北京大学
PEKING UNIVERSITY

AI ASIC: Design and Practice (ADaP)

Fall 2023

Systolic Array

燕博南

AI背后的芯片架构是什么?

算法：Neural Network + Monte Carlo Tree Search

Google Tensor Processing Unit 2010s

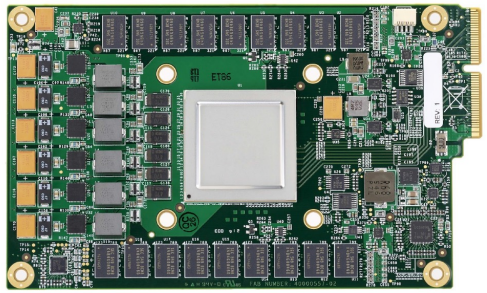


Figure 3. TPU Printed Circuit Board. It can be inserted in the slot for an SATA disk in a server, but the card uses PCIe Gen3 x16.

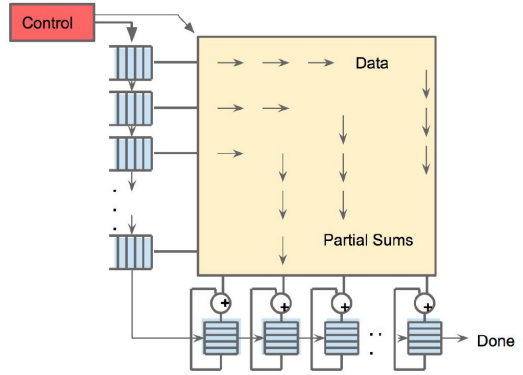
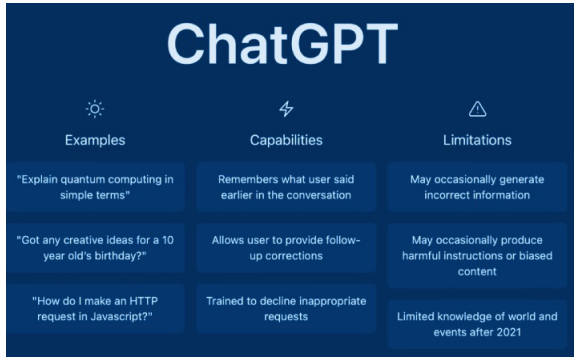
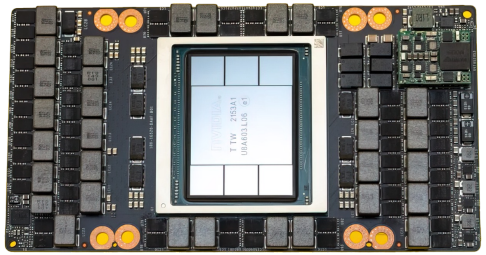


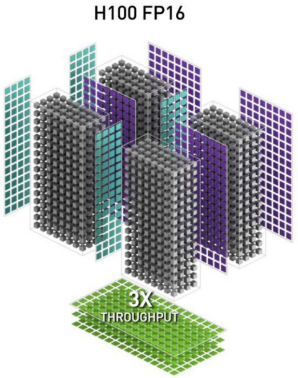
Figure 4. Systolic data flow of the Matrix Multiply Unit. Software has the illusion that each 256B input is read at once, and they instantly update one location of each of 256 accumulator RAMs.



算法：Generative Pre-trained Transformer (GPT)

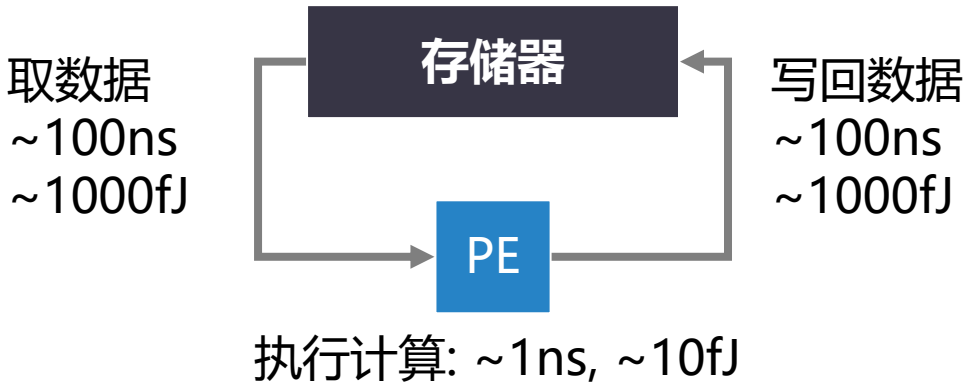
NVIDIA Tensor Core GPU 2020s





问题：硬件限制

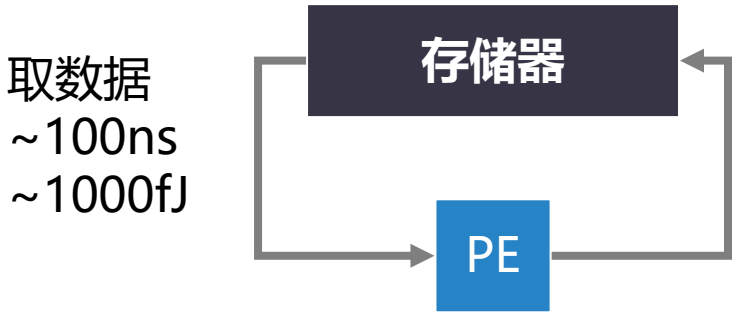
PE:计算单元 Processing Element



计算微架构的3步核心操作：

学术思想：脉动阵列Systolic Array

PE:计算单元 Processing Element

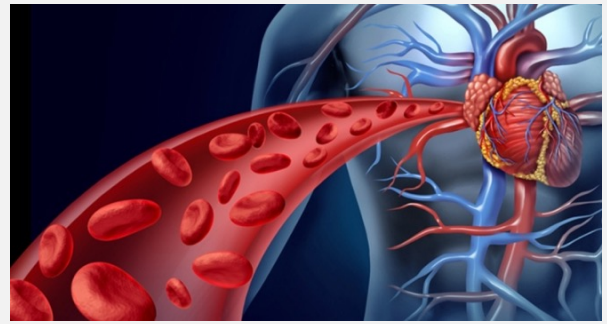


- 如果 存取数据很“贵”（能耗高、延时长）
- 那么 尽可能复用取到的数据

Prof. Hsiang-Tsung Kung

~比喻~

存储器: 心脏
数据: 血液
PEs: 细胞



“存储器”将“数据”泵至身体细胞中

H. T. Kung, “Why Systolic Architectures?,” IEEE Computer 1982

Systolic Arrays: Motivation

北京大学 PEKING UNIVERSITY

- Goal: design an accelerator that has
  - Simple, regular design (keep # unique parts small and regular)
  - High concurrency → high performance
  - Balanced computation and I/O (memory) bandwidth

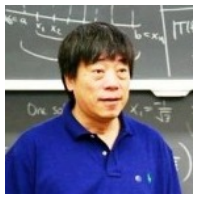
Wait a min! "concurrency vs. parallelism"

并发 vs. 并行

- Idea: Replace a single processing element (PE) with a regular array of PEs and carefully orchestrate flow of data between the PEs
  - such that they collectively transform a piece of input data before outputting it to memory

- Benefit: Maximizes computation done on a single piece of data element brought from memory

Systolic Arrays



Hsiang-Tsung Kung

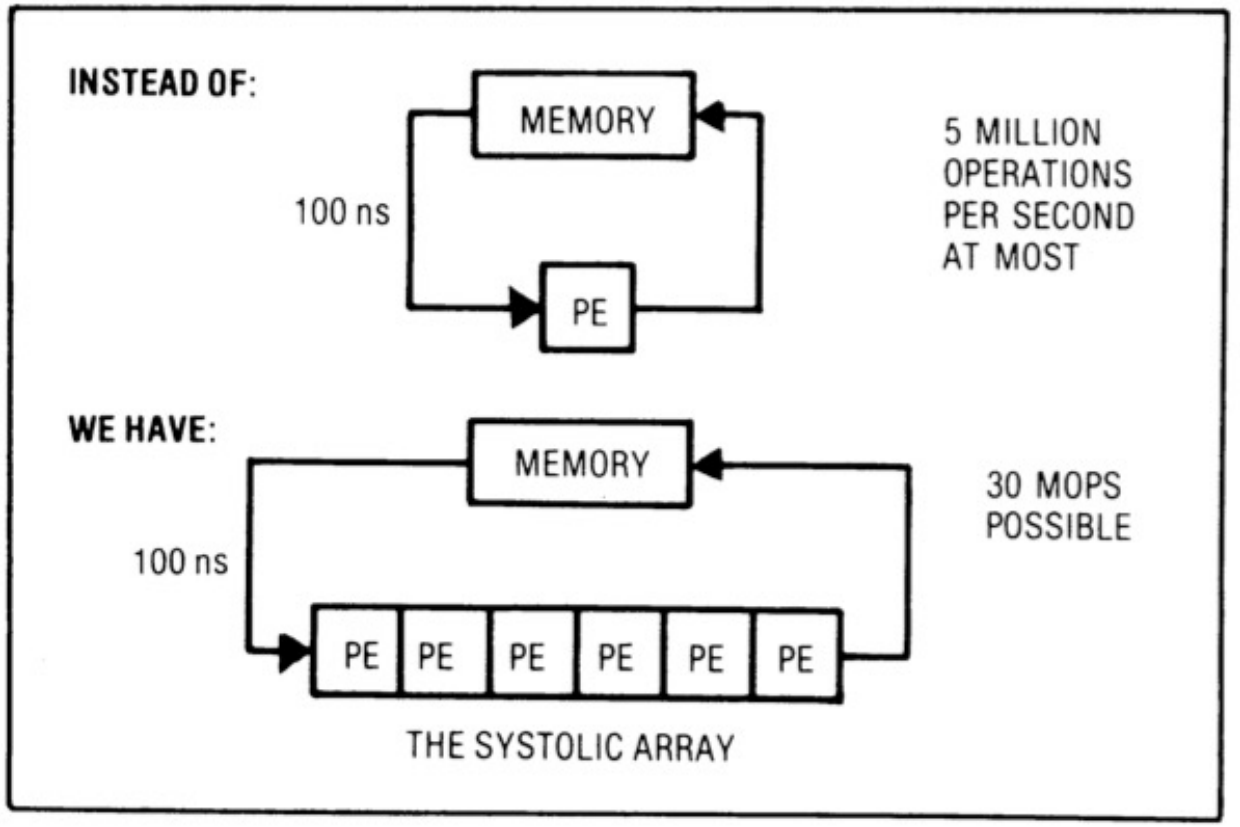


Figure 1. Basic principle of a systolic system.

Memory: heart
Data: blood
PEs: cells

Memory pulses data through PEs

- H. T. Kung, "Why Systolic Architectures?," IEEE Computer 1982.
- H.T. Kung | Ph.D. Advice (harvard.edu)

Why Systolic Architectures?

PEKING UNIVERSITY

- Idea: Data flows from the computer memory in a rhythmic fashion, passing through many processing elements before it returns to memory

- Similar to blood flow: heart → many cells → heart
  - Different cells "process" the blood
  - Many veins operate simultaneously
  - Can be many-dimensional

- Why? Special purpose accelerators/architectures need
  - Simple, regular design (keep # unique parts small and regular)
  - High concurrency → high performance
  - Balanced computation and I/O (memory) bandwidth

Systolic Computation Example

- Convolution
  - Used in filtering, pattern matching, correlation, polynomial evaluation, etc …
  - Many image processing tasks
  - Machine learning: up to hundreds of convolutional layers in Convolutional Neural Networks (CNN) (renaissance after 40years !!)

Given the sequence of weights $\{w_1, w_2, \ldots, w_k\}$
and the input sequence $\{x_1, x_2, \ldots, x_n\}$,

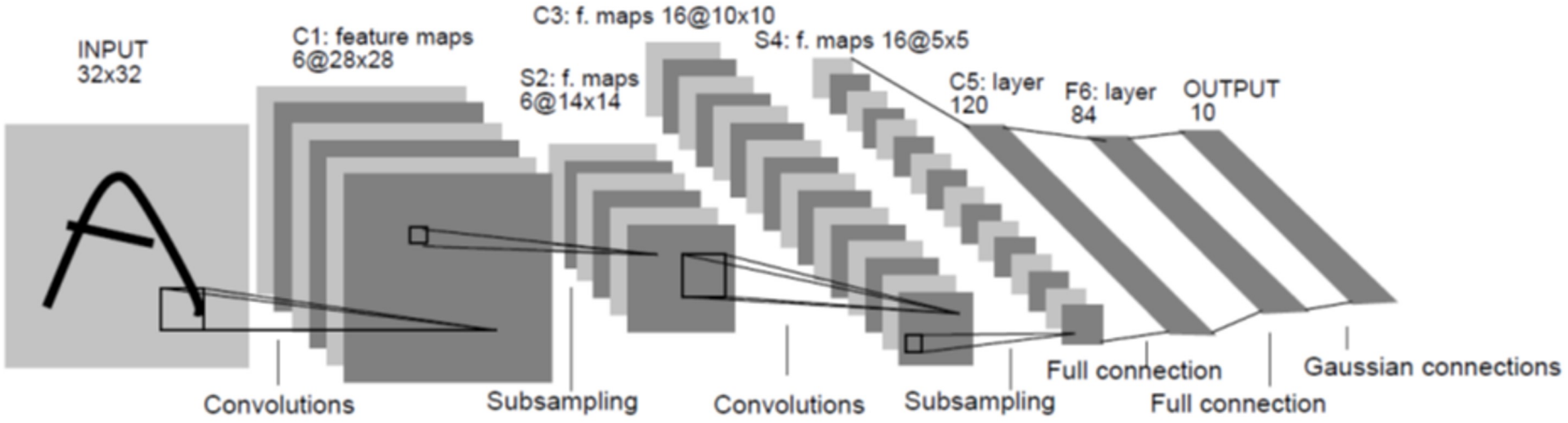
compute the result sequence $\{y_1, y_2, \ldots, y_{n+1-k}\}$
defined by

$$y_i = w_1 x_i + w_2 x_{i+1} + \ldots + w_k x_{i+k-1}$$

… 77 227 62 138 36 132 111 93 207 112 199 124 84 111 35 40 130 88 244 165 206 118 …

LeNet-5, a Convolutional Neural Network for Hand-Written Digit Recognition

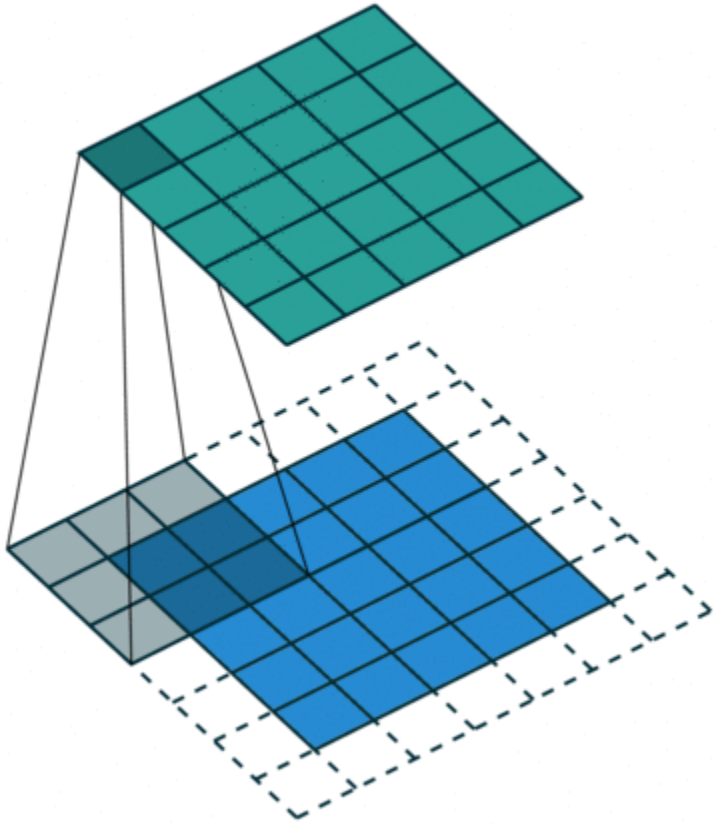
This is a 1024*8 bit input, which will have a truth table of 2^{8196} entries



