# Hardware for Machine Learning

## Challenges and Opportunities

深圳大学 杨树 2018-10-26

1. Sze V, Chen Y H, Emer J, et al. Hardware for machine learning: Challenges and opportunities[C]. Custom Integrated Circuits Conference (CICC). IEEE, 2018: 1-8.
2. Chen Y H, Emer J, Sze V. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks[C]. ACM SIGARCH Computer Architecture News. IEEE Press, 2016, 44(3): 367-379.
3. Chen Y H, Krishna T, Emer J S, et al. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks[J]. IEEE Journal of Solid-State Circuits, 2017, 52(1): 127-138.

# 目录 CONTENTS

01 **面临的挑战**

Challenges

02 **架构上的改进**

Opportunities in Architectures

03 **具体的设计**

以 Eyeriss 为实例

04 **总结**

Summary

01

PART ONE

# 面临的挑战

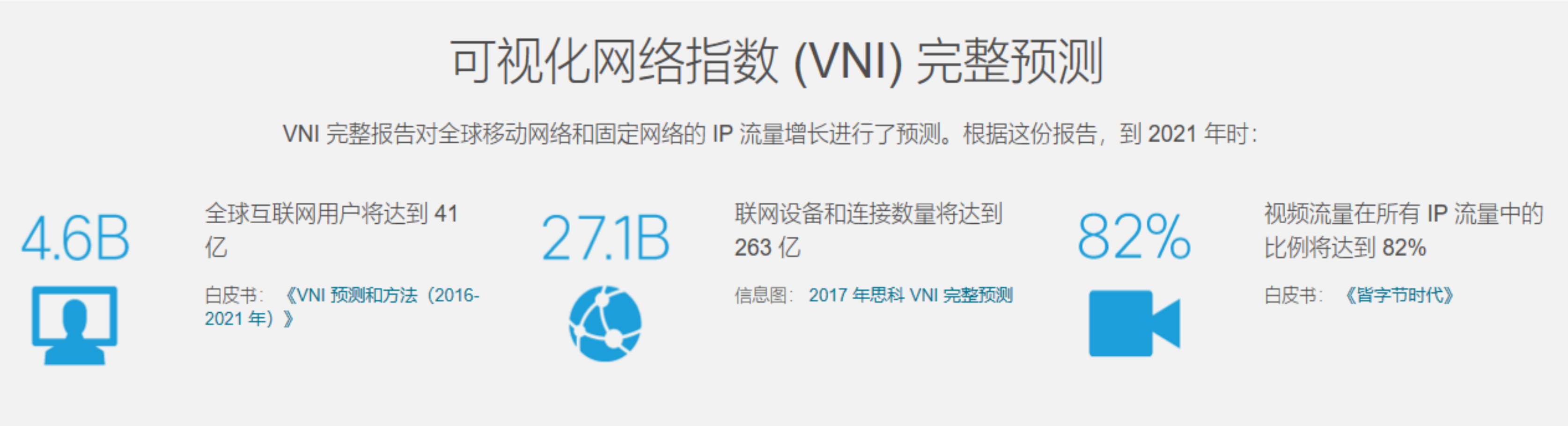
01

# 面临的挑战

Challenges

## BigData

This is the era of big data.



可视化网络指数 (VNI) 完整预测

VNI 完整报告对全球移动网络和固定网络的 IP 流量增长进行了预测。根据这份报告，到 2021 年时：

4.6B

全球互联网用户将达到 41 亿

白皮书：《VNI 预测和方法（2016-2021 年）》

27.1B

联网设备和连接数量将达到 263 亿

信息图：2017 年思科 VNI 完整预测

82%

视频流量在所有 IP 流量中的比例将达到 82%

白皮书：《皆字节时代》

数据来源：https://www.cisco.com/c/zh_cn/solutions/service-provider/visual-networking-index-vni/index.html

01

# 面临的挑战

Challenges

## Computer Vision

视频数据数量庞大、图片像素越来越高

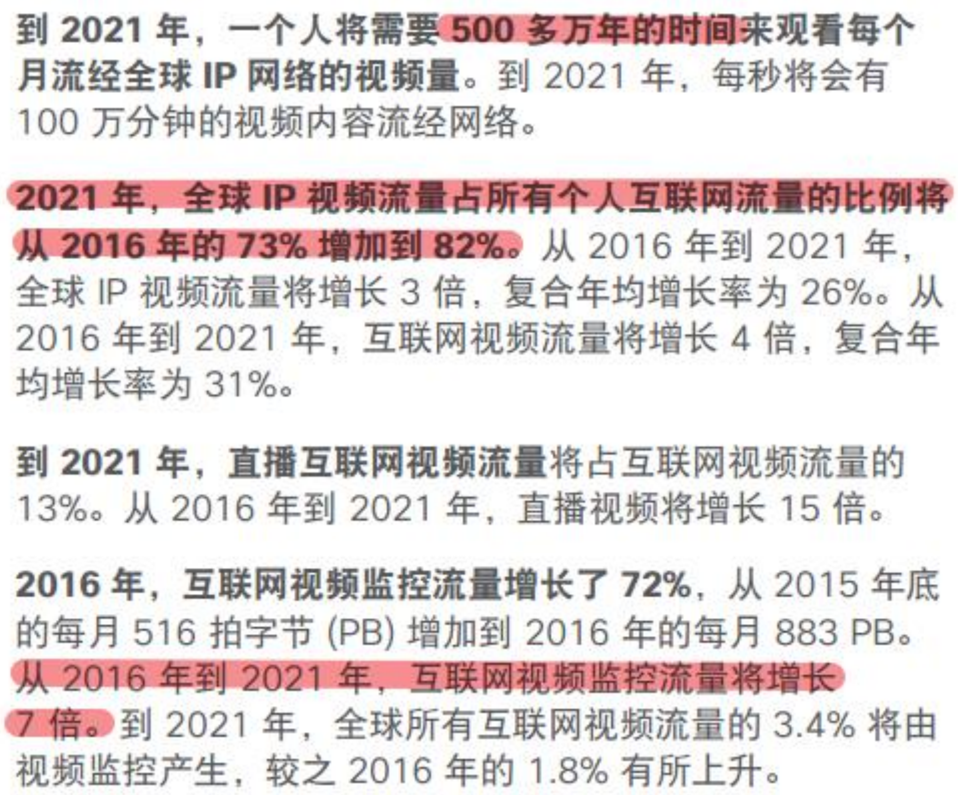
白皮书
思科公开信息

cisco

思科可视化网络指数：预测和方法，2016-2021 年

2017 年 8 月

算法有待改进
硬件同样不可忽视！



**到 2021 年，一个人将需要 500 多万年的时间来观看每个月流经全球 IP 网络的视频量。**到 2021 年，每秒将会有 100 万分钟的视频内容流经网络。

**2021 年，全球 IP 视频流量占所有个人互联网流量的比例将从 2016 年的 73% 增加到 82%。**从 2016 年到 2021 年，全球 IP 视频流量将增长 3 倍，复合年均增长率为 26%。从 2016 年到 2021 年，互联网视频流量将增长 4 倍，复合年均增长率为 31%。

**到 2021 年，直播互联网视频流量**将占互联网视频流量的 13%。从 2016 年到 2021 年，直播视频将增长 15 倍。

**2016 年，互联网视频监控流量增长了 72%，**从 2015 年底的每月 516 拍字节 (PB) 增加到 2016 年的每月 883 PB。从 2016 年到 2021 年，互联网视频监控流量将增长 7 倍。到 2021 年，全球所有互联网视频流量的 3.4% 将由视频监控产生，较之 2016 年的 1.8% 有所上升。

数据来源网址：
https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html

01

# 面临的挑战

Challenges

目前，大多数语音服务的处理仍在云端，例如Apple Siri、Amazon Alexa

**语音识别**

## 提升终端设备性能的好处

- ✓ 减少延迟
- ✓ 减少对连接的依赖
- ✓ 保护用户隐私

医疗领域需要能够帮助患者检测、诊断和治疗各种疾病的设备

**医疗领域**

01

# 面临的挑战

Challenges

**问题：终端设备的硬件该如何设计?**

✓ 减少计算

✓ 数据移动

✓ 存储要求

目前，许多硬件研究都集中在降低乘法和累加的成本上（MAC，the cost of a multiply and accumulate ）

# 01 面临的挑战

Challenges

Deep Neural Network（DNN）

深度神经网络中就包含着许多乘和加的操作

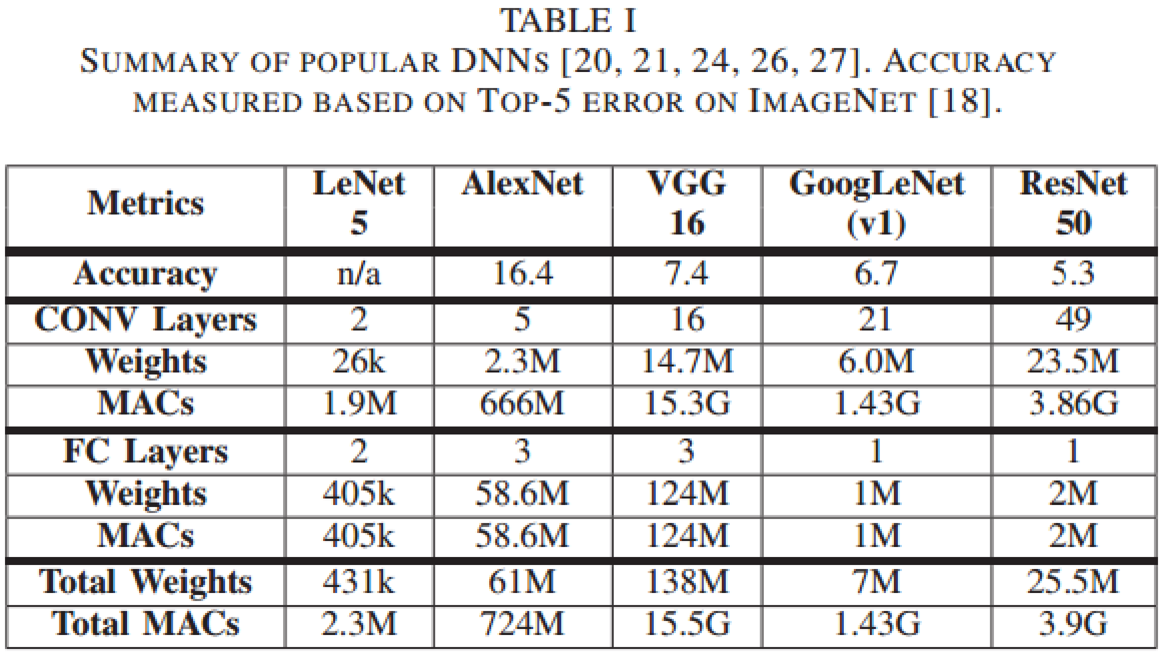


TABLE I

SUMMARY OF POPULAR DNNS [20, 21, 24, 26, 27]. ACCURACY MEASURED BASED ON TOP-5 ERROR ON IMAGENET [18].

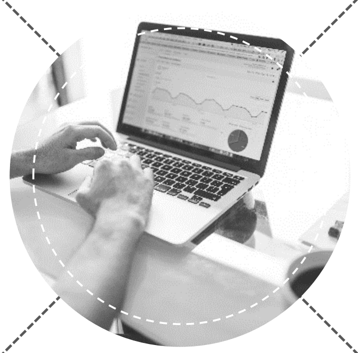
| Metrics | LeNet 5 | AlexNet | VGG 16 | GoogLeNet (v1) | ResNet 50 |
|---|---|---|---|---|---|
| Accuracy | n/a | 16.4 | 7.4 | 6.7 | 5.3 |
| CONV Layers | 2 | 5 | 16 | 21 | 49 |
| Weights | 26k | 2.3M | 14.7M | 6.0M | 23.5M |
| MACs | 1.9M | 666M | 15.3G | 1.43G | 3.86G |
| FC Layers | 2 | 3 | 3 | 1 | 1 |
| Weights | 405k | 58.6M | 124M | 1M | 2M |
| MACs | 405k | 58.6M | 124M | 1M | 2M |
| Total Weights | 431k | 61M | 138M | 7M | 25.5M |
| Total MACs | 2.3M | 724M | 15.5G | 1.43G | 3.9G |

01

# 面临的挑战

Challenges

## 移动终端设备



### 精度 (Accuracy)

应该在足够大的数据集上测量机器学习算法的准确性

### 能耗 (Energy Consumptiion)

计算和数据的移动
（数据移动的成本通常要高于计算成本）

### 吞吐量/延迟 (Throughput/Latency)

吞吐量由计算量决定，计算量也随着数据的维数而增加

### 成本 (Cost)

成本取决于芯片所需的存储量
通过降低存储成本以减少芯片面积，同时保持低片外存储器带宽

01

# 面临的挑战

Challenges

## 小结

针对如下的两个问题：

- 首先，若是移动设备无法连接到互联网，就无法执行深度学习任务。而当你将数据传输到互联网上的远程服务器的时候，又会有**隐私**方面的担忧。
- 其次，数据传输和远程处理会带来**延迟**。

让**移动设备**执行自然语言处理和面部识别等任务，而无需连接至互联网。

显然，这是让操作系统的机器学习变得**更加便携**的最新尝试，同时也减轻了**服务器**的负担。

**重要的是如何降低能耗！！！**

02

PART TWO

# 架构上的改进

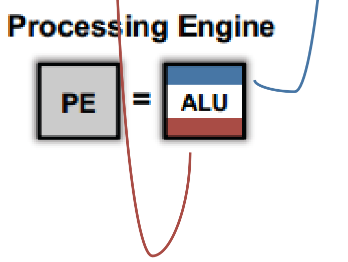
# Two common highly-parallel compute paradigms

■ Memory Hierarchy

■ Register File

□ ALU

■ Control



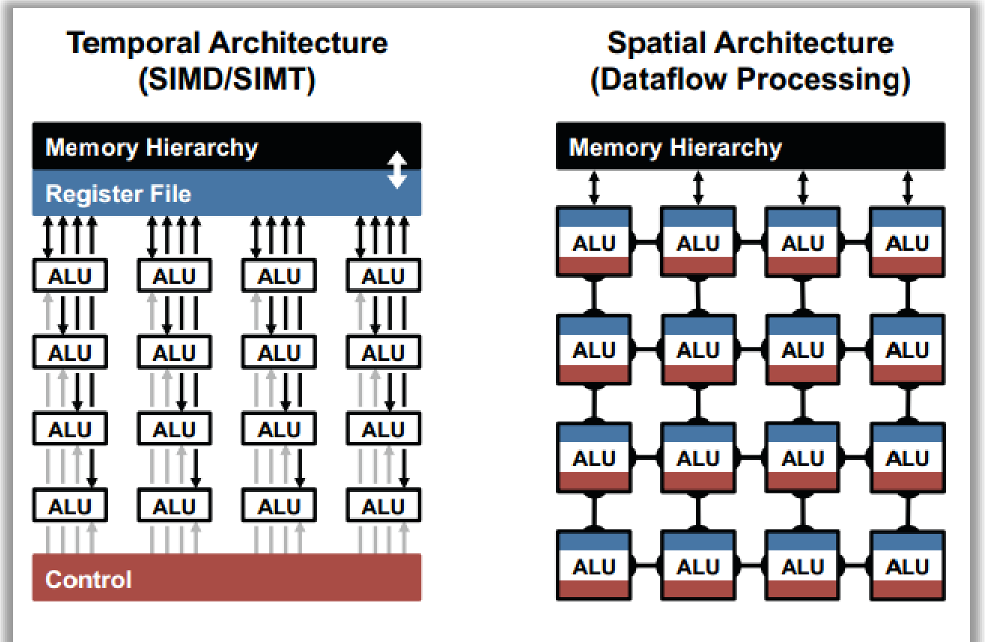


Fig. 6. Highly-parallel compute paradigms.

CPU 和 GPU

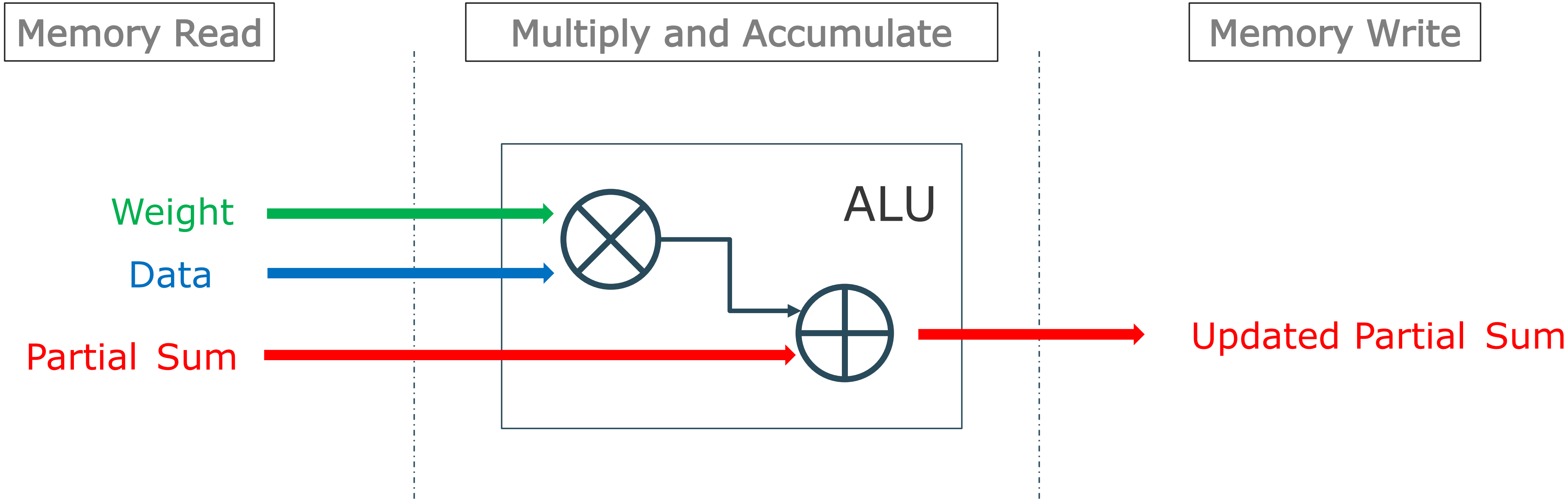
使用诸如SIMD或SIMT的时间架构来并行执行乘加操作

所有ALU共享相同的控制和内存（寄存器文件）

02

## 架构上的改进

Opportunities in Architectures



**频繁的数据移动会影响硬件的性能**

**(对内存的访问是一个瓶颈)**

02

# 架构上的改进

Opportunities in Architectures

右图为 Eyeriss芯片 的架构

图片的下半部分是从不同的存储位置移动数据所带来的能耗

## Eyeriss

采用了空间架构（Spatial Architecture）

**主要改进在加速器部分（即Accelerator)**

具体是优化了数据移动（Dataflow架构），尽量减少对全局缓冲器和片外存储器的访问

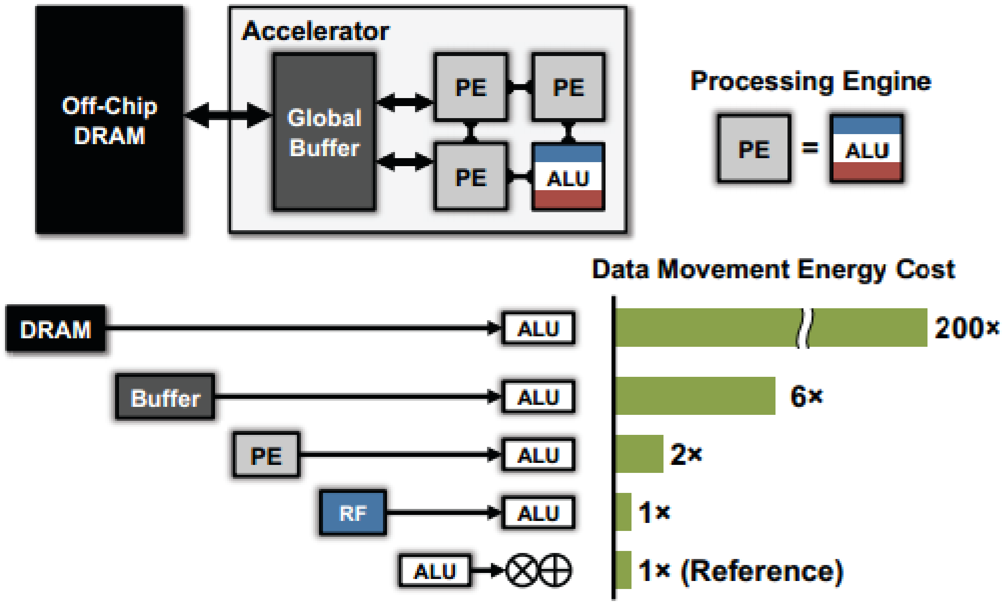


Fig. 7.  Memory hierarchy and data movement energy [34].

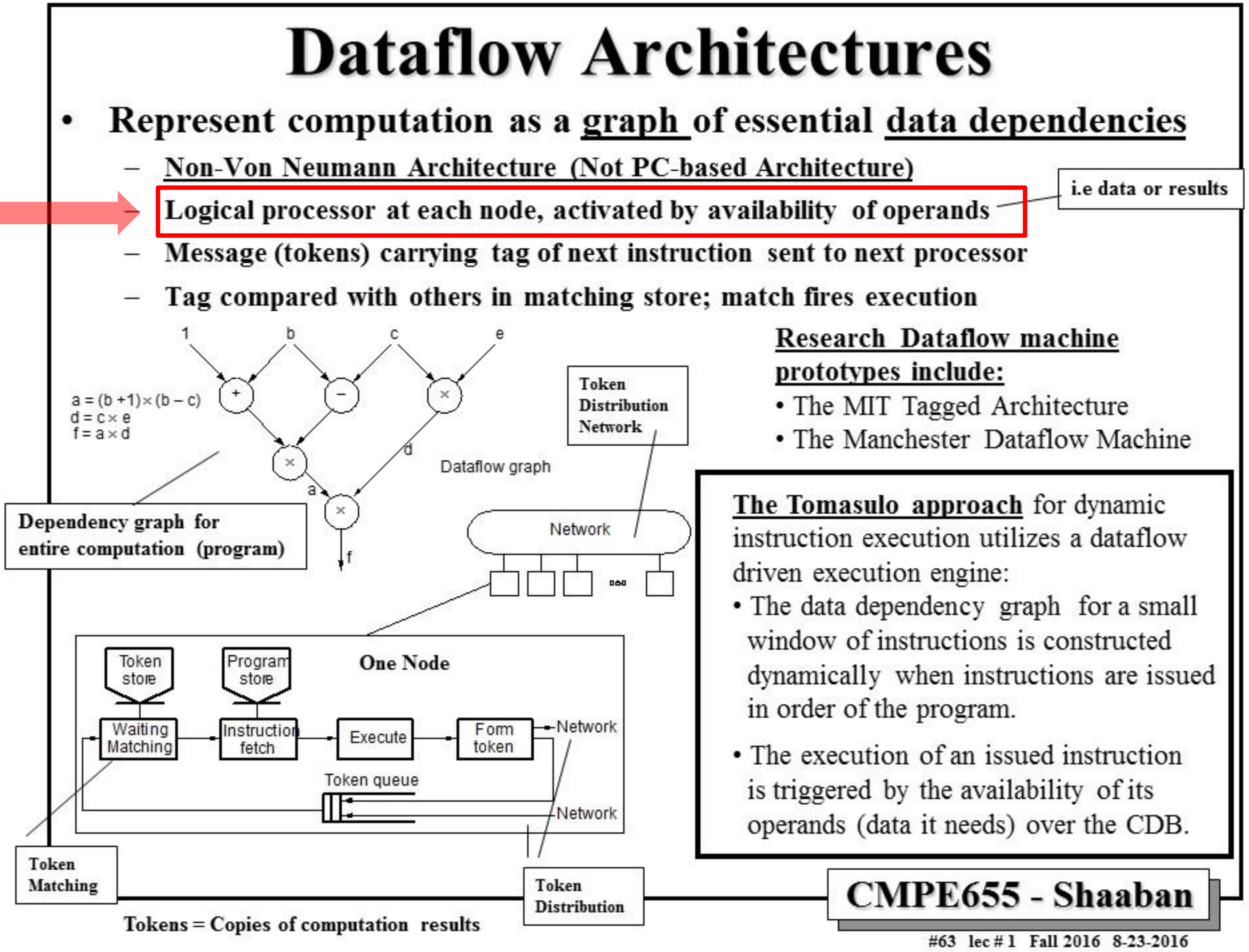
# 02 架构上的改进

Opportunities in Architectures

**Dataflow**

是一种没有复杂程序指令控制且由操作数，即数据或者中间结果，激活子计算单元，来实现并行计算的一种计算方式。

[Dataflow]处理方式与基于深度神经网络推断部分的计算需求非常吻合



## Dataflow Architectures

- Represent computation as a graph of essential data dependencies
  - Non-Von Neumann Architecture (Not PC-based Architecture)
  - Logical processor at each node, activated by availability of operands
  - Message (tokens) carrying tag of next instruction sent to next processor
  - Tag compared with others in matching store; match fires execution

i.e data or results

Dependency graph for entire computation (program)

Token Matching

Token Distribution

Tokens = Copies of computation results

**Research Dataflow machine prototypes include:**

- The MIT Tagged Architecture
- The Manchester Dataflow Machine

**The Tomasulo approach** for dynamic instruction execution utilizes a dataflow driven execution engine:

- The data dependency graph for a small window of instructions is constructed dynamically when instructions are issued in order of the program.
- The execution of an issued instruction is triggered by the availability of its operands (data it needs) over the CDB.

CMPE655 - Shaaban

#63 lec # 1 Fall 2016 8-23-2016

图片来自罗切斯特理工学院教授Shaaban的CMPE655

02

# 架构上的改进

Opportunities in Architectures

层次存储器中，存储量大的存储器读写操作所消耗的能量要比小存储的存储器大很多。

因而，一旦一块数据从大存储器搬移到小存储器后，要尽可能最大程度复用（reuse）该数据块来最小化能耗。

但是低功耗存储器的存储空间有限。如何最大化复用率是设计基于Dataflow加速器时最受关注的条件。

即，通过最大化数据复用率来降低I/O要求，减小数据处理能耗，从而提升吞吐量并降低总体能耗。

02

# 架构上的改进

Opportunities in Architectures

## 常见的DNN数据流类型

(a)权值固定数据流（Weight Stationary)

将读取到的权值数据存放在PE中的RF

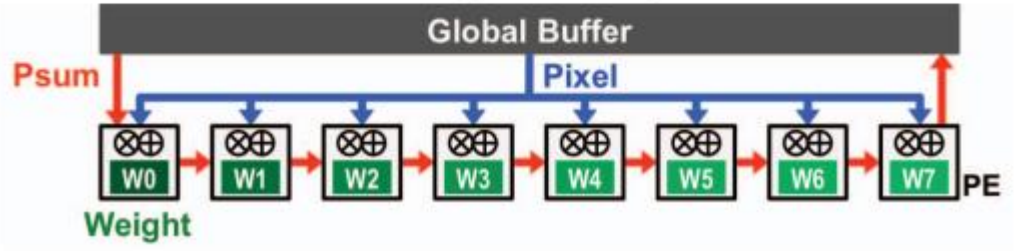
最大化从PE的RF中读取权值数据次数

最小化直接从DRAM中读取权值次数

从而实现**最大化权值的复用率**来减小能耗

例如：NeuFlow芯片

(a) Weight Stationary



02

# 架构上的改进

Opportunities in Architectures

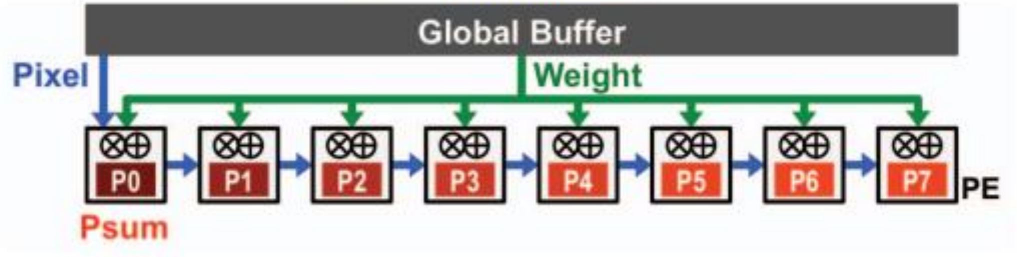
## 常见的DNN数据流类型

### (b)输出固定数据流（Output Stationary)

最小化读写部分和的能耗

例如：寒武纪的ShiDianNao芯片

(b) Output Stationary



# 02 架构上的改进

Opportunities in Architectures

## 常见的DNN数据流类型

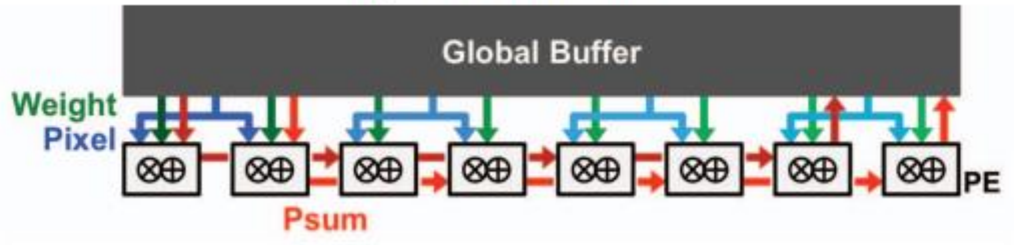
### (c)NLP数据流（No Local Reuse)

所有数据的读写操作都是在全局buffer中完成

所有的PE不分配局部缓存，而将该区域全部贡献给全局缓存以增加其容量

例如：寒武纪的DianNao与DaNiaoNao芯片、Google的TPU

(c) No Local Reuse



# NLP Example：TPU

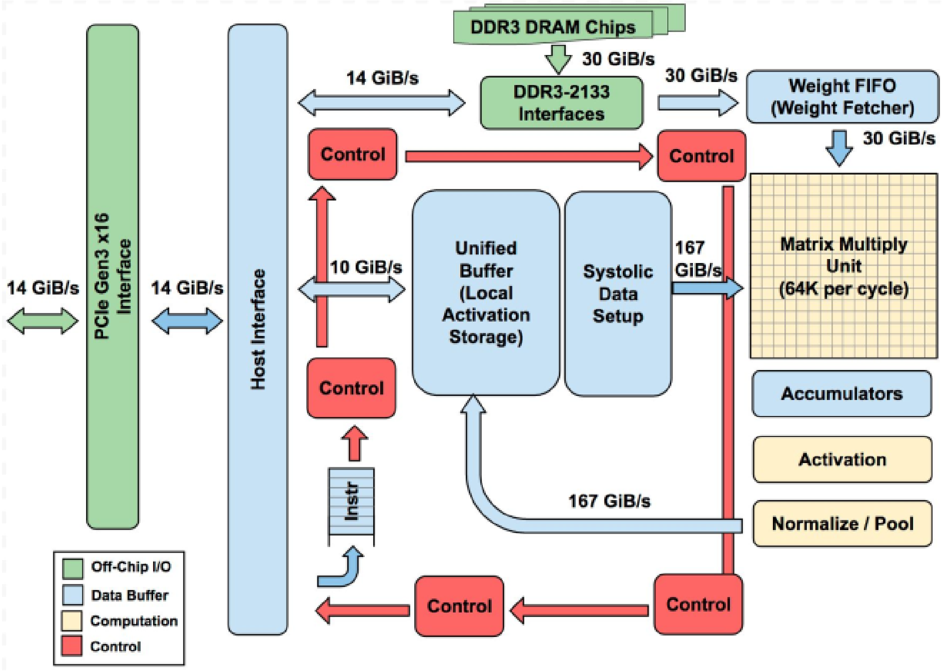


**Figure 1.** TPU Block Diagram. The main computation part is the yellow Matrix Multiply unit in the upper right hand corner. Its inputs are the blue Weight FIFO and the blue Unified Buffer (UB) and its output is the blue Accumulators (Acc). The yellow Activation Unit performs the nonlinear functions on the Acc, which go to the UB.

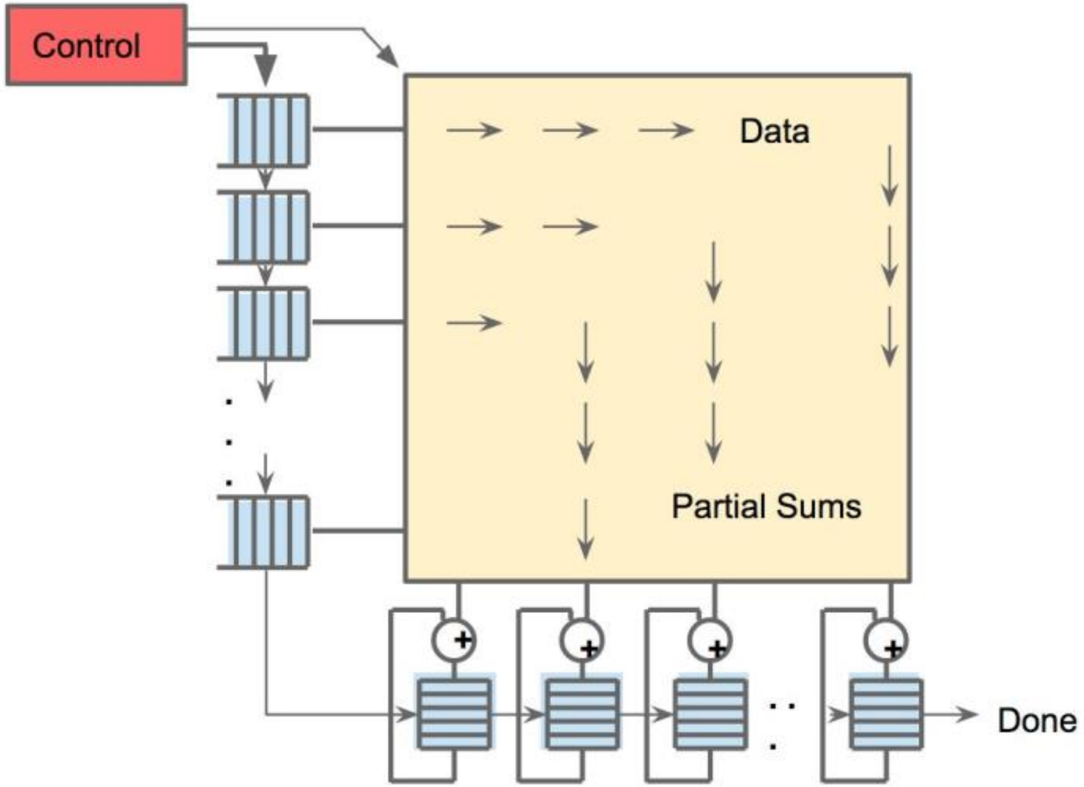


**Figure 4.** Systolic data flow of the Matrix Multiply Unit. Software has the illusion that each 256B input is read at once, and they instantly update one location of each of 256 accumulator RAMs.

图源：

N. P. Jouppi et al., "In-datacenter performance analysis of a tensor processing unit," 2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA), Toronto, ON, Canada, 2017, pp. 1-12.

