Network

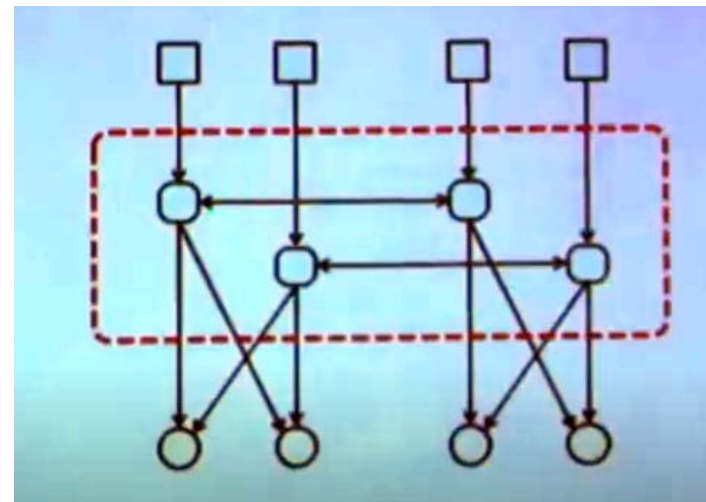
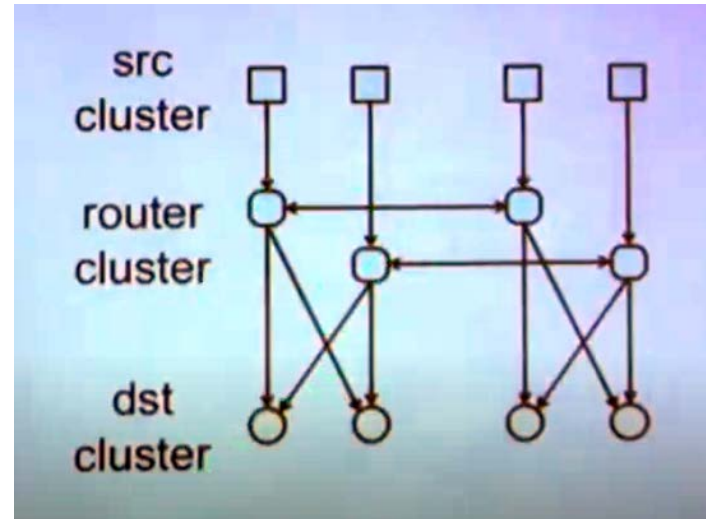
- ▶ Support high-bandwidth or high-reuse
- ▶ Not good for interleaved-multicast mode



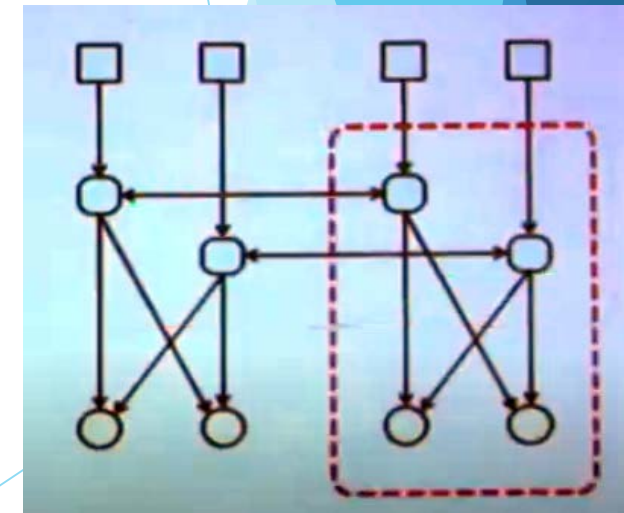
Hierarchical Mesh Network for Eyeriss v2

► Example

- Group 2 src, 2 dst, 2 router into a cluster respectively
- Mesh to connect inter-cluster
- All-to-all network for intra-cluster connections to limit complexity in a cluster



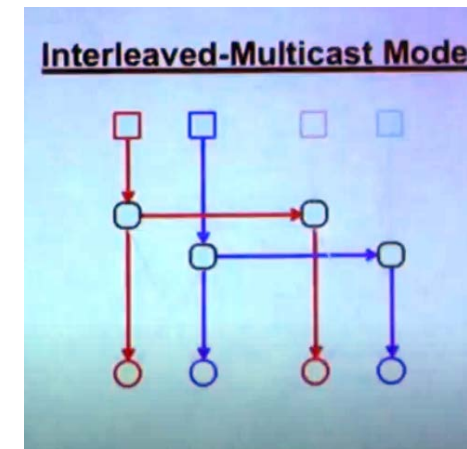
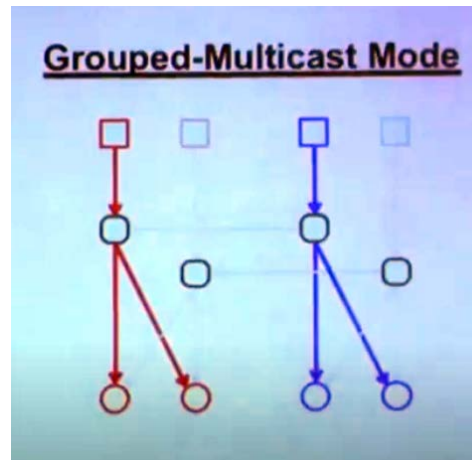
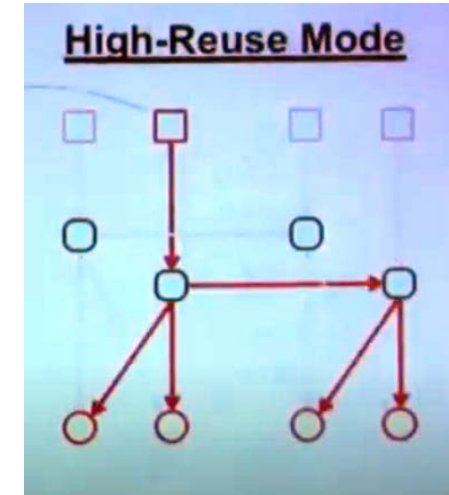
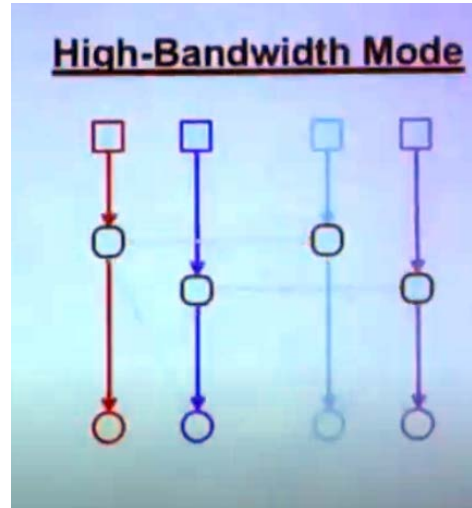
Inter-cluster