Slide credit: Hwu & Kirk

An Example of 2D Convolution

Output feature map



Input feature map

Structure information

Input: 5*5 (blue)

Kernel (filter): 3*3 (grey)

Output: 5*5 (green)

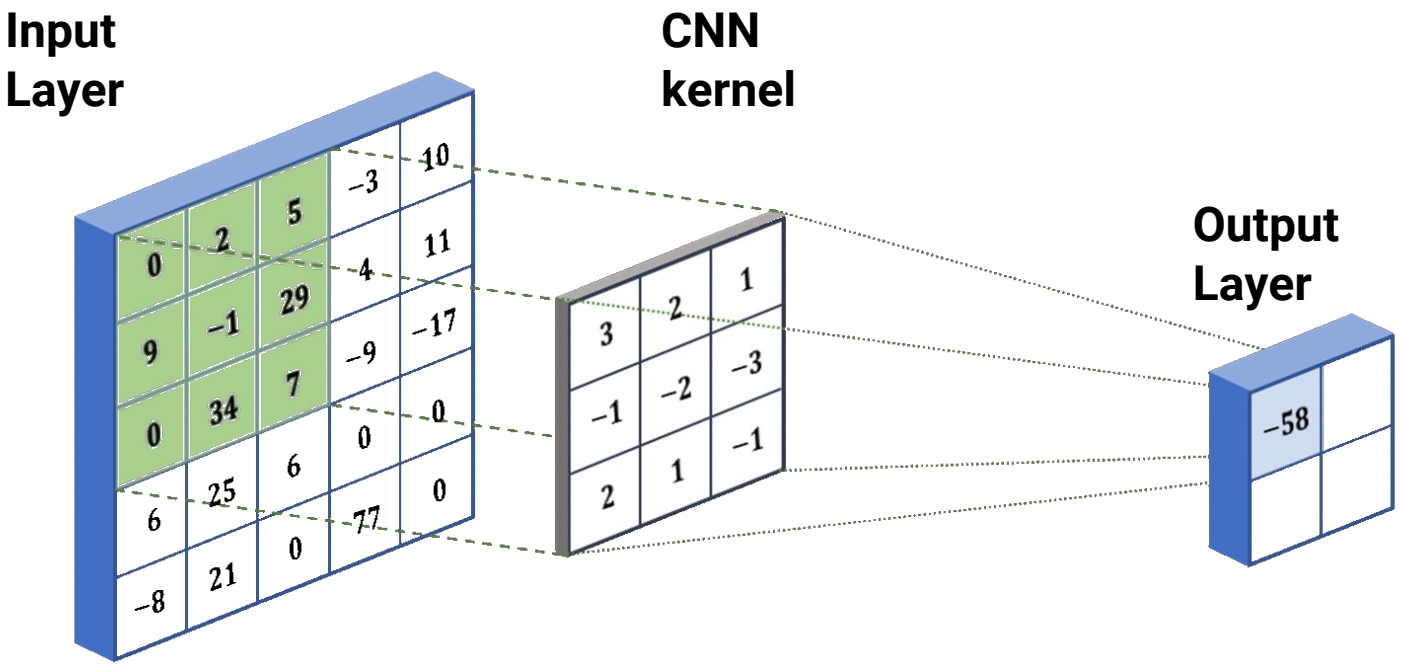
Computation information

Stride: 1

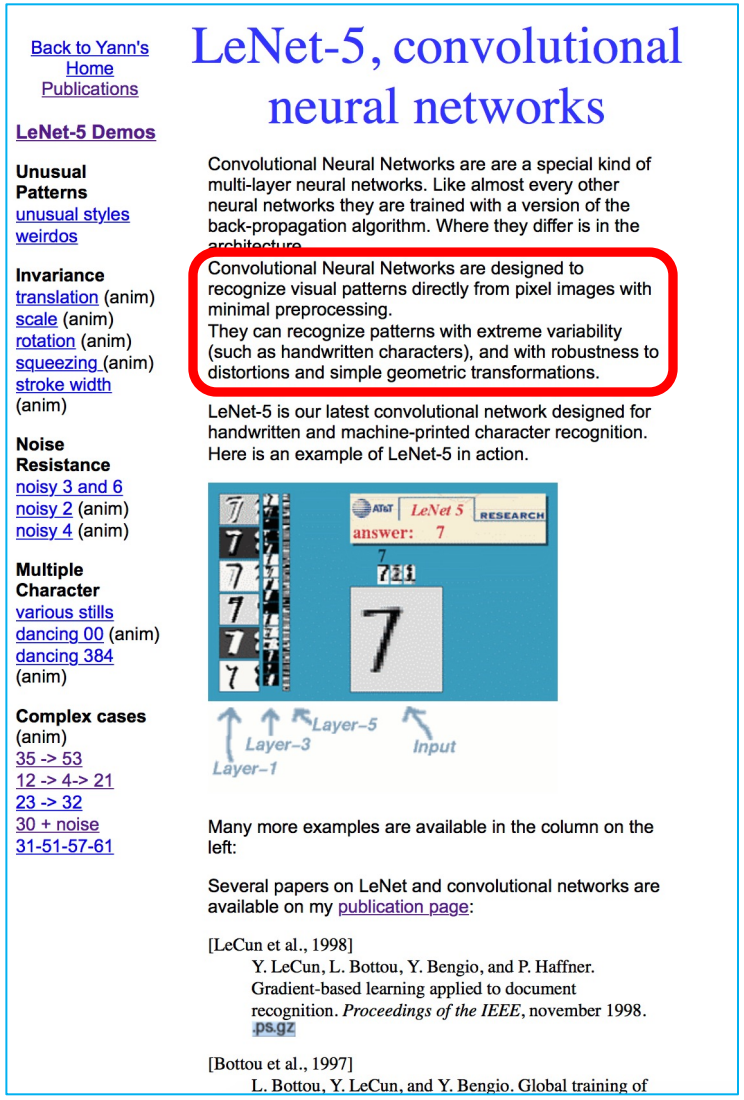
Padding: 1 (white)

Output Dim = (Input + 2*Padding - Kernel) / Stride + 1

An Example of 2D Convolution



Convolutional Neural Networks: Demo



Back to Yann's Home Publications

LeNet-5 Demos

Unusual Patterns
unusual styles
weirdos

Invariance
translation (anim)
scale (anim)
rotation (anim)
squeezing (anim)
stroke width (anim)

Noise Resistance
noisy 3 and 6
noisy 2 (anim)
noisy 4 (anim)

Multiple Character
various stills
dancing 00 (anim)
dancing 384 (anim)

Complex cases (anim)
35 -> 53
12 -> 4-> 21
23 -> 32
30 + noise
31-51-57-61

LeNet-5, convolutional neural networks

Convolutional Neural Networks are are a special kind of multi-layer neural networks. Like almost every other neural networks they are trained with a version of the back-propagation algorithm. Where they differ is in the architecture.

Convolutional Neural Networks are designed to recognize visual patterns directly from pixel images with minimal preprocessing.
They can recognize patterns with extreme variability (such as handwritten characters), and with robustness to distortions and simple geometric transformations.

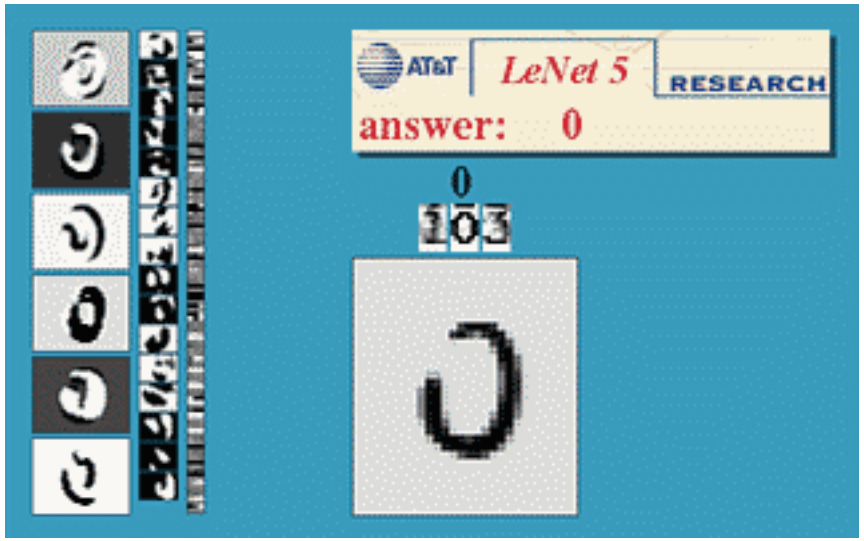
LeNet-5 is our latest convolutional network designed for handwritten and machine-printed character recognition. Here is an example of LeNet-5 in action.

Many more examples are available in the column on the left:

Several papers on LeNet and convolutional networks are available on my publication page:

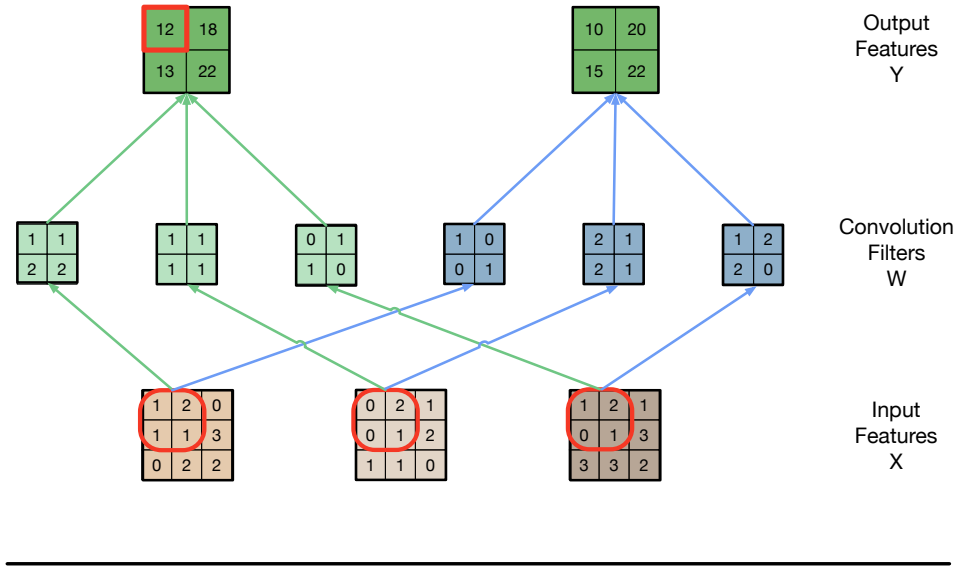
[LeCun et al., 1998]
Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, november 1998. ps.gz

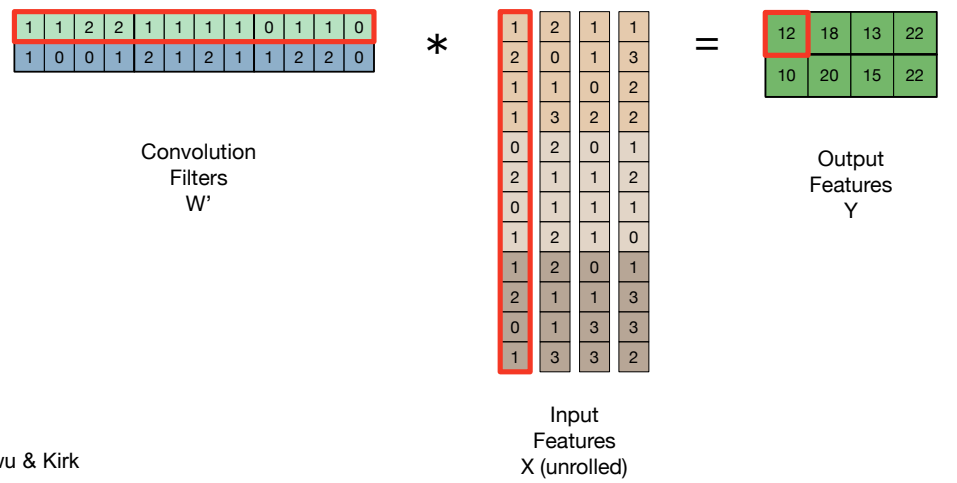
[Bottou et al., 1997]
L. Bottou, Y. LeCun, and Y. Bengio. Global training of



http://yann.lecun.com/exdb/lenet/index.html

Implementing a Convolutional Layer with Matrix Multiplication





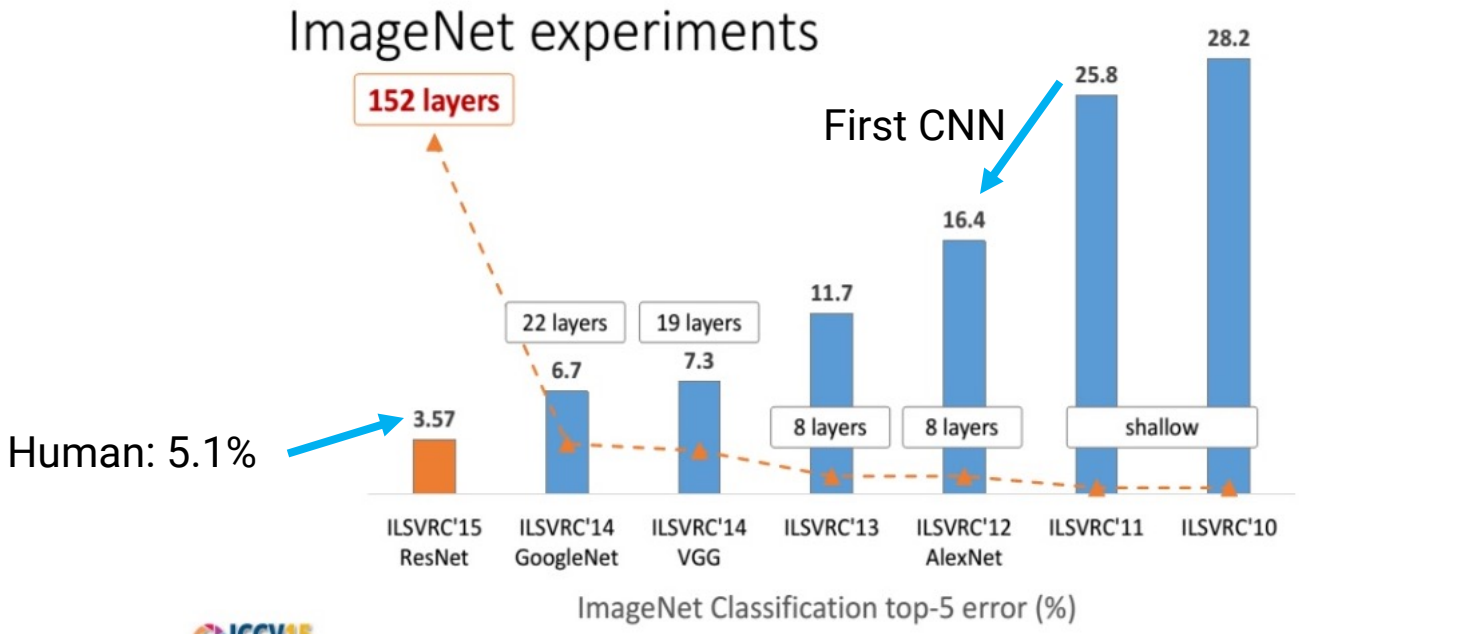
Slide credit: Reproduced from Hwu & Kirk

Example: ResNet (2015)

- He et al., "Deep Residual Learning for Image Recognition", CVPR 2016.

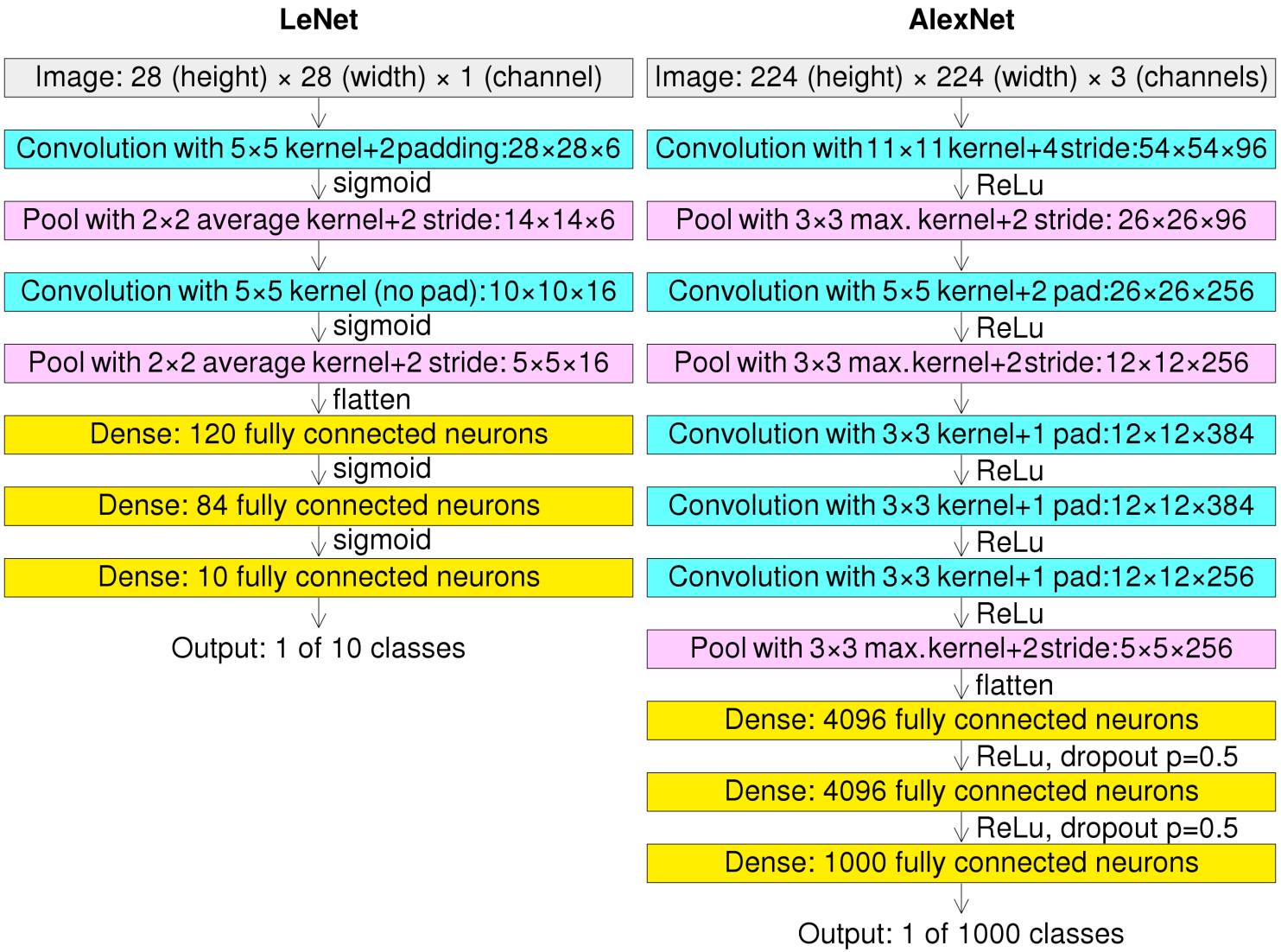
Deep Residual Learning for Image Recognition

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com



Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

Neural Network Layer Examples



Systolic Computation Example: Convolution (I)

PEKING UNIVERSITY

- Convolution
  - Used in filtering, pattern matching, correlation, polynomial evaluation, etc …
  - Many image processing tasks
  - Machine learning: up to hundreds of convolutional layers in Convolutional Neural Networks (CNN)

Given the sequence of weights $\{w_1, w_2, \ldots, w_k\}$
and the input sequence $\{x_1, x_2, \ldots, x_n\}$,

compute the result sequence $\{y_1, y_2, \ldots, y_{n+1-k}\}$
defined by

$$y_i = w_1 x_i + w_2 x_{i+1} + \ldots + w_k x_{i+k-1}$$

Systolic Computation Example: Convolution (II)

- $y1 = w1x1 + w2x2 + w3x3$
- $y2 = w1x2 + w2x3 + w3x4$
- $y3 = w1x3 + w2x4 + w3x5$

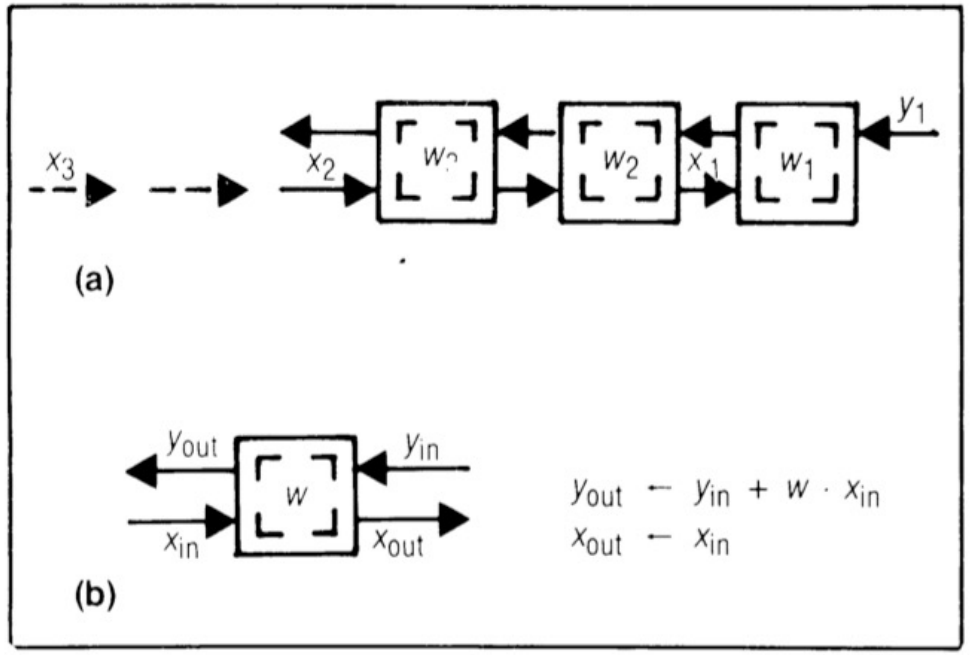


Figure 8. Design W1: systolic convolution array (a) and cell (b) where w_i's stay and x_i's and y_i's move systolically in opposite directions.

Systolic Computation Example: Convolution (III)

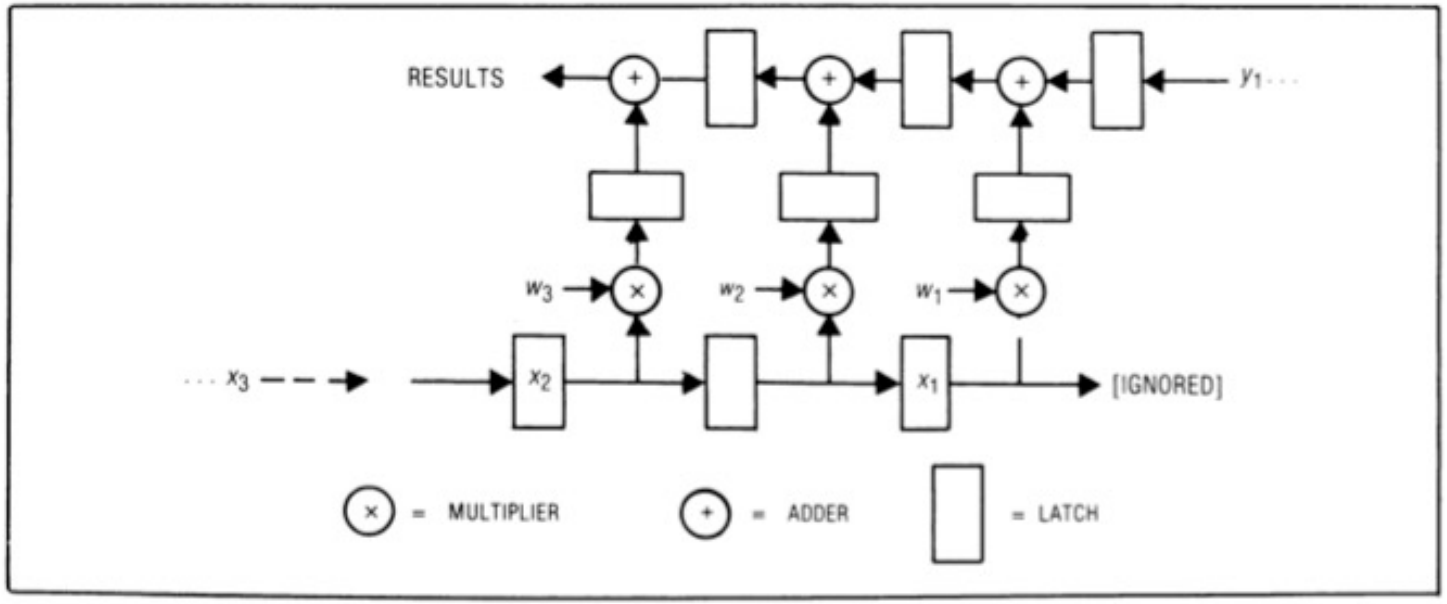


Figure 10. Overlapping the executions of multiply and add in design W1.

- Worthwhile to implement adder and multiplier separately to allow overlapping of add/mul executions

Systolic Computation Example: Convolution (IV)

- One needs to carefully orchestrate when data elements are input to the array
- And when output is buffered

- This gets more involved when
  - Array dimensionality increases
  - PEs are less predictable in terms of latency

2维脉动阵列计算 实例

- 3x3 矩阵乘法

$$\begin{bmatrix} a00 & a01 & a02 \\ a10 & a11 & a12 \\ a20 & a21 & a22 \end{bmatrix} \begin{bmatrix} b00 & b01 & b02 \\ b10 & b11 & b12 \\ b20 & b21 & b22 \end{bmatrix} = \begin{bmatrix} c00 & c01 & c02 \\ c10 & c11 & c12 \\ c20 & c21 & c22 \end{bmatrix}$$

2维脉动阵列计算 实例

北京大学 PEKING UNIVERSITY

- 3x3 矩阵乘法

$$\begin{bmatrix} a00 & a01 & a02 \\ a10 & a11 & a12 \\ a20 & a21 & a22 \end{bmatrix} \begin{bmatrix} b00 & b01 & b02 \\ b10 & b11 & b12 \\ b20 & b21 & b22 \end{bmatrix} = \begin{bmatrix} c00 & c01 & c02 \\ c10 & c11 & c12 \\ c20 & c21 & c22 \end{bmatrix}$$

2维脉动阵列计算 实例

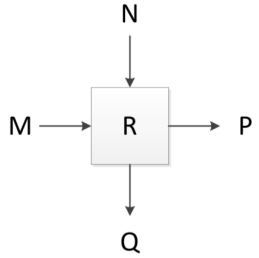
北京大学 PEKING UNIVERSITY

- 3x3 矩阵乘法

$$\begin{bmatrix} a00 & a01 & a02 \\ a10 & a11 & a12 \\ a20 & a21 & a22 \end{bmatrix} \begin{bmatrix} b00 & b01 & b02 \\ b10 & b11 & b12 \\ b20 & b21 & b22 \end{bmatrix} = \begin{bmatrix} c00 & c01 & c02 \\ c10 & c11 & c12 \\ c20 & c21 & c22 \end{bmatrix}$$

```
c00= a00*b00 + a01*b10 + a02*b20
c01= a00*b01 + a01*b11 + a02*b21
c02= a00*b02 + a01*b12 + a02*b22

c10= a10*b00 + a11*b10 + a12*b20
c11= a10*b01 + a11*b11 + a12*b21
c12= a10*b02 + a11*b12 + a12*b22

c20= a20*b00 + a21*b10 + a22*b20
c21= a20*b01 + a21*b11 + a22*b21
c22= a20*b02 + a21*b12 + a22*b22
```

• 设计硬件单元：



P = M

Q = N

R = R + M*N

Figure 1: A systolic array processing element

- 3x3 矩阵乘法

$$\begin{bmatrix} a00 & a01 & a02 \\ a10 & a11 & a12 \\ a20 & a21 & a22 \end{bmatrix} \begin{bmatrix} b00 & b01 & b02 \\ b10 & b11 & b12 \\ b20 & b21 & b22 \end{bmatrix} = \begin{bmatrix} c00 & c01 & c02 \\ c10 & c11 & c12 \\ c20 & c21 & c22 \end{bmatrix}$$

```
c00= a00*b00 + a01*b10 + a02*b20
c01= a00*b01 + a01*b11 + a02*b21
c02= a00*b02 + a01*b12 + a02*b22

c10= a10*b00 + a11*b10 + a12*b20
c11= a10*b01 + a11*b11 + a12*b21
c12= a10*b02 + a11*b12 + a12*b22

c20= a20*b00 + a21*b10 + a22*b20
c21= a20*b01 + a21*b11 + a22*b21
c22= a20*b02 + a21*b12 + a22*b22
```

• 设计硬件单元：

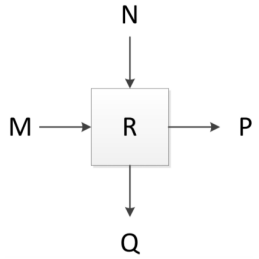


Figure 1: A systolic array processing element

P = M

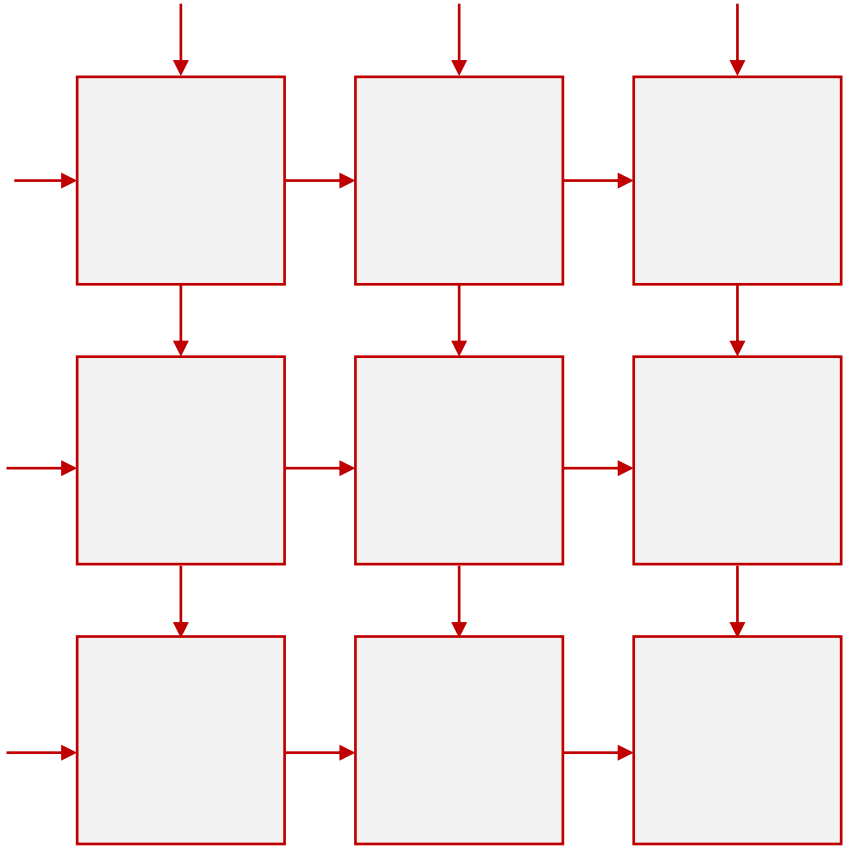
Q = N

R = R + M*N

- 3x3 矩阵乘法

$$\begin{bmatrix} a00 & a01 & a02 \\ a10 & a11 & a12 \\ a20 & a21 & a22 \end{bmatrix} \begin{bmatrix} b00 & b01 & b02 \\ b10 & b11 & b12 \\ b20 & b21 & b22 \end{bmatrix} = \begin{bmatrix} c00 & c01 & c02 \\ c10 & c11 & c12 \\ c20 & c21 & c22 \end{bmatrix}$$

```
c00= a00*b00 + a01*b10 + a02*b20
c01= a00*b01 + a01*b11 + a02*b21
c02= a00*b02 + a01*b12 + a02*b22

c10= a10*b00 + a11*b10 + a12*b20
c11= a10*b01 + a11*b11 + a12*b21
c12= a10*b02 + a11*b12 + a12*b22

c20= a20*b00 + a21*b10 + a22*b20
c21= a20*b01 + a21*b11 + a22*b21
c22= a20*b02 + a21*b12 + a22*b22
```



• 设计硬件单元：

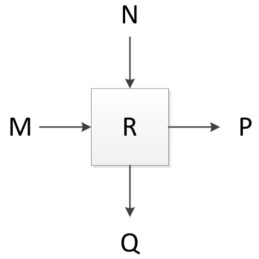


Figure 1: A systolic array processing element

P = M

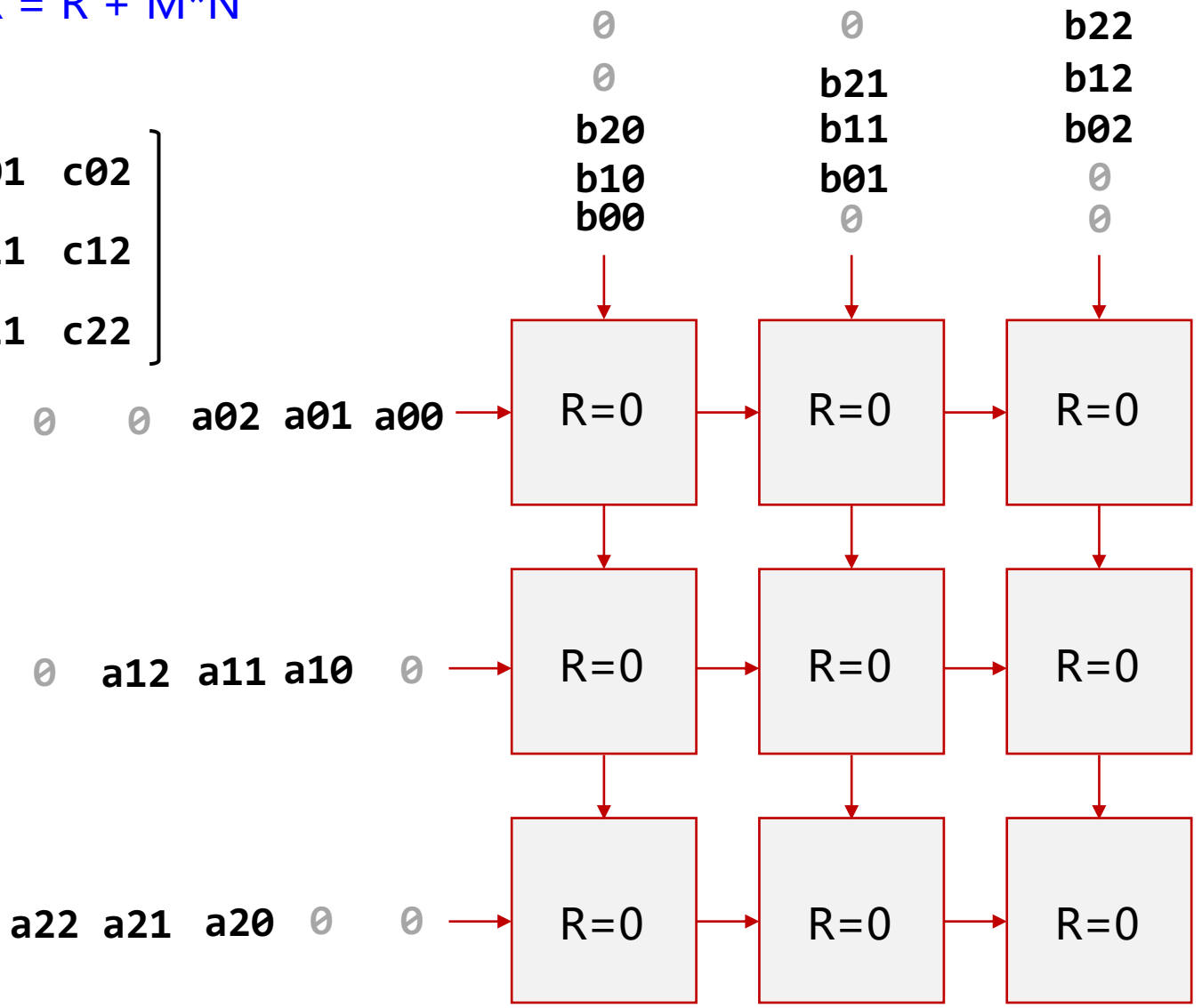
Q = N

R = R + M*N

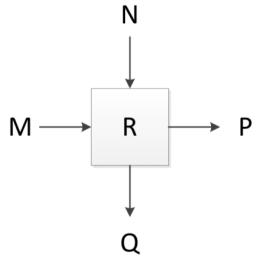
- 3x3 矩阵乘法

$$\begin{bmatrix} a00 & a01 & a02 \\ a10 & a11 & a12 \\ a20 & a21 & a22 \end{bmatrix} \begin{bmatrix} b00 & b01 & b02 \\ b10 & b11 & b12 \\ b20 & b21 & b22 \end{bmatrix} = \begin{bmatrix} c00 & c01 & c02 \\ c10 & c11 & c12 \\ c20 & c21 & c22 \end{bmatrix}$$

```
c00= a00*b00 + a01*b10 + a02*b20
c01= a00*b01 + a01*b11 + a02*b21
c02= a00*b02 + a01*b12 + a02*b22

c10= a10*b00 + a11*b10 + a12*b20
c11= a10*b01 + a11*b11 + a12*b21
c12= a10*b02 + a11*b12 + a12*b22

c20= a20*b00 + a21*b10 + a22*b20
c21= a20*b01 + a21*b11 + a22*b21
c22= a20*b02 + a21*b12 + a22*b22
```



• 设计硬件单元：



P = M

Q = N

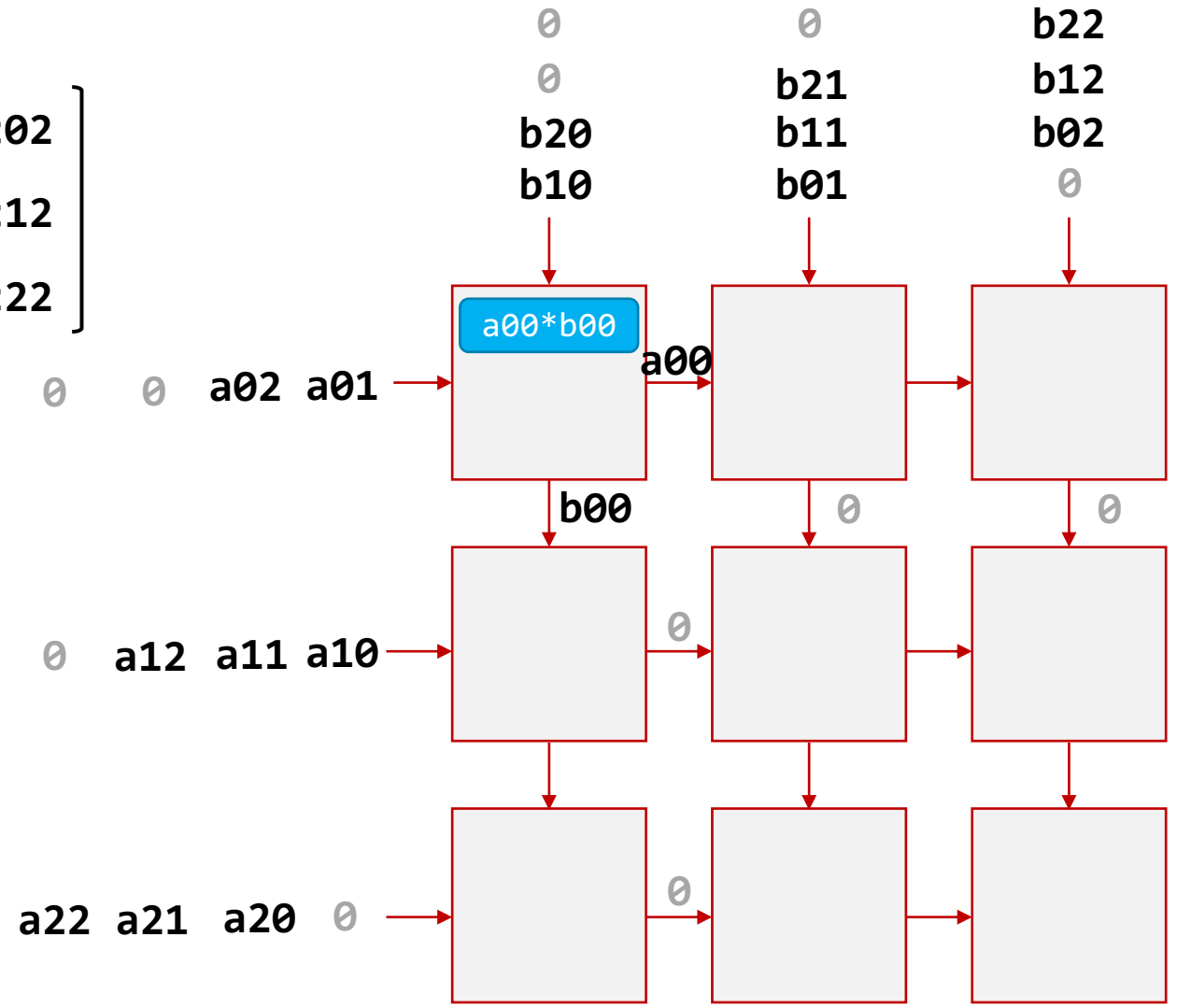
R = R + M*N

Figure 1: A systolic array processing element

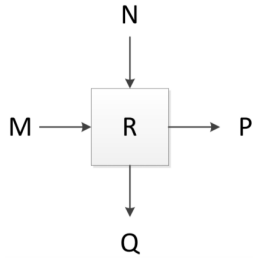
- 3x3 矩阵乘法

$$\begin{bmatrix} a00 & a01 & a02 \\ a10 & a11 & a12 \\ a20 & a21 & a22 \end{bmatrix} \begin{bmatrix} b00 & b01 & b02 \\ b10 & b11 & b12 \\ b20 & b21 & b22 \end{bmatrix} = \begin{bmatrix} c00 & c01 & c02 \\ c10 & c11 & c12 \\ c20 & c21 & c22 \end{bmatrix}$$

```
c00= a00*b00 + a01*b10 + a02*b20
c01= a00*b01 + a01*b11 + a02*b21
c02= a00*b02 + a01*b12 + a02*b22

c10= a10*b00 + a11*b10 + a12*b20
c11= a10*b01 + a11*b11 + a12*b21
c12= a10*b02 + a11*b12 + a12*b22

c20= a20*b00 + a21*b10 + a22*b20
c21= a20*b01 + a21*b11 + a22*b21
c22= a20*b02 + a21*b12 + a22*b22
```



• 设计硬件单元：



P = M

Q = N

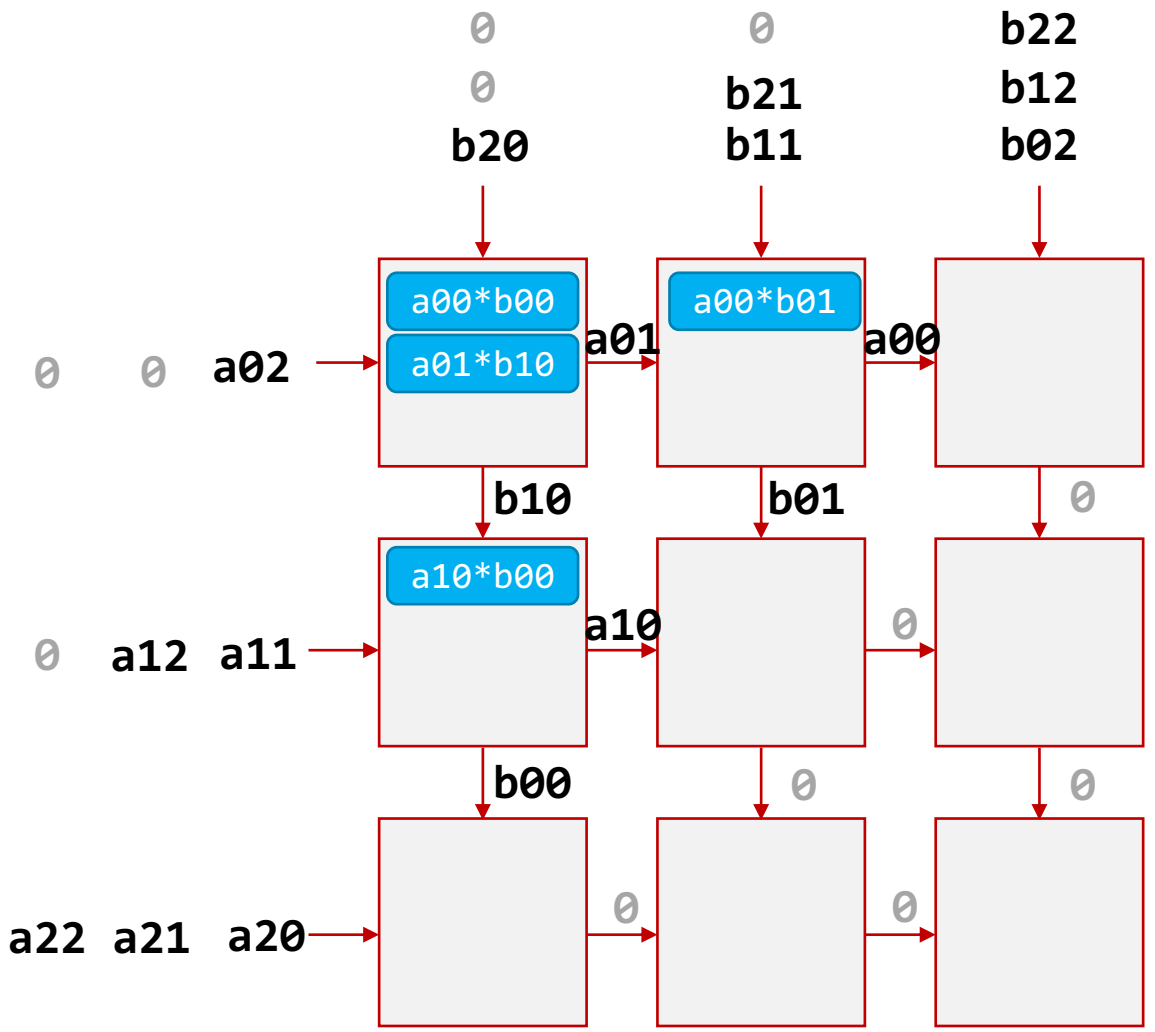
R = R + M*N

Figure 1: A systolic array processing element

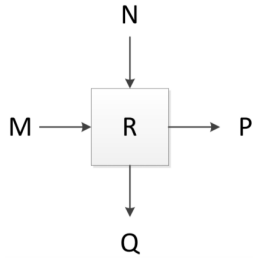
- 3x3 矩阵乘法

$$\begin{bmatrix} a00 & a01 & a02 \\ a10 & a11 & a12 \\ a20 & a21 & a22 \end{bmatrix} \begin{bmatrix} b00 & b01 & b02 \\ b10 & b11 & b12 \\ b20 & b21 & b22 \end{bmatrix} = \begin{bmatrix} c00 & c01 & c02 \\ c10 & c11 & c12 \\ c20 & c21 & c22 \end{bmatrix}$$

```
c00= a00*b00 + a01*b10 + a02*b20
c01= a00*b01 + a01*b11 + a02*b21
c02= a00*b02 + a01*b12 + a02*b22

c10= a10*b00 + a11*b10 + a12*b20
c11= a10*b01 + a11*b11 + a12*b21
c12= a10*b02 + a11*b12 + a12*b22

c20= a20*b00 + a21*b10 + a22*b20
c21= a20*b01 + a21*b11 + a22*b21
c22= a20*b02 + a21*b12 + a22*b22
```



• 设计硬件单元：



P = M

Q = N