# 02 架构上的改进

Opportunities in Architectures

## 行固定数据流（Row Stationary)

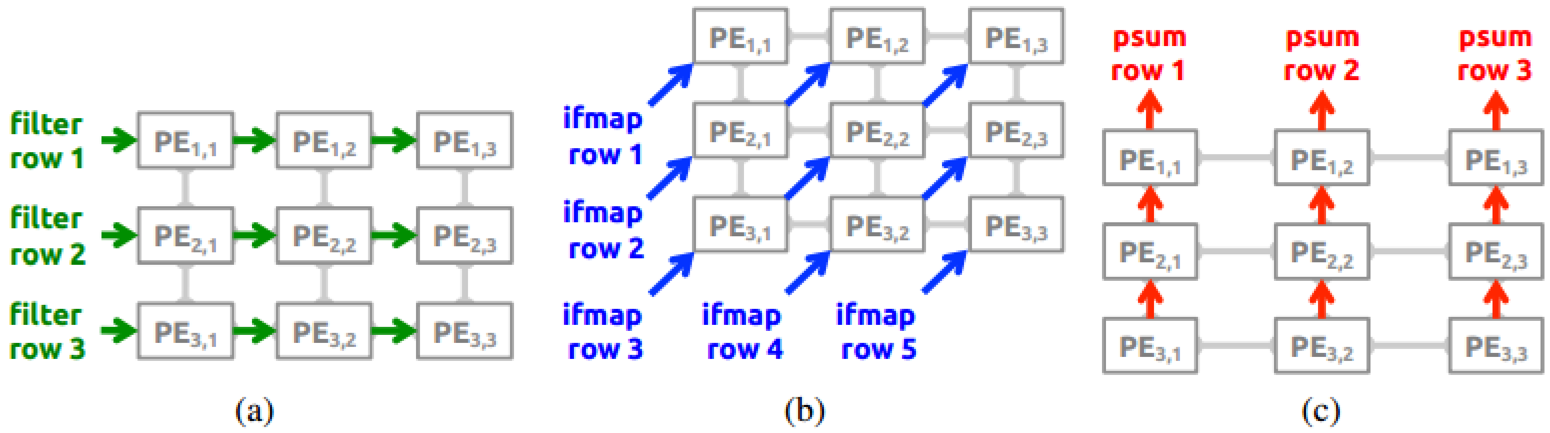
- 最大化所有数据复用率（而不是其中一种数据）
- 并尽可能的使得所有数据的读写操作都在RF中完成

**例：Eyeriss芯片**

每个PE能够完成1D的卷积运算

多个PE能够完成2D的卷积运算

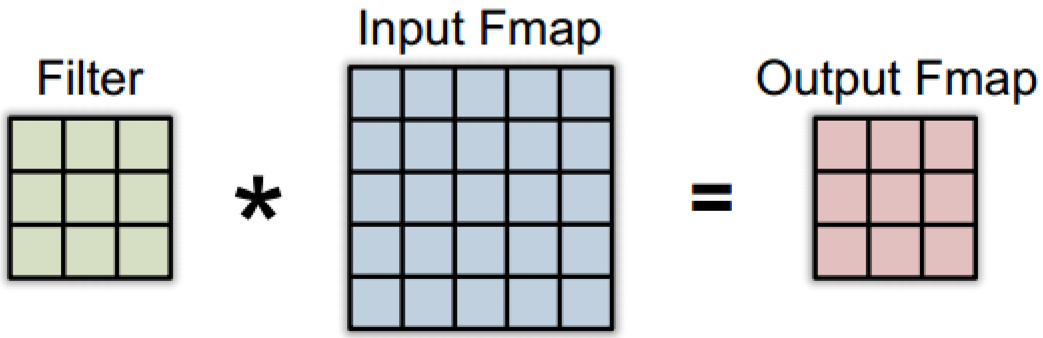
在二维的PE阵列中，
水平轴上的PE单元上，每一行的**权值数据**被复用
对角线上的PE单元上，每一行的**输入数据**被复用
垂直轴上的PE单元上，每一行的**部分和数据**被复用



(a) (b) (c)

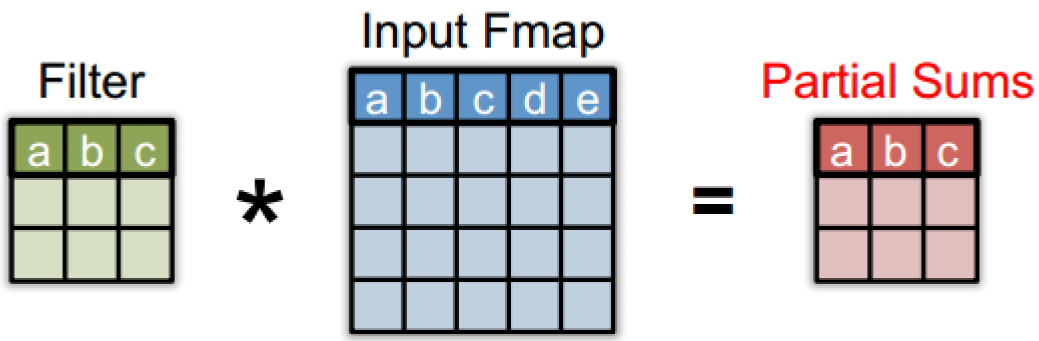
Figure 6. The dataflow in a logical PE set to process a 2D convolution. (a) rows of filter weight are reused across PEs horizontally. (b) rows of ifmap pixel are reused across PEs diagonally. (c) rows of psum are accumulated across PEs vertically. In this example, $R = 3$ and $H = 5$.

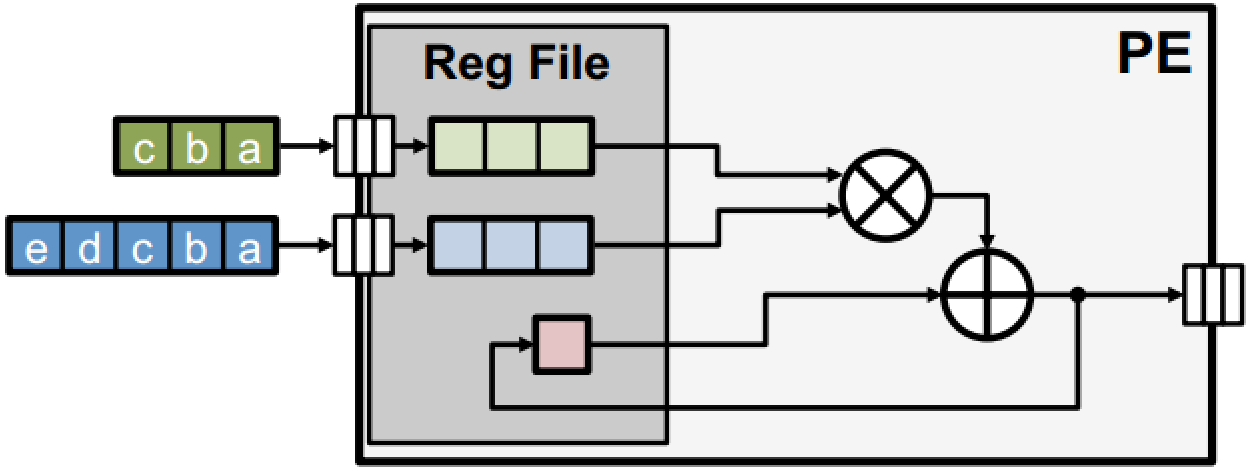
# Row Stationary: Energy-efficient Dataflow



图片来源：http://eyeriss.mit.edu/tutorial.html#ISCA

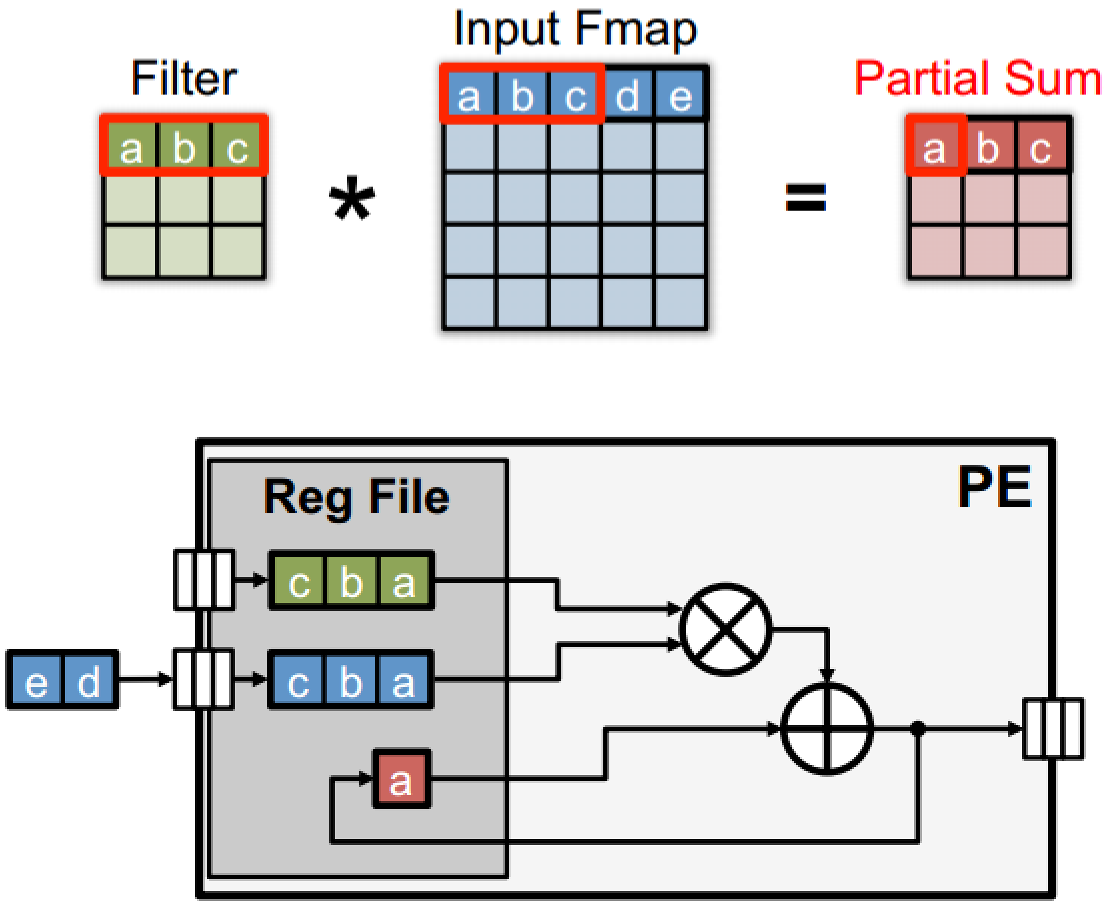
# 1D Row Convolution in PE





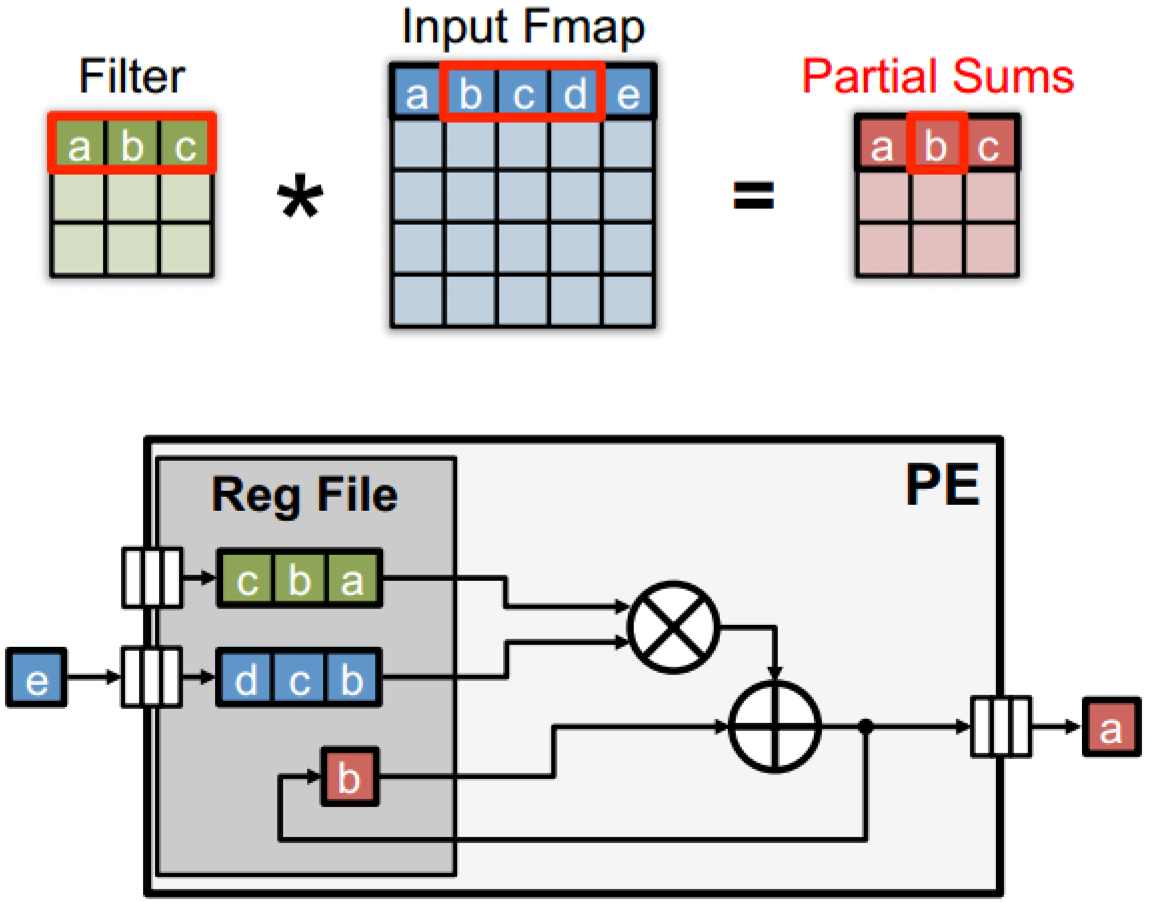
图片来源：http://eyeriss.mit.edu/tutorial.html#ISCA

# 1D Row Convolution in PE



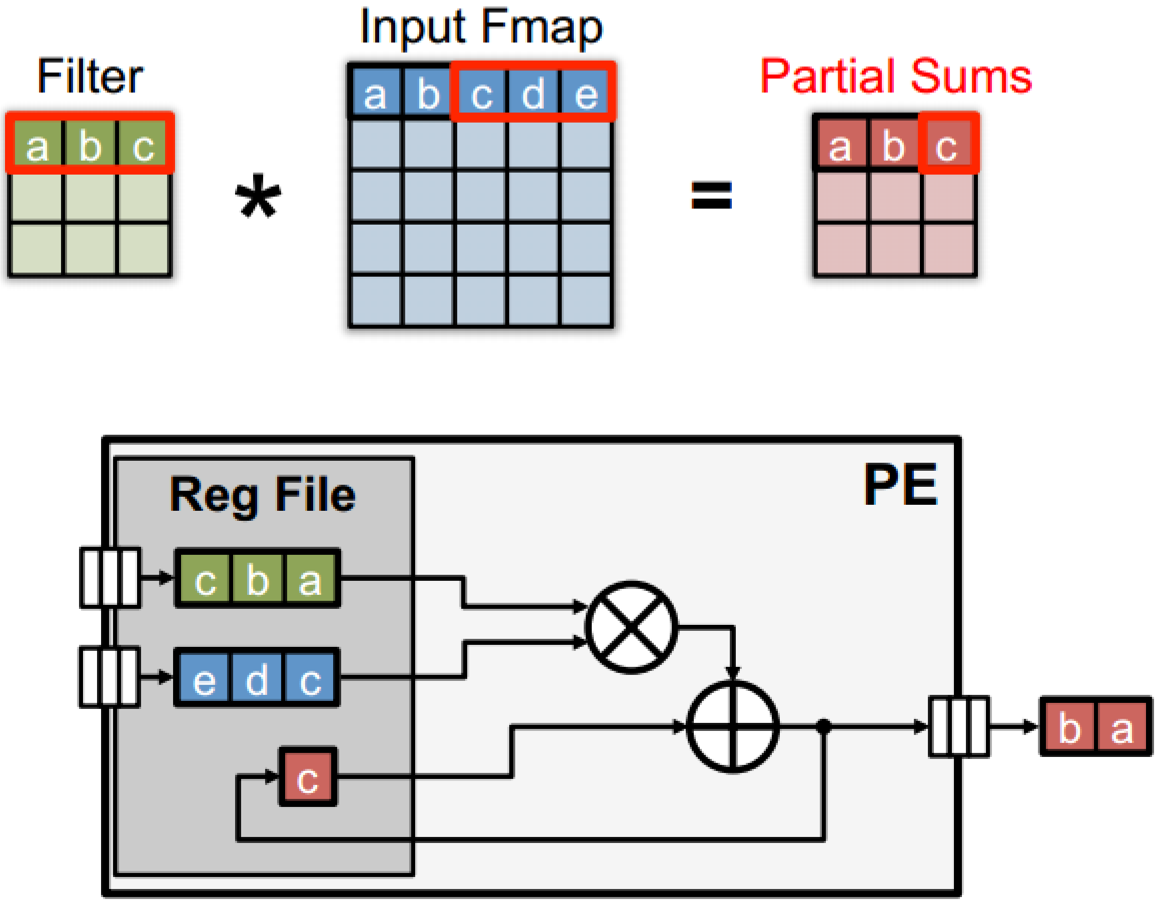
图片来源：http://eyeriss.mit.edu/tutorial.html#ISCA

# 1D Row Convolution in PE



图片来源：http://eyeriss.mit.edu/tutorial.html#ISCA

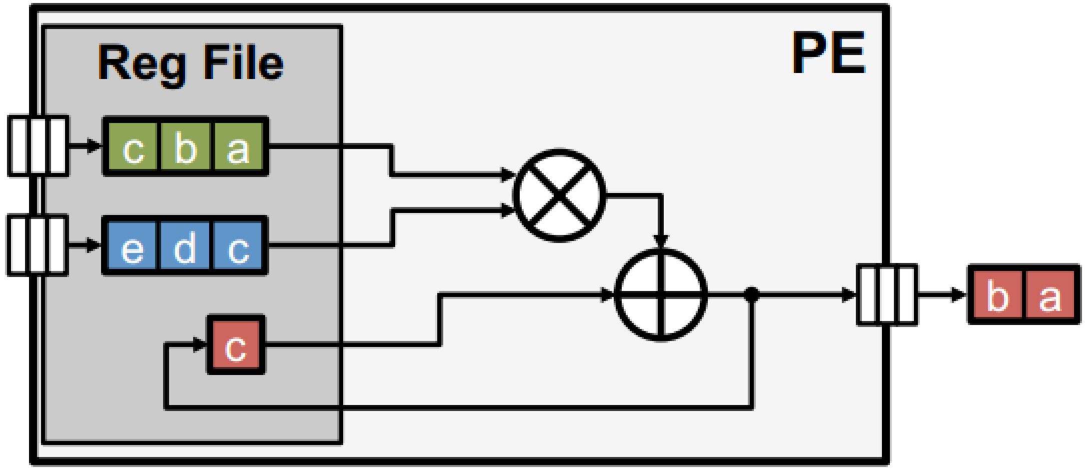
# 1D Row Convolution in PE



图片来源：http://eyeriss.mit.edu/tutorial.html#ISCA

# 1D Row Convolution in PE

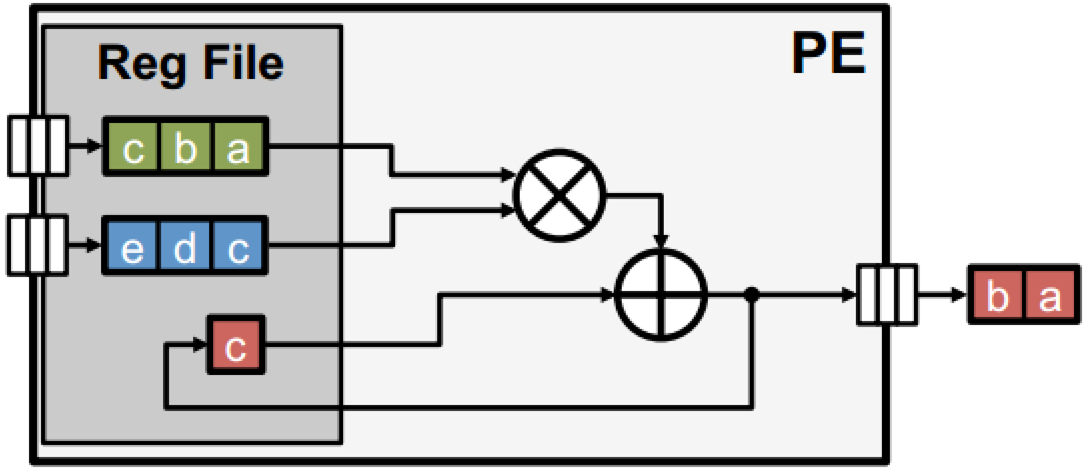
- Maximize row **convolutional reuse** in RF
  - Keep a **filter** row and **fmap** sliding window in RF
- Maximize row **psum accumulation** in RF



图片来源：http://eyeriss.mit.edu/tutorial.html#ISCA

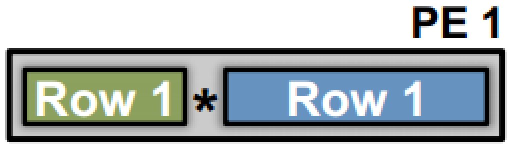
# 1D Row Convolution in PE

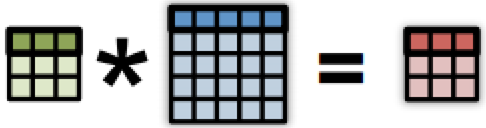
- Maximize row **convolutional reuse** in RF
  - Keep a **filter** row and **fmap** sliding window in RF
- Maximize row **psum accumulation** in RF



图片来源：http://eyeriss.mit.edu/tutorial.html#ISCA

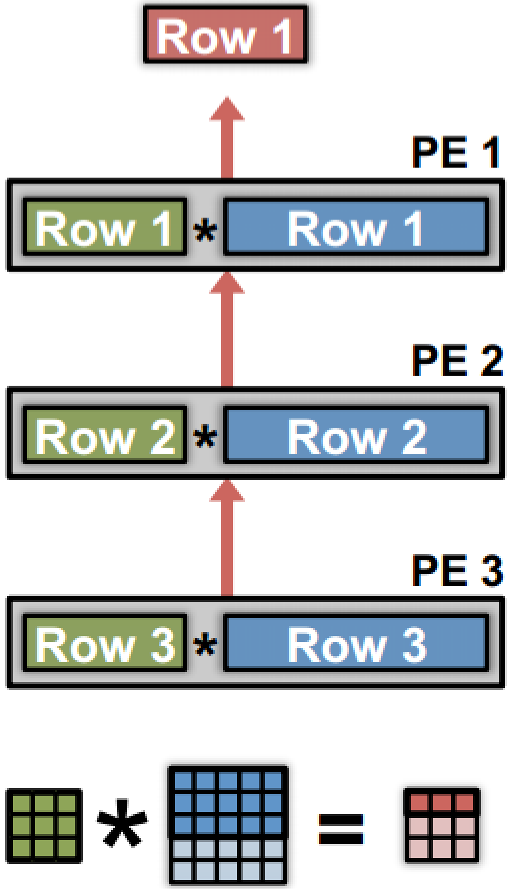
# 2D Convolution in PE Array





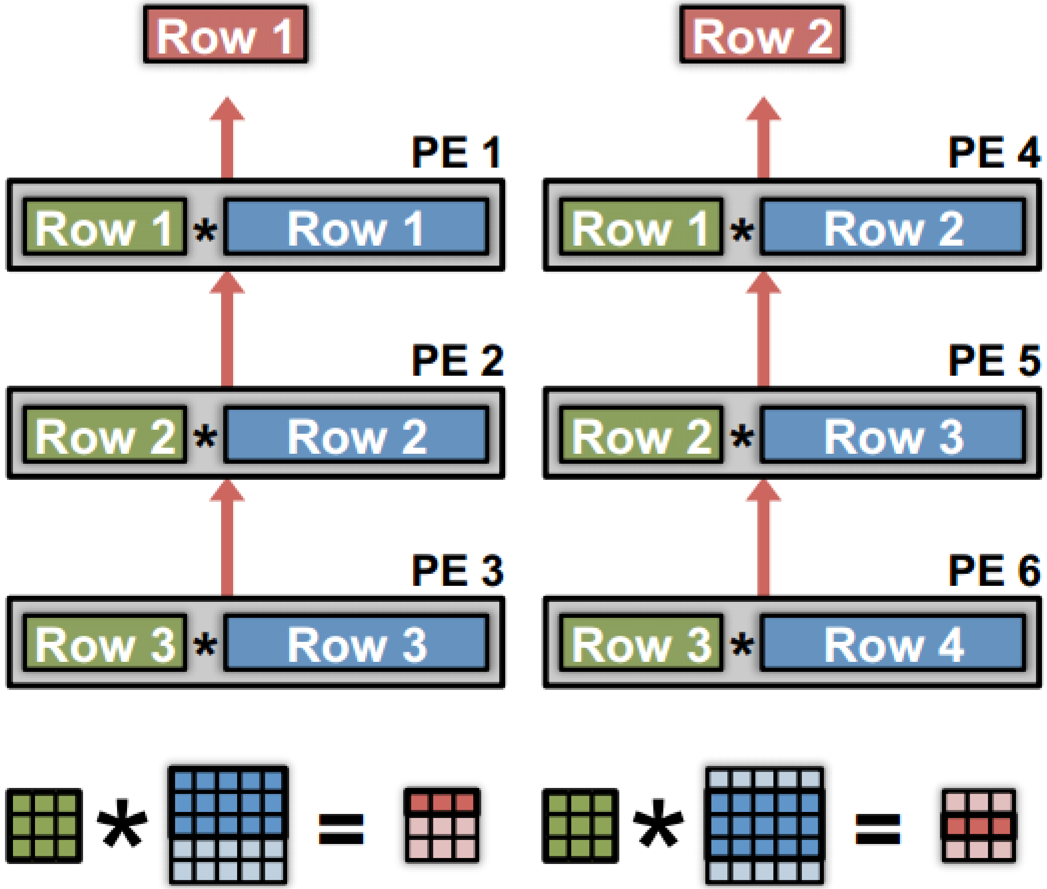
图片来源：http://eyeriss.mit.edu/tutorial.html#ISCA

# 2D Convolution in PE Array



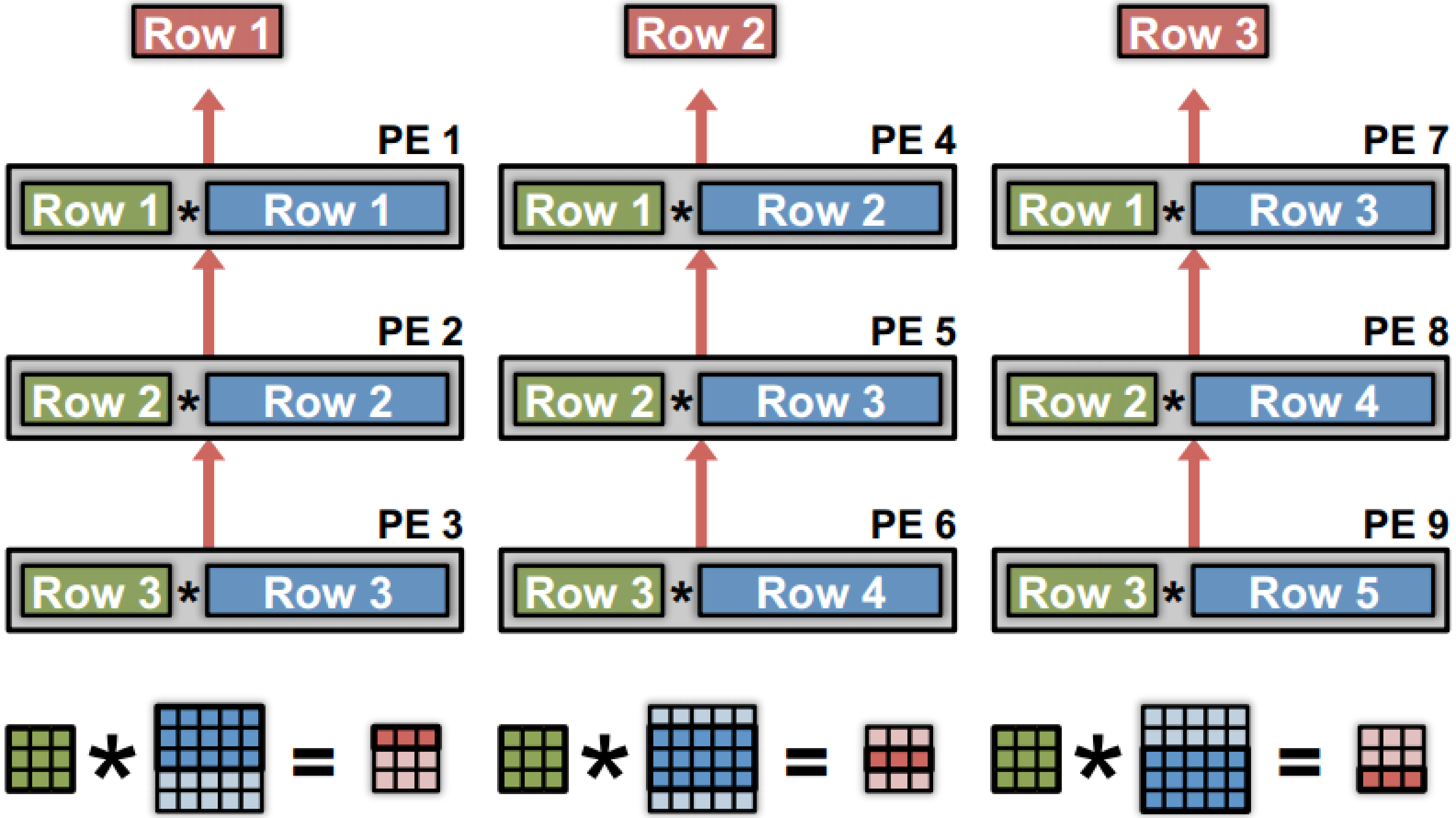
图片来源：http://eyeriss.mit.edu/tutorial.html#ISCA

# 2D Convolution in PE Array



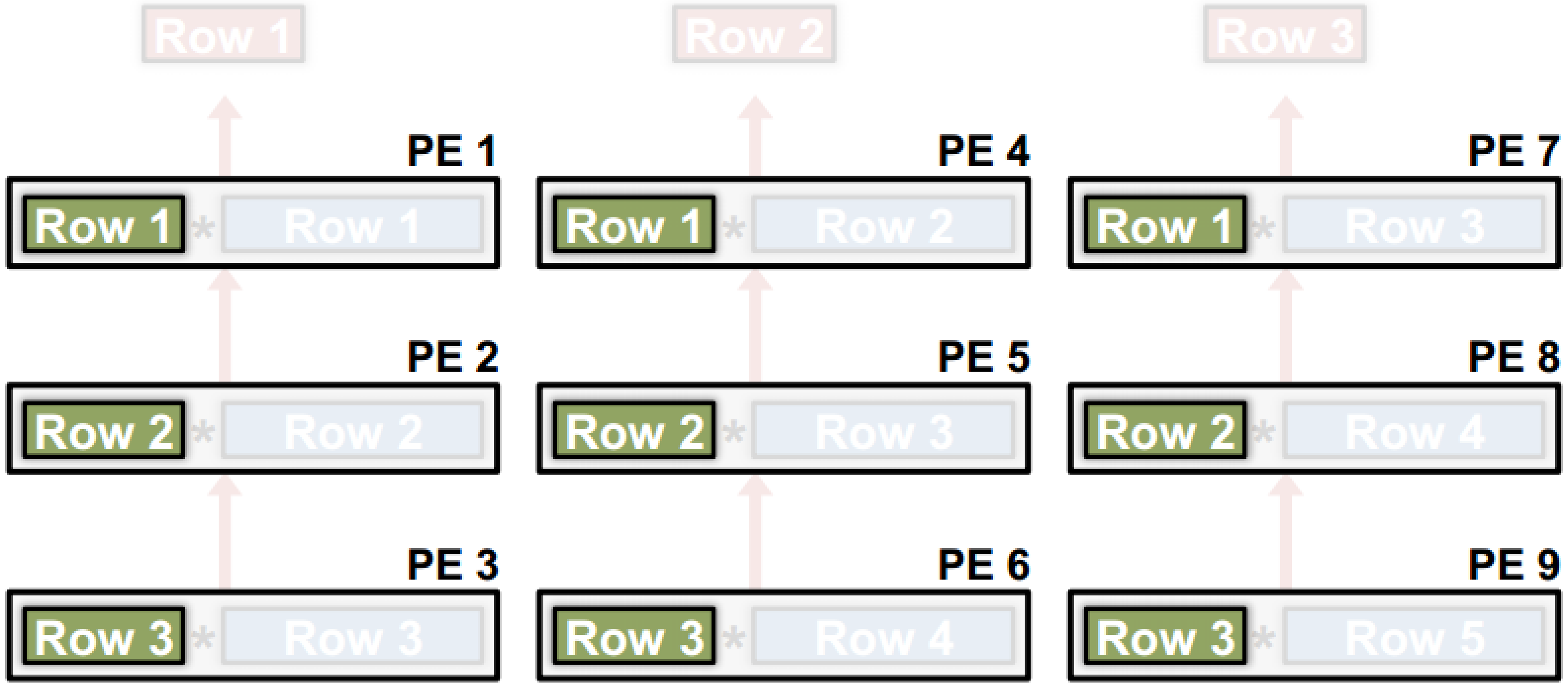
图片来源：http://eyeriss.mit.edu/tutorial.html#ISCA

# 2D Convolution in PE Array



图片来源：http://eyeriss.mit.edu/tutorial.html#ISCA

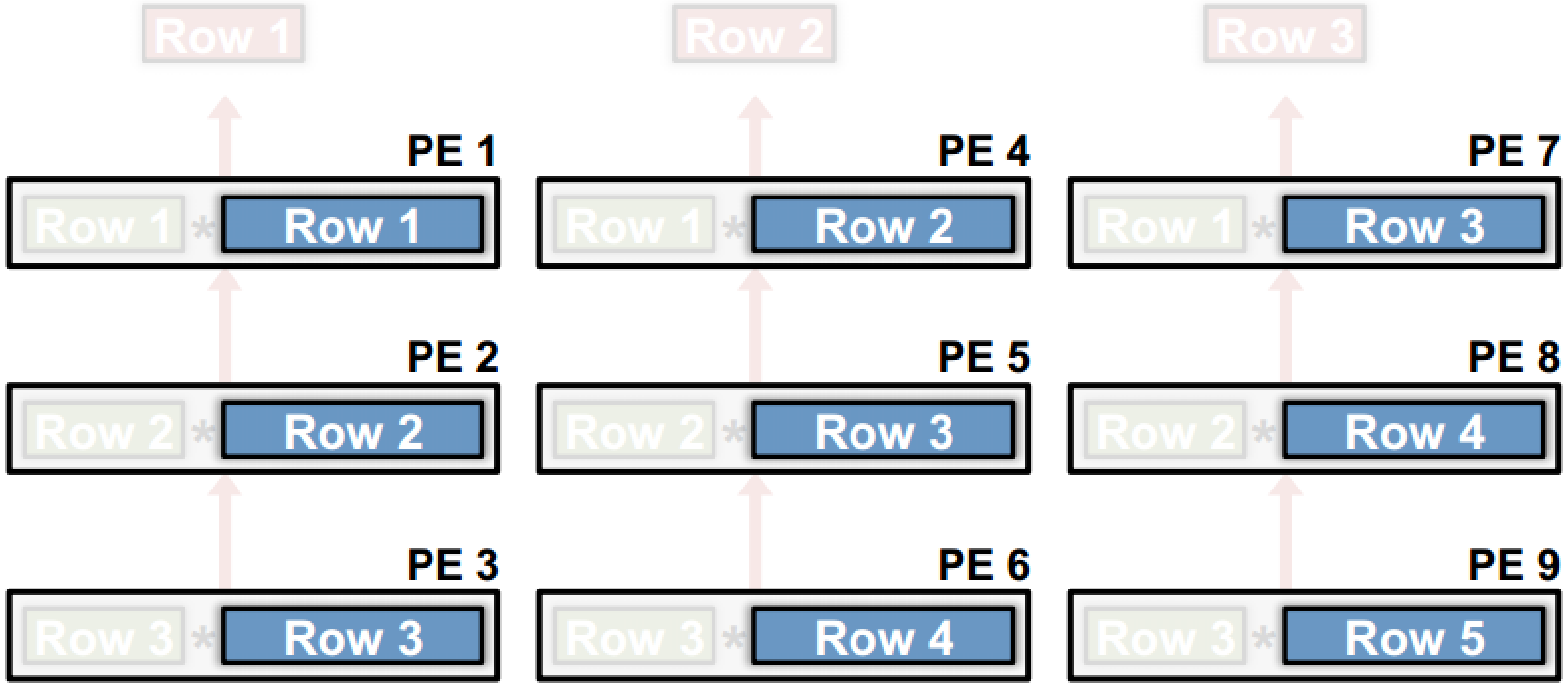
# Convolutional Reuse Maximized



**Filter rows** are reused across PEs **horizontally**

图片来源：http://eyeriss.mit.edu/tutorial.html#ISCA

# Convolutional Reuse Maximized



Row 1 Row 2 Row 3

PE 1: Row 1 * Row 1

PE 2: Row 2 * Row 2

PE 3: Row 3 * Row 3

PE 4: Row 1 * Row 2

PE 5: Row 2 * Row 3

PE 6: Row 3 * Row 4

PE 7: Row 1 * Row 3

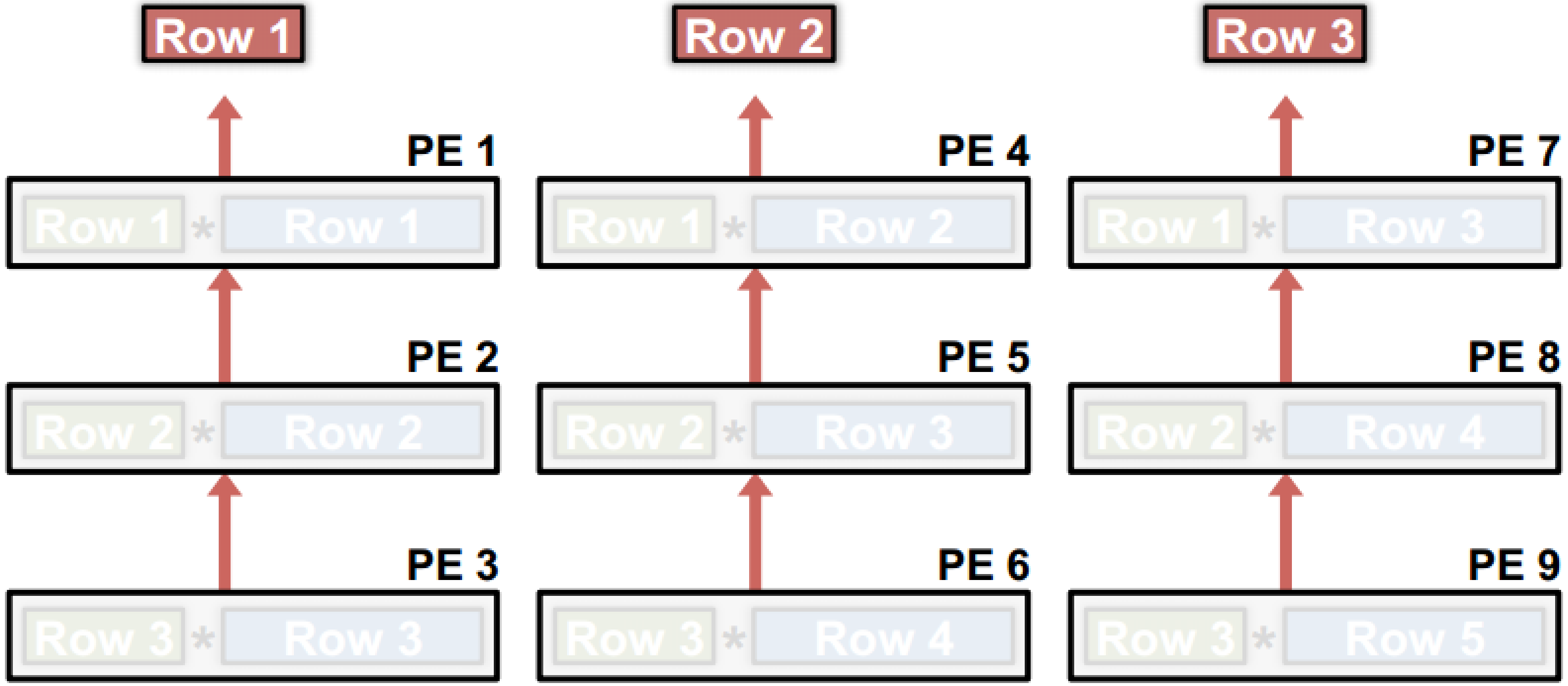
PE 8: Row 2 * Row 4

PE 9: Row 3 * Row 5

**Fmap rows** are reused across PEs **diagonally**

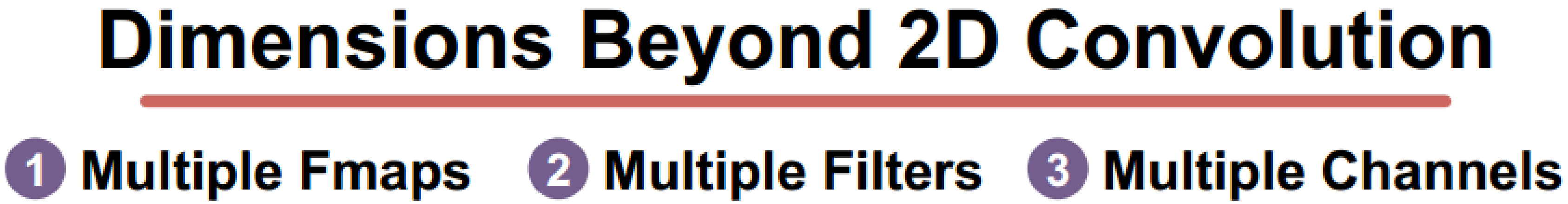
图片来源：http://eyeriss.mit.edu/tutorial.html#ISCA

# Maximize 2D Accumulation in PE Array



**Partial sums** accumulate across PEs **vertically**

图片来源：http://eyeriss.mit.edu/tutorial.html#ISCA



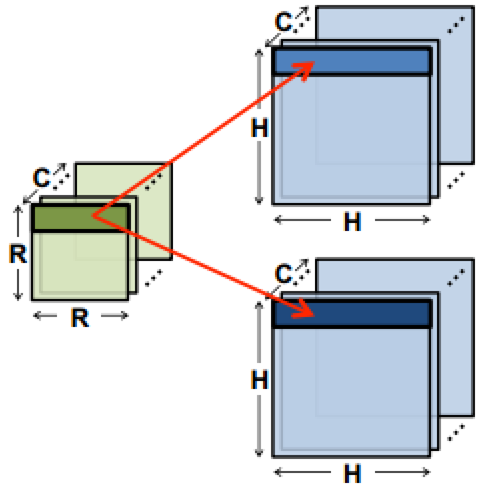
## Dimensions Beyond 2D Convolution

1. Multiple Fmaps
2. Multiple Filters
3. Multiple Channels

图片来源：http://eyeriss.mit.edu/tutorial.html#ISCA

# Filter Reuse in PE

1 Multiple Fmaps 2 Multiple Filters 3 Multiple Channels

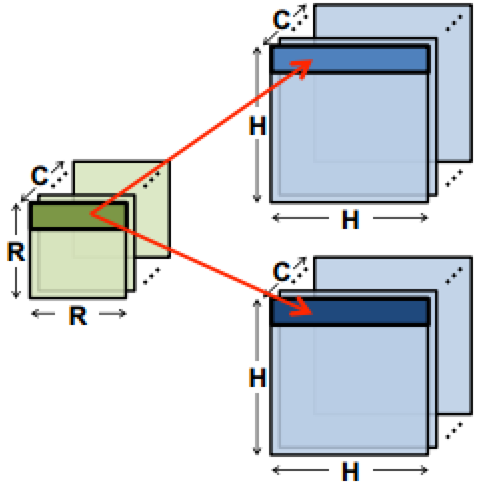


| | Filter 1 | | Fmap 1 | | Psum 1 |
|---|---|---|---|---|---|
| Channel 1 | Row 1 | * | Row 1 | = | Row 1 |
| | **Filter 1** | | **Fmap 2** | | **Psum 2** |
| Channel 1 | Row 1 | * | Row 1 | = | Row 1 |

图片来源：http://eyeriss.mit.edu/tutorial.html#ISCA

# Filter Reuse in PE

1 Multiple Fmaps 2 Multiple Filters 3 Multiple Channels



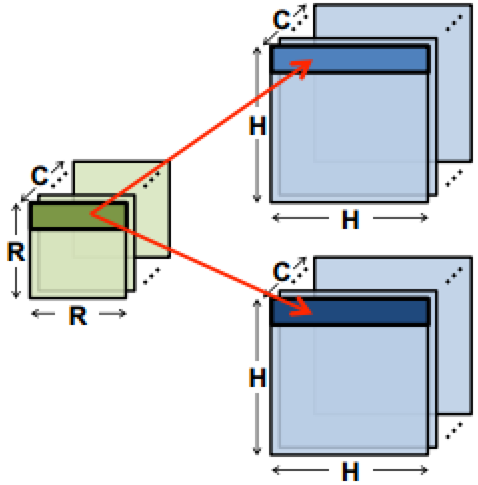
| | Filter 1 | | Fmap 1 | | Psum 1 |
|---|---|---|---|---|---|
| Channel 1 | Row 1 | * | Row 1 | = | Row 1 |
| | Filter 1 | | Fmap 2 | | Psum 2 |
| Channel 1 | Row 1 | * | Row 1 | = | Row 1 |

share the same filter row

图片来源：http://eyeriss.mit.edu/tutorial.html#ISCA

# Filter Reuse in PE

1 Multiple Fmaps 2 Multiple Filters 3 Multiple Channels



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

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

share the same filter row

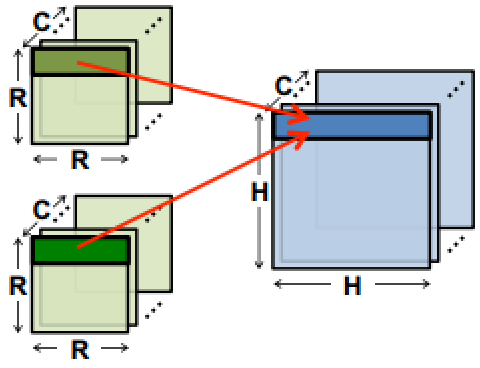
Processing in PE: **concatenate fmap rows**

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

图片来源：http://eyeriss.mit.edu/tutorial.html#ISCA

# Fmap Reuse in PE

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



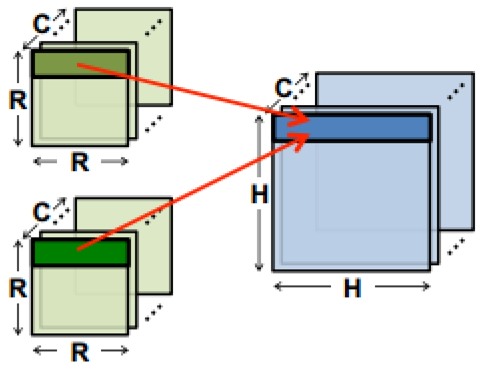
| | Filter 1 | | Fmap 1 | | Psum 1 |
|---|---|---|---|---|---|
| Channel 1 | Row 1 | * | Row 1 | = | Row 1 |

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

图片来源：http://eyeriss.mit.edu/tutorial.html#ISCA

# Fmap Reuse in PE

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



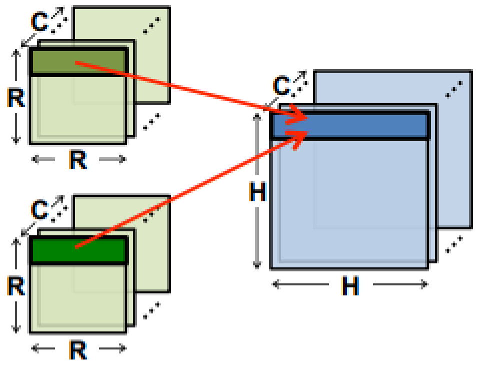
| | Filter 1 | | Fmap 1 | | Psum 1 |
|---|---|---|---|---|---|
| Channel 1 | Row 1 | * | Row 1 | = | Row 1 |
| | Filter 2 | | Fmap 1 | | Psum 2 |
| Channel 1 | Row 1 | * | Row 1 | = | Row 1 |

share the same fmap row

图片来源：http://eyeriss.mit.edu/tutorial.html#ISCA

# Fmap Reuse in PE

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

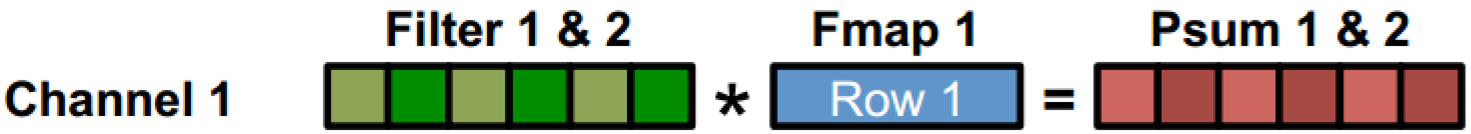


Channel 1  Filter 1 Row 1 * Fmap 1 Row 1 = Psum 1 Row 1

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

share the same fmap row

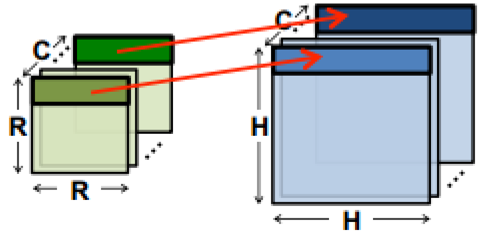
Processing in PE: **interleave filter rows**


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

图片来源：http://eyeriss.mit.edu/tutorial.html#ISCA

# Channel Accumulation in PE

1 Multiple Fmaps　2 Multiple Filters　3 Multiple Channels

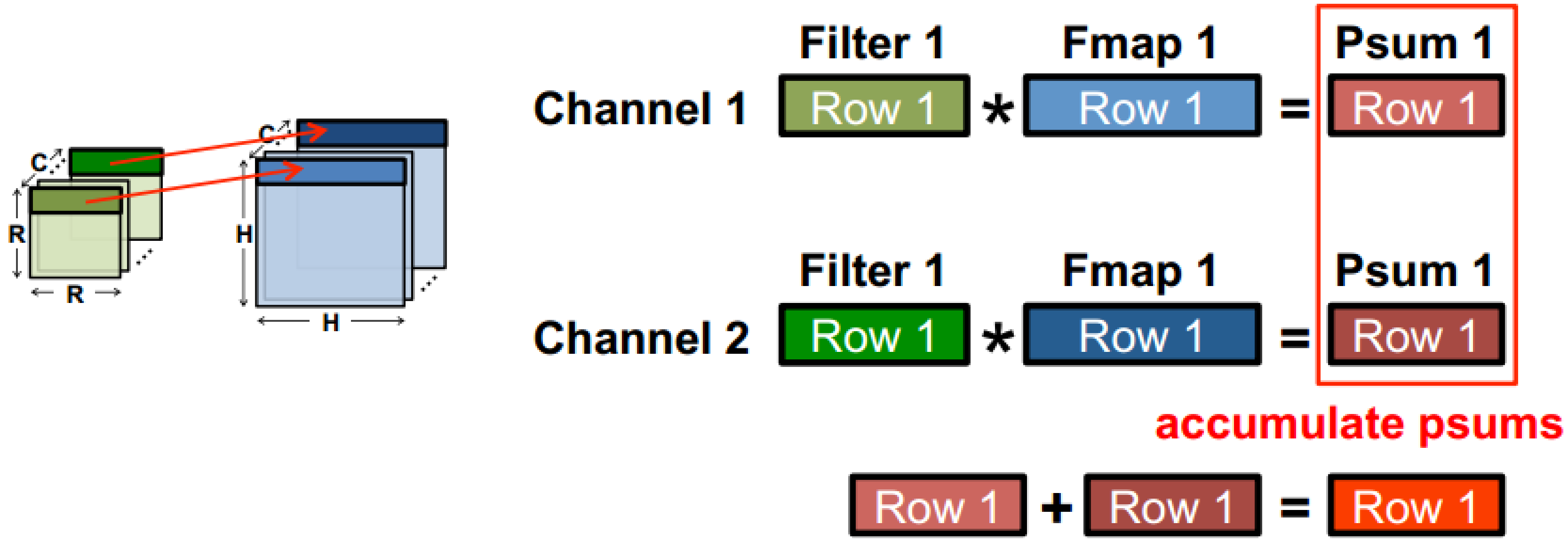


| | Filter 1 | | Fmap 1 | | Psum 1 |
|---|---|---|---|---|---|
| Channel 1 | Row 1 | * | Row 1 | = | Row 1 |
| Channel 2 | Row 1 | * | Row 1 | = | Row 1 |

图片来源：http://eyeriss.mit.edu/tutorial.html#ISCA

# 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 |
| Channel 2 | Row 1 | * | Row 1 | = | Row 1 |

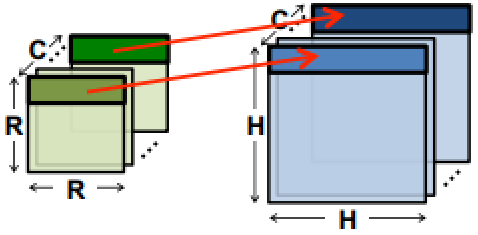
accumulate psums

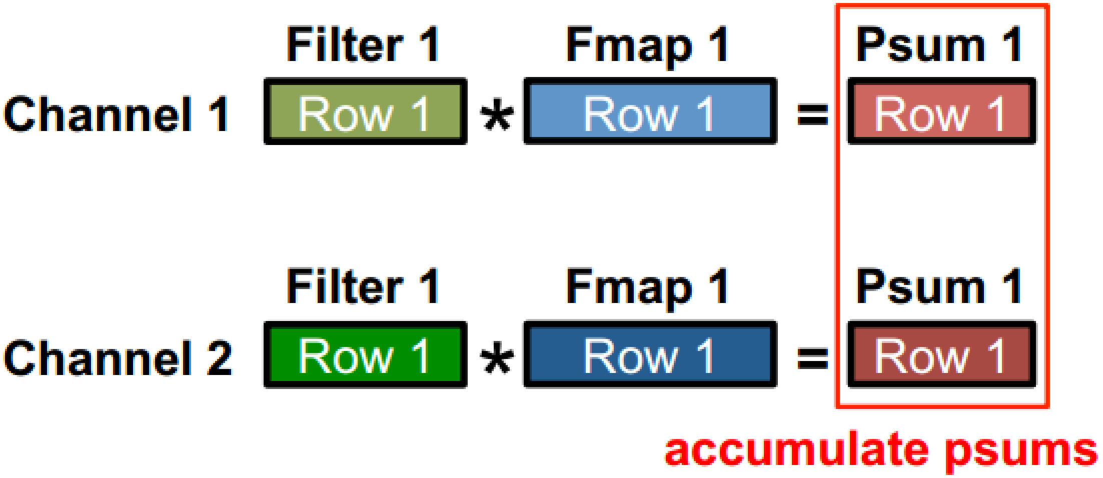
Row 1 + Row 1 = Row 1

图片来源：http://eyeriss.mit.edu/tutorial.html#ISCA

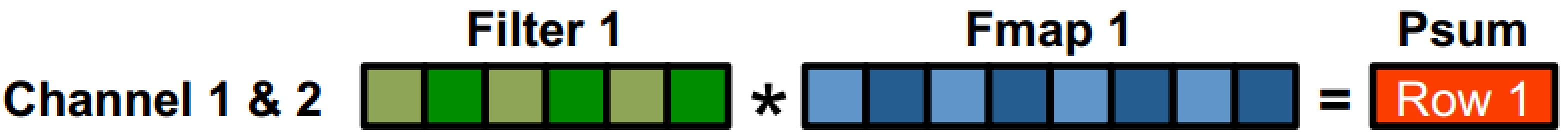
# Channel Accumulation in PE

1 Multiple Fmaps 2 Multiple Filters 3 Multiple Channels





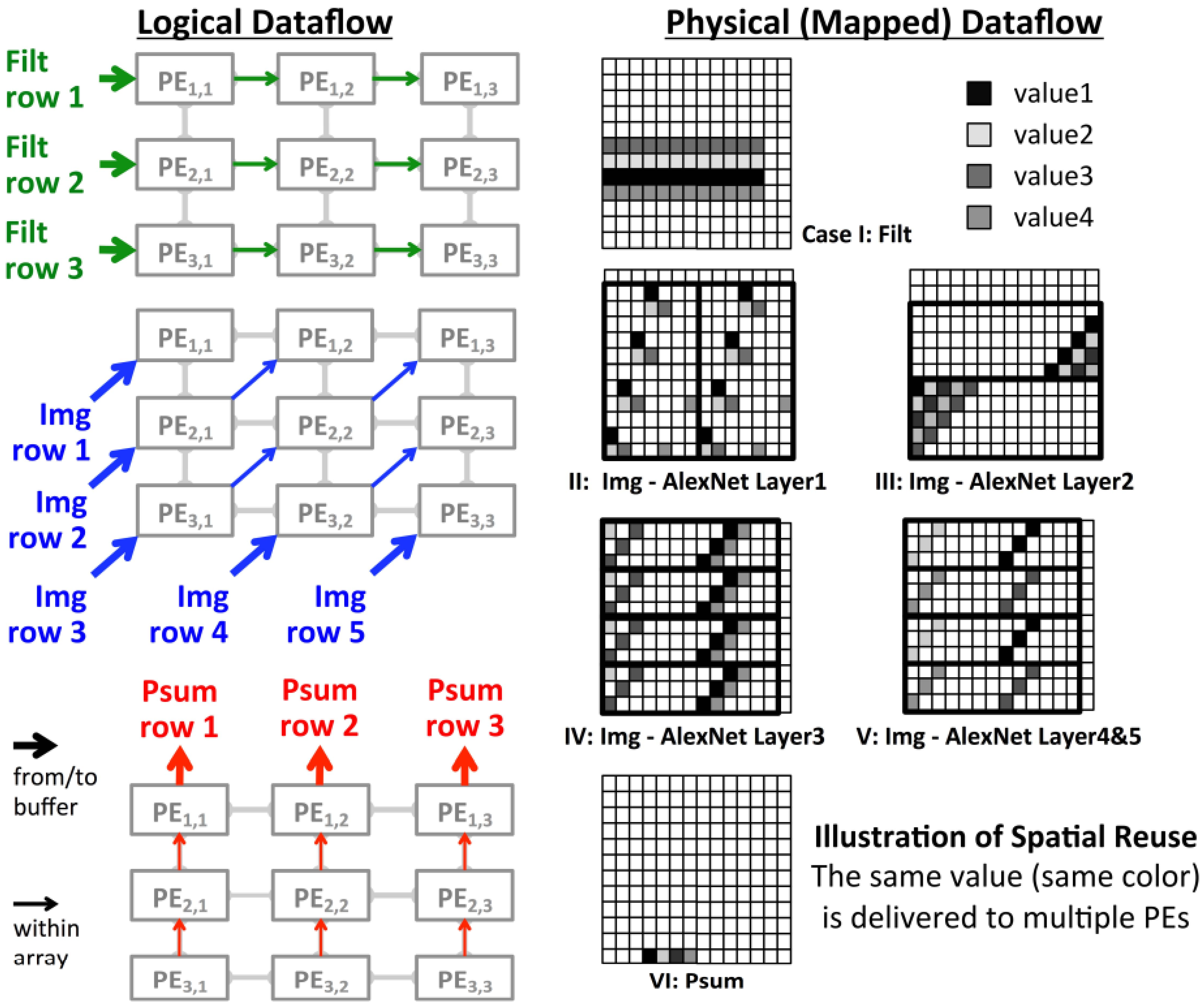
Processing in PE: **interleave channels**

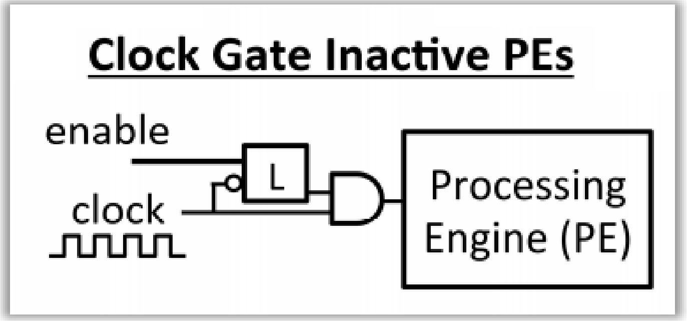


图片来源：http://eyeriss.mit.edu/tutorial.html#ISCA

# 架构上的改进

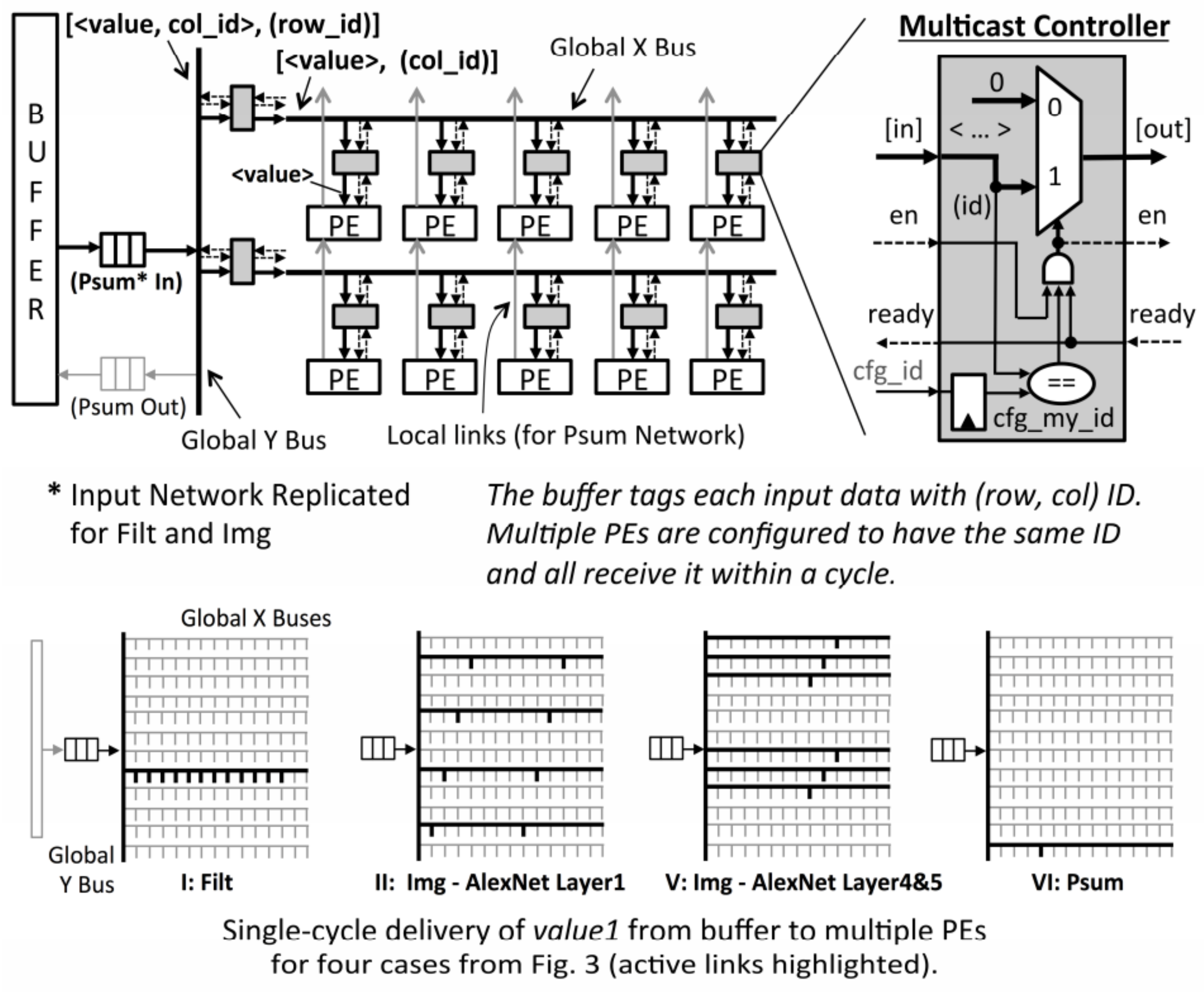
Opportunities in Architectures





# 架构上的改进

Opportunities in Architectures



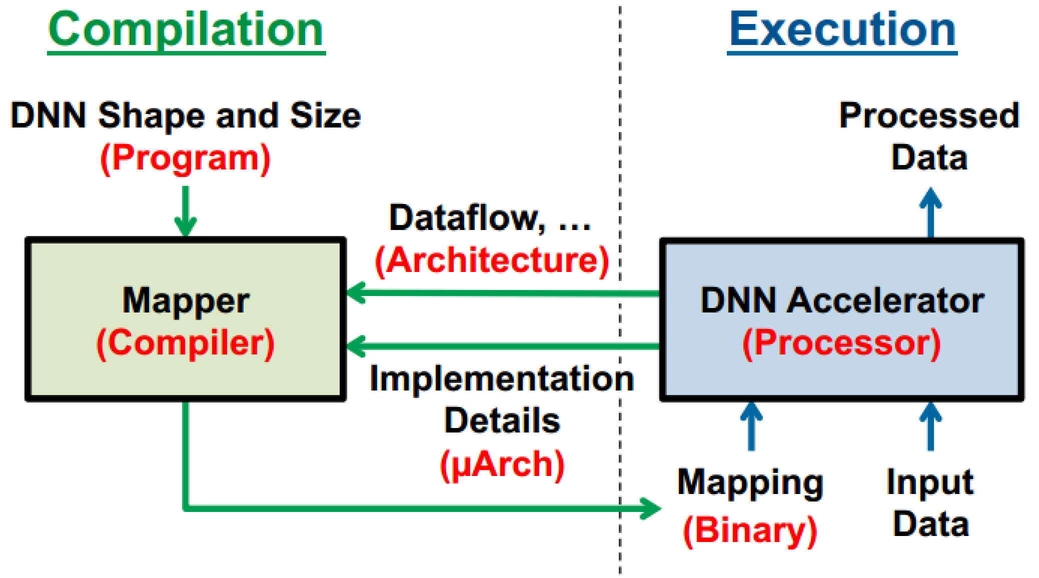
* Input Network Replicated for Filt and Img

*The buffer tags each input data with (row, col) ID. Multiple PEs are configured to have the same ID and all receive it within a cycle.*

Single-cycle delivery of *value1* from buffer to multiple PEs for four cases from Fig. 3 (active links highlighted).

**Figure 14.5.4: Network-on-Chip (NoC) for multicasting**

# Computer Architecture Analogy



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

来源：http://eyeriss.mit.edu/tutorial.html#ISCA

03

PART THREE

# 具体的设计

# 具体的设计

以 Eyeriss 为实例

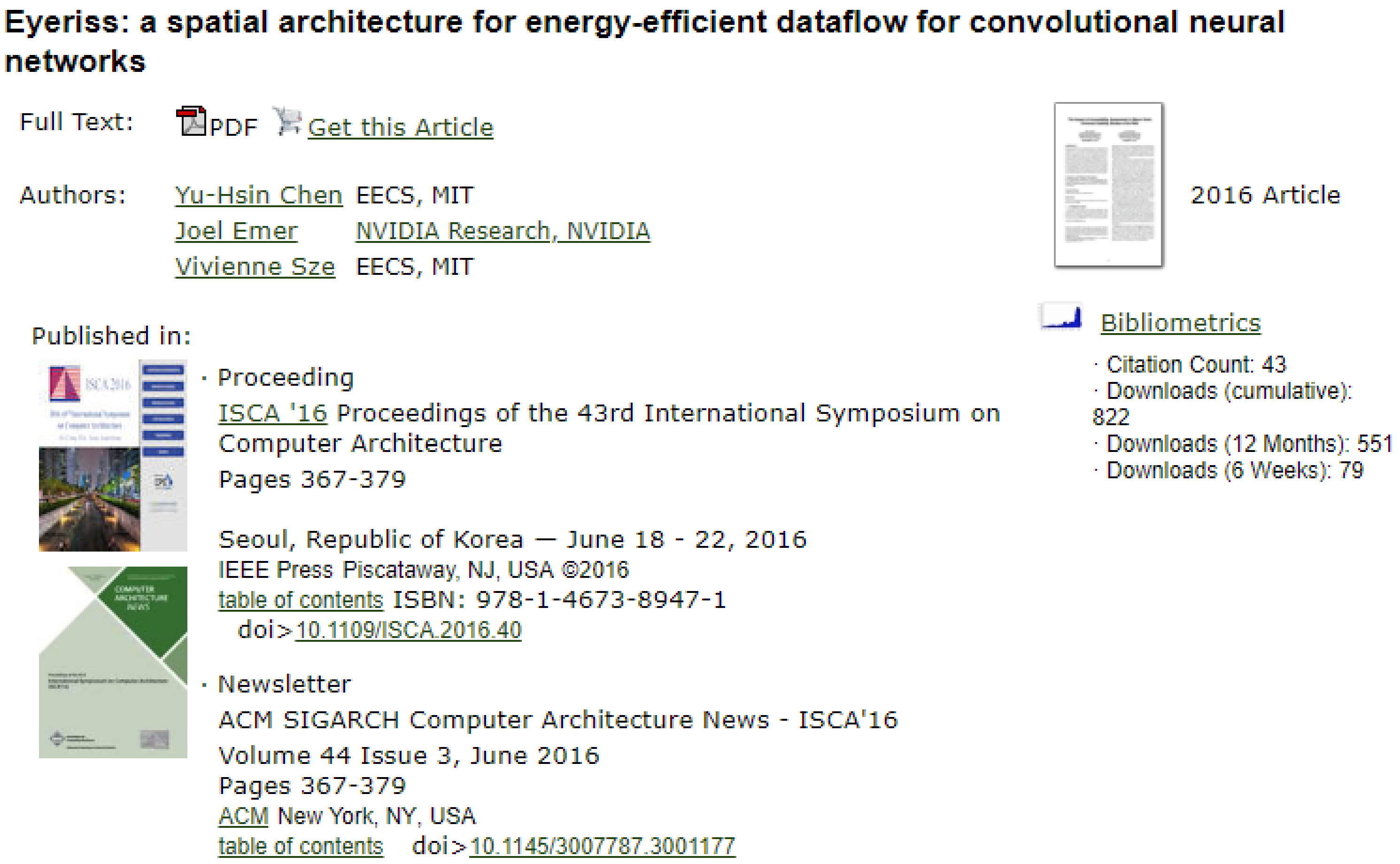
## Eyeriss芯片

采用了Row Stationary技术

（RS是Eyeriss的一个关键特性）

此外还有

- Reduce Precision
- Sparsity
- Compression



**Eyeriss: a spatial architecture for energy-efficient dataflow for convolutional neural networks**

Full Text: PDF Get this Article

2016 Article

Authors: Yu-Hsin Chen EECS, MIT
Joel Emer NVIDIA Research, NVIDIA
Vivienne Sze EECS, MIT

Bibliometrics
· Citation Count: 43
· Downloads (cumulative): 822
· Downloads (12 Months): 551
· Downloads (6 Weeks): 79

Published in:

· Proceeding
ISCA '16 Proceedings of the 43rd International Symposium on Computer Architecture
Pages 367-379

Seoul, Republic of Korea — June 18 - 22, 2016
IEEE Press Piscataway, NJ, USA ©2016
table of contents ISBN: 978-1-4673-8947-1
doi>10.1109/ISCA.2016.40

· Newsletter
ACM SIGARCH Computer Architecture News - ISCA'16
Volume 44 Issue 3, June 2016
Pages 367-379
ACM New York, NY, USA
table of contents doi>10.1145/3007787.3001177

03

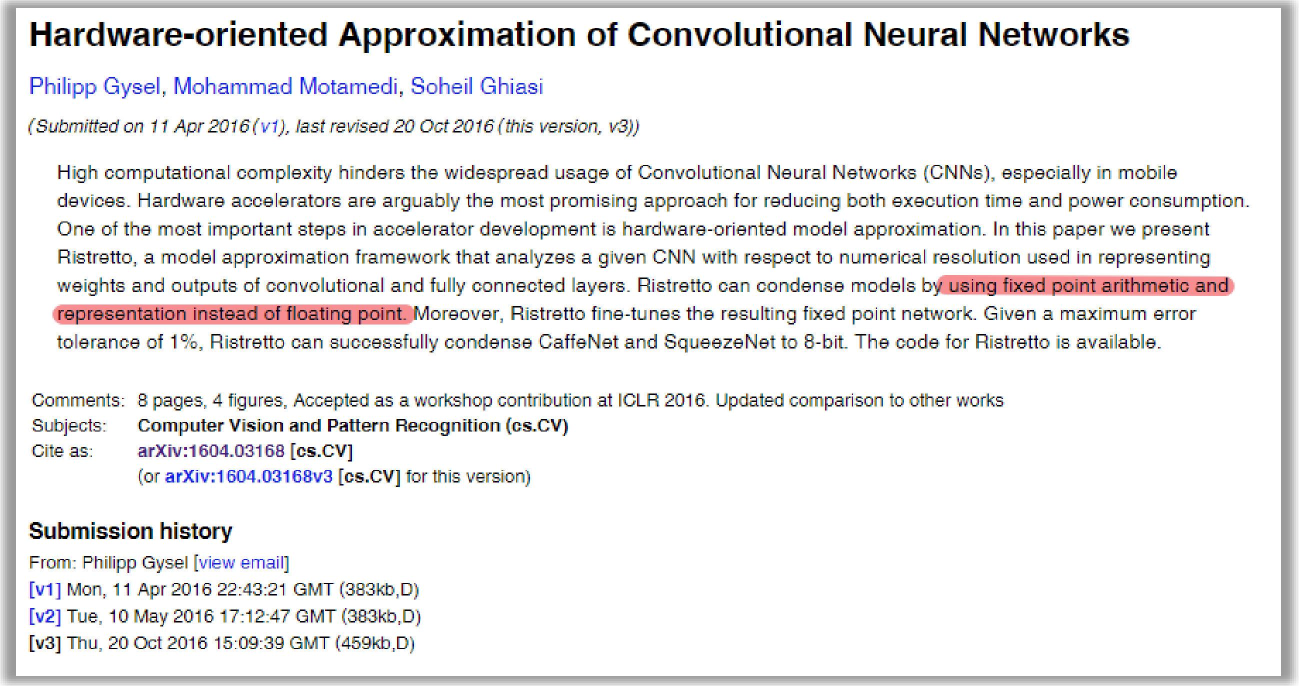
# 具体的设计

以 Eyeriss 为实例

## Reduce Precision

通常情况下，CPU和GPU都使用32位或64位的浮点数进行运算

Eyeriss支持16位的定点数，与此类似的有Google的TPU支持8位的定点数运算



### Hardware-oriented Approximation of Convolutional Neural Networks

Philipp Gysel, Mohammad Motamedi, Soheil Ghiasi

*(Submitted on 11 Apr 2016 (v1), last revised 20 Oct 2016 (this version, v3))*

High computational complexity hinders the widespread usage of Convolutional Neural Networks (CNNs), especially in mobile devices. Hardware accelerators are arguably the most promising approach for reducing both execution time and power consumption. One of the most important steps in accelerator development is hardware-oriented model approximation. In this paper we present Ristretto, a model approximation framework that analyzes a given CNN with respect to numerical resolution used in representing weights and outputs of convolutional and fully connected layers. Ristretto can condense models by using fixed point arithmetic and representation instead of floating point. Moreover, Ristretto fine-tunes the resulting fixed point network. Given a maximum error tolerance of 1%, Ristretto can successfully condense CaffeNet and SqueezeNet to 8-bit. The code for Ristretto is available.

| | |
|---|---|
| Comments: | 8 pages, 4 figures, Accepted as a workshop contribution at ICLR 2016. Updated comparison to other works |
| Subjects: | **Computer Vision and Pattern Recognition (cs.CV)** |
| Cite as: | arXiv:1604.03168 [cs.CV] |
| | (or arXiv:1604.03168v3 [cs.CV] for this version) |

**Submission history**

From: Philipp Gysel [view email]
[v1] Mon, 11 Apr 2016 22:43:21 GMT (383kb,D)
[v2] Tue, 10 May 2016 17:12:47 GMT (383kb,D)
[v3] Thu, 20 Oct 2016 15:09:39 GMT (459kb,D)

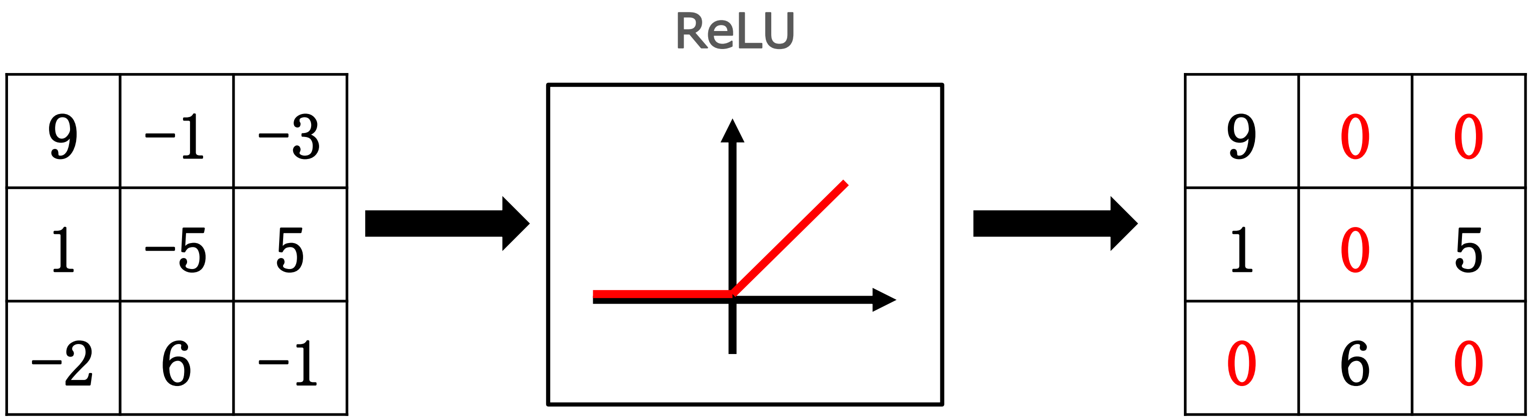
# 03 具体的设计

以 Eyeriss 为实例

## Sparsity

数据本身可能是稀疏的、训练得到的权重也可能是稀疏的

例：SVM中的权重投影、DNN中的修剪权重、图像处理中的特征提取



ReLU

| | | |
|---|---|---|
| 9 | −1 | −3 |
| 1 | −5 | 5 |
| −2 | 6 | −1 |

| | | |
|---|---|---|
| 9 | 0 | 0 |
| 1 | 0 | 5 |
| 0 | 6 | 0 |

# 03 具体的设计

以 Eyeriss 为实例

## Sparsity

Eyeriss芯片在硬件层面上利用了稀疏性来提高速度和降低能耗

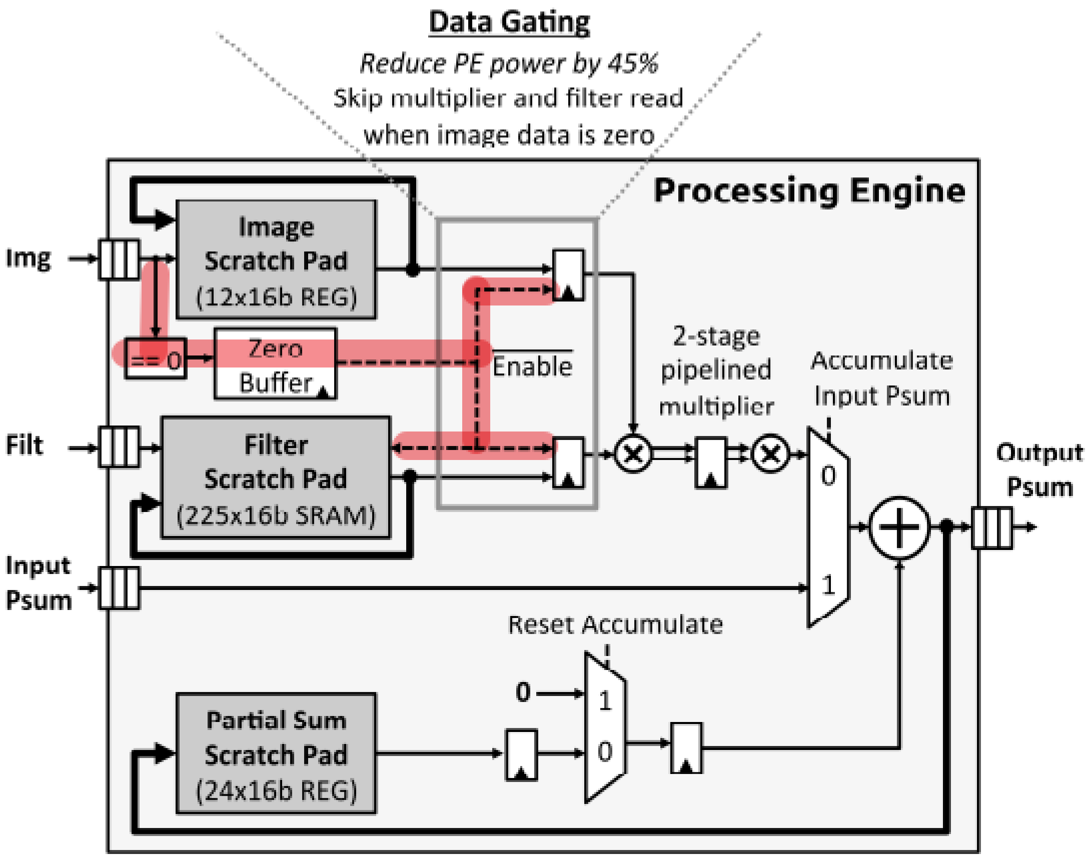
在处理有关零值参与的乘法运算时，Eyeriss使用了skip的方法：

遇到零则

跳过乘法计算（从图上看，实际是计算的0*0）

跳过weight参数的读取

这可以节省PE的功耗**45%**



# 03 具体的设计

以 Eyeriss 为实例

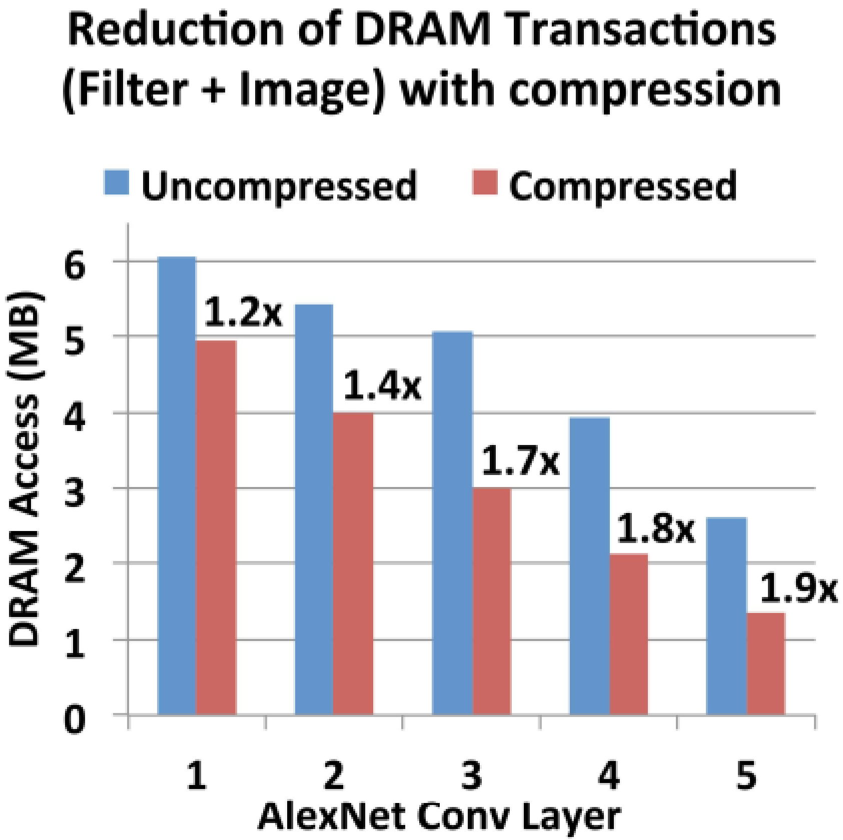
## Compression

Eyeriss芯片上采用压缩算法对数据压缩（利用稀疏性）

将稀疏的数据压缩后再存储到DRAM内

考虑到DRAM访问的功耗不小，也算是降低了功耗

例：AlexNet的五个Conv层的数据见右图

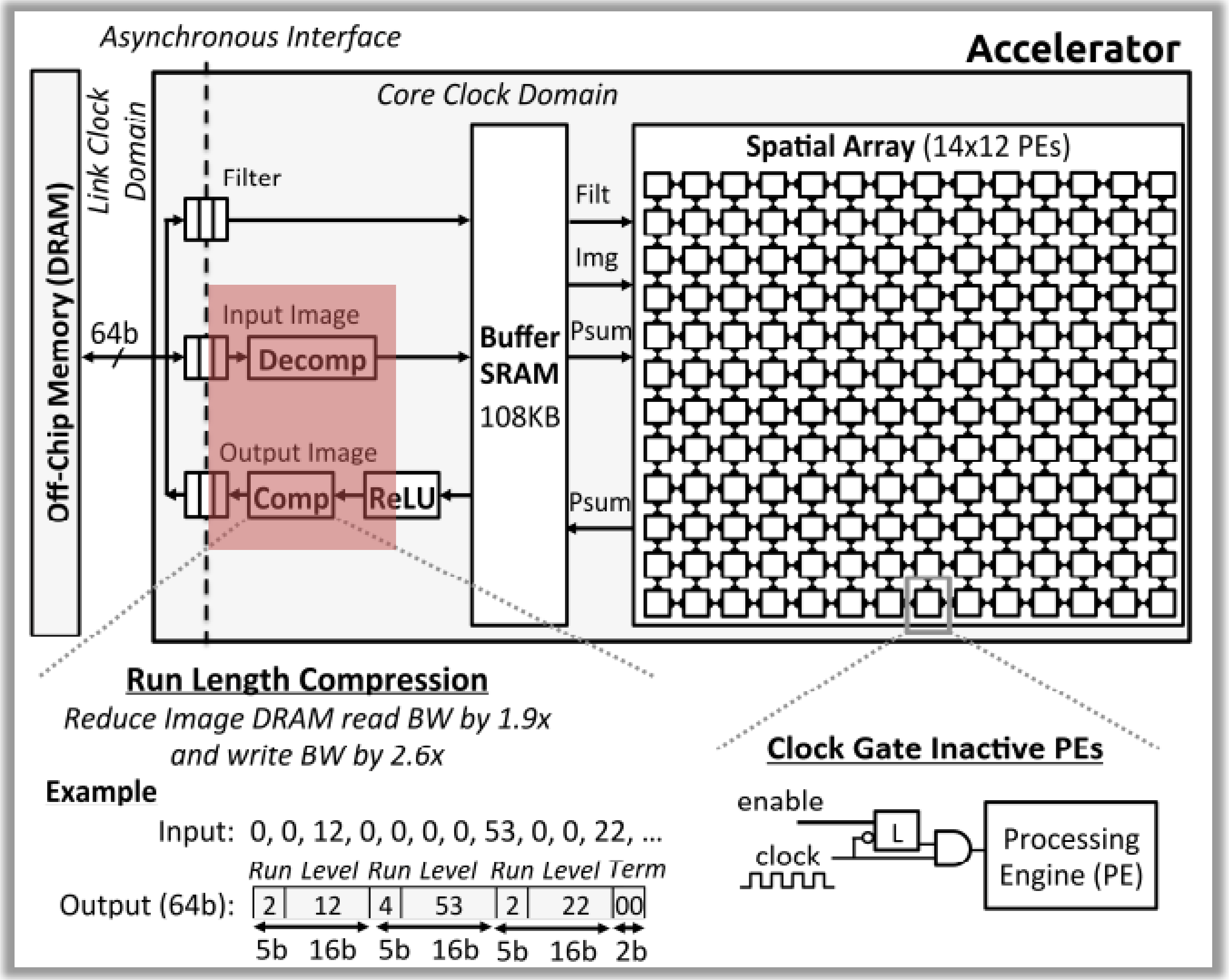


# 03 具体的设计

以 Eyeriss 为实例

## Run Length Compression

例:
Input: "aaaaaaabbccccdefffffffgg"
Output:"a7b2c4d1e1f7g2"



**Run Length Compression**

*Reduce Image DRAM read BW by 1.9x and write BW by 2.6x*

**Example**

Input: 0, 0, 12, 0, 0, 0, 0, 53, 0, 0, 22, …

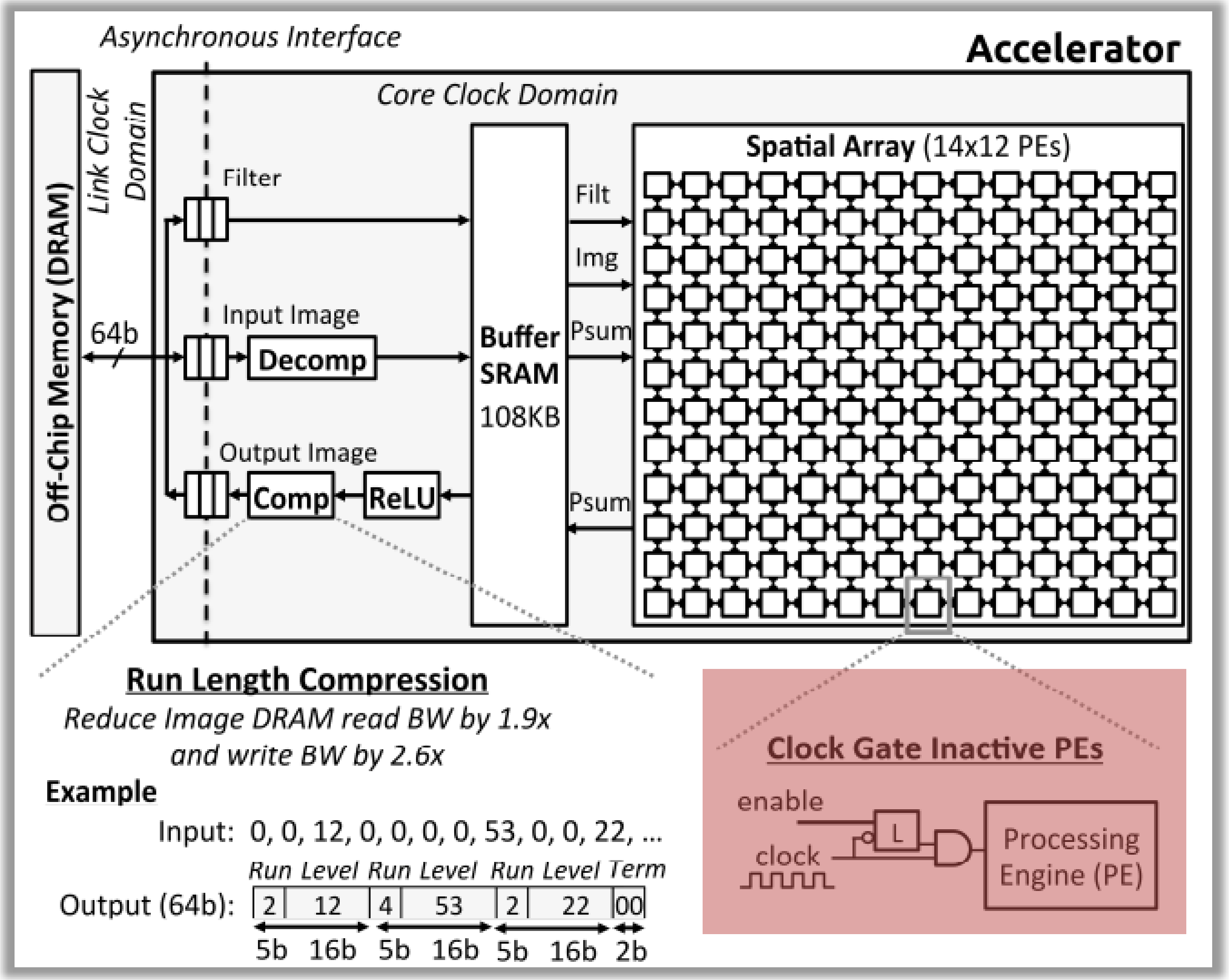
Output (64b):

| *Run* | *Level* | *Run* | *Level* | *Run* | *Level* | *Term* |
|---|---|---|---|---|---|---|
| 2 | 12 | 4 | 53 | 2 | 22 | 00 |
| 5b | 16b | 5b | 16b | 5b | 16b | 2b |

03

# 具体的设计

以 Eyeriss 为实例



**Run Length Compression**

*Reduce Image DRAM read BW by 1.9x and write BW by 2.6x*

**Example**

Input: 0, 0, 12, 0, 0, 0, 0, 53, 0, 0, 22, ...

Run Level Run Level Run Level Term
Output (64b): 2 12 4 53 2 22 00
5b 16b 5b 16b 5b 16b 2b

# 03 具体的设计

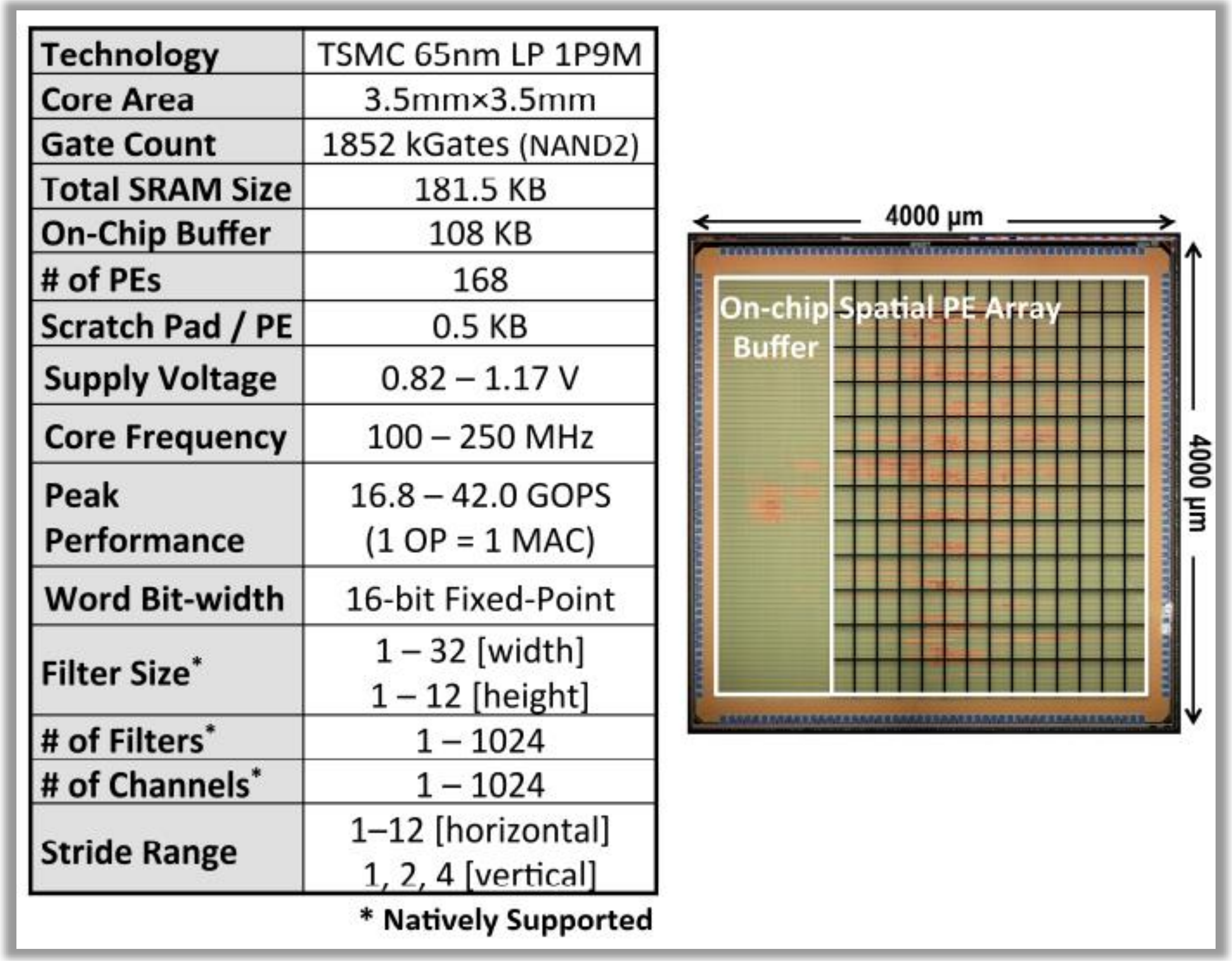
以 Eyeriss 为实例

## Eyeriss芯片

采用了 65nm CMOS 工艺

RF size/PE：0.5KB

On-chip Buffer Size：108KB



| | |
|---|---|
| Technology | TSMC 65nm LP 1P9M |
| Core Area | 3.5mm×3.5mm |
| Gate Count | 1852 kGates (NAND2) |
| Total SRAM Size | 181.5 KB |
| On-Chip Buffer | 108 KB |
| # of PEs | 168 |
| Scratch Pad / PE | 0.5 KB |
| Supply Voltage | 0.82 – 1.17 V |
| Core Frequency | 100 – 250 MHz |
| Peak Performance | 16.8 – 42.0 GOPS<br>(1 OP = 1 MAC) |
| Word Bit-width | 16-bit Fixed-Point |
| Filter Size* | 1 – 32 [width]<br>1 – 12 [height] |
| # of Filters* | 1 – 1024 |
| # of Channels* | 1 – 1024 |
| Stride Range | 1–12 [horizontal]<br>1, 2, 4 [vertical] |

* Natively Supported

Die photo and spec

04

PART FOUR

# 总结

## Eyeriss芯片

主要思想：

减少计算（部分和的重用、精度的降低即降低bitwidth）

减少数据移动（利用RS数据流、利用Sparsity减少零元素的读取、采用压缩技术减少数据传输量）

好处：

降低能耗使得移动终端设备续航时间长

计算能力增强使得在移动终端设备处理机器学习任务成为可能，降低了对服务器的依赖，减少了延迟

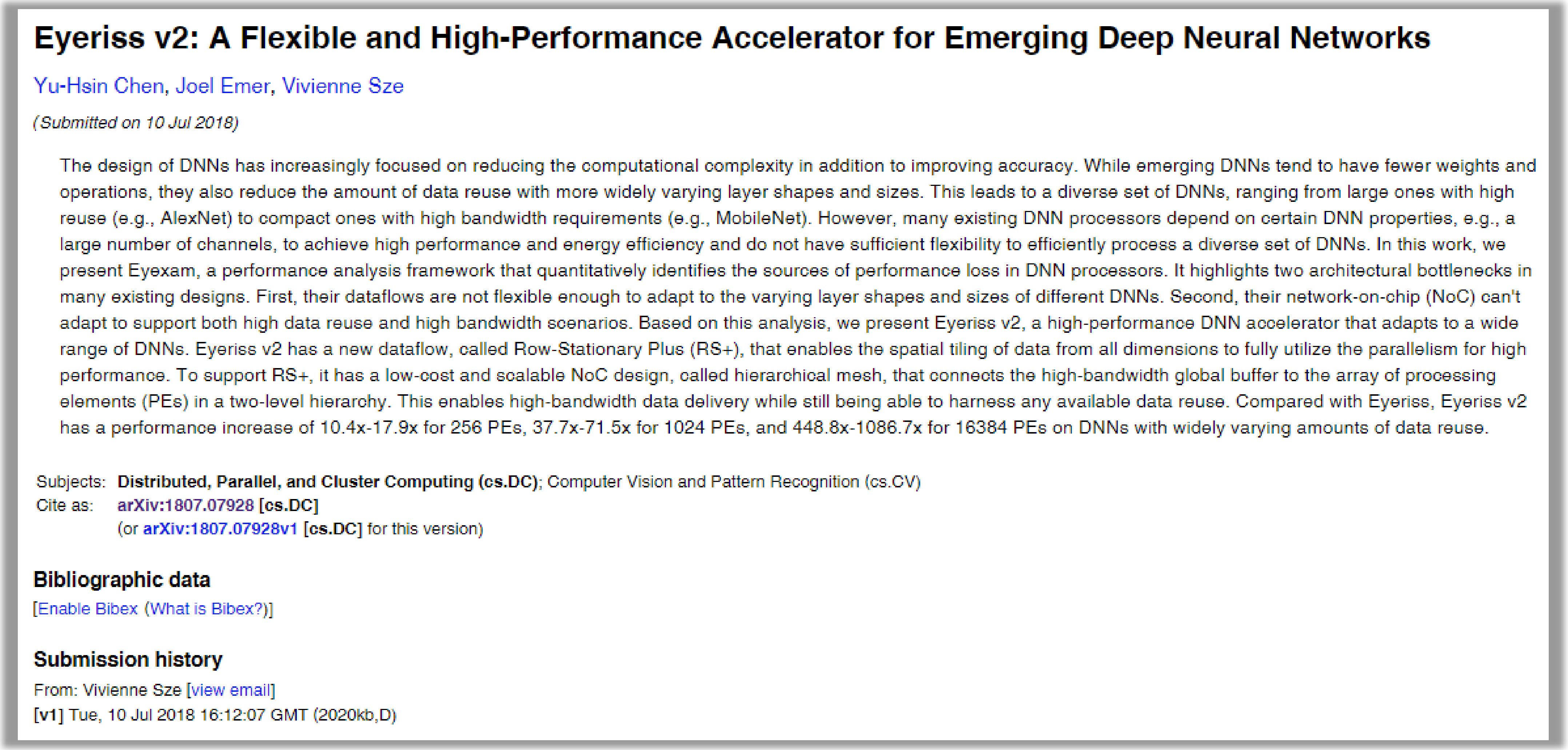
注：MIT的这支研究团队由该校电气工程与计算机科学系的Vivienne Sze带领，并在旧金山召开的国际固态电路会议（ISSCC 2016）上介绍了这款芯片，并演示了图像识别任务。

04 总结

Summary

大佬又出文章了！

Eyeriss v2：Row Stationary Plus（RS+）！



## Eyeriss v2: A Flexible and High-Performance Accelerator for Emerging Deep Neural Networks

Yu-Hsin Chen, Joel Emer, Vivienne Sze

*(Submitted on 10 Jul 2018)*

The design of DNNs has increasingly focused on reducing the computational complexity in addition to improving accuracy. While emerging DNNs tend to have fewer weights and operations, they also reduce the amount of data reuse with more widely varying layer shapes and sizes. This leads to a diverse set of DNNs, ranging from large ones with high reuse (e.g., AlexNet) to compact ones with high bandwidth requirements (e.g., MobileNet). However, many existing DNN processors depend on certain DNN properties, e.g., a large number of channels, to achieve high performance and energy efficiency and do not have sufficient flexibility to efficiently process a diverse set of DNNs. In this work, we present Eyexam, a performance analysis framework that quantitatively identifies the sources of performance loss in DNN processors. It highlights two architectural bottlenecks in many existing designs. First, their dataflows are not flexible enough to adapt to the varying layer shapes and sizes of different DNNs. Second, their network-on-chip (NoC) can't adapt to support both high data reuse and high bandwidth scenarios. Based on this analysis, we present Eyeriss v2, a high-performance DNN accelerator that adapts to a wide range of DNNs. Eyeriss v2 has a new dataflow, called Row-Stationary Plus (RS+), that enables the spatial tiling of data from all dimensions to fully utilize the parallelism for high performance. To support RS+, it has a low-cost and scalable NoC design, called hierarchical mesh, that connects the high-bandwidth global buffer to the array of processing elements (PEs) in a two-level hierarchy. This enables high-bandwidth data delivery while still being able to harness any available data reuse. Compared with Eyeriss, Eyeriss v2 has a performance increase of 10.4x-17.9x for 256 PEs, 37.7x-71.5x for 1024 PEs, and 448.8x-1086.7x for 16384 PEs on DNNs with widely varying amounts of data reuse.

Subjects: **Distributed, Parallel, and Cluster Computing (cs.DC)**; Computer Vision and Pattern Recognition (cs.CV)
Cite as: **arXiv:1807.07928 [cs.DC]**
(or **arXiv:1807.07928v1** [**cs.DC**] for this version)

### Bibliographic data

[Enable Bibex (What is Bibex?)]

### Submission history

From: Vivienne Sze [view email]
**[v1]** Tue, 10 Jul 2018 16:12:07 GMT (2020kb,D)

# Hardware for Machine Learning

Challenges and Opportunities

THANK YOU FOR LISTENING