Intra-cluster

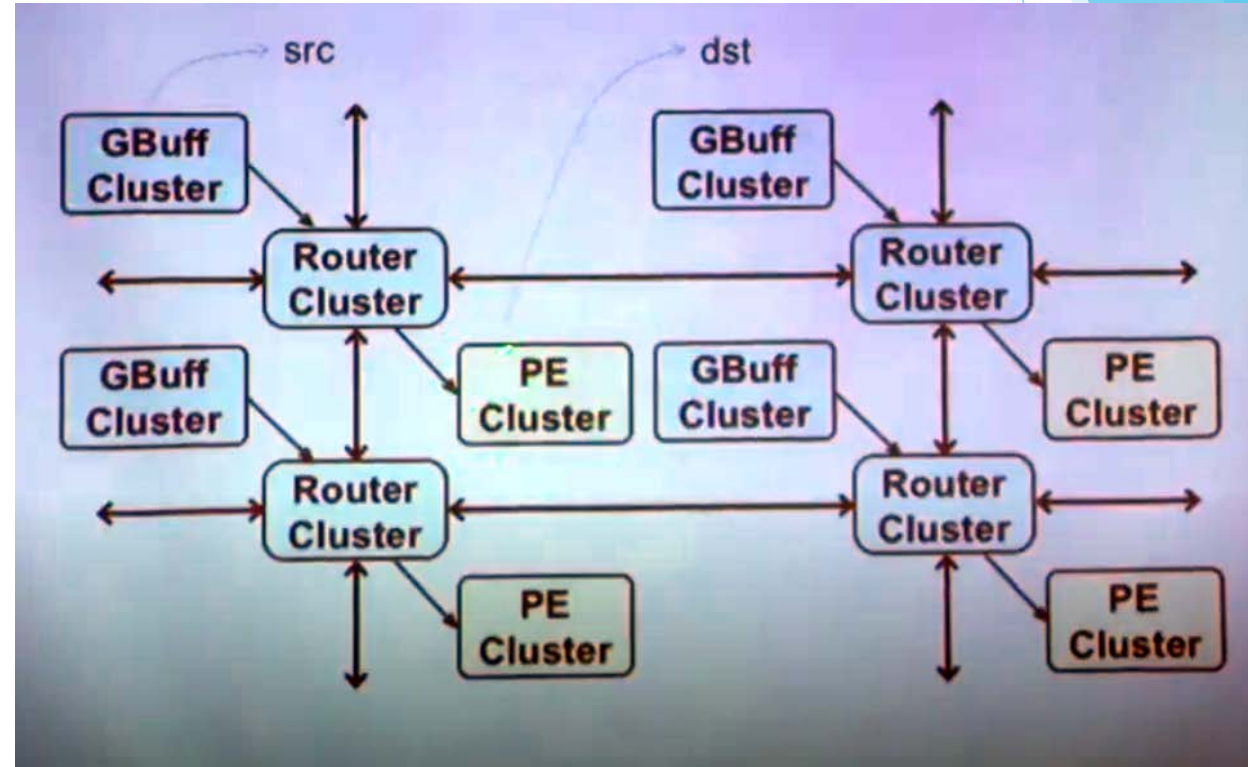
Hierarchical Mesh Network for Eyeriss v2

- ▶ Supported traffic types
- ▶ High bandwidth
- ▶ High-reuse
- ▶ Grouped-multicast
- ▶ Interleaved-multicast
- ▶ Build up cluster size to support higher number of multicast
- ▶ Routers are MUXes; routes determined at configuration time.
 - ▶ Circuit-switched



Eyeriss v2 Architecture

- ▶ V2 is a cluster, distributed, hierarchical mesh architecture
- ▶ Accelerator calls for
 - ▶ Good data reuse or dataflow for diverse set of DNNs.
 - ▶ Rely on efficient dataflow-based micro-architecture
 - ▶ Good PE utilization
 - ▶ Mapping for various filter size, channel size, image size
 - ▶ Mapping of DNN application IR graph to the hardware and dataflow features



Machine Learning

Hello!! R2-D2



Thanks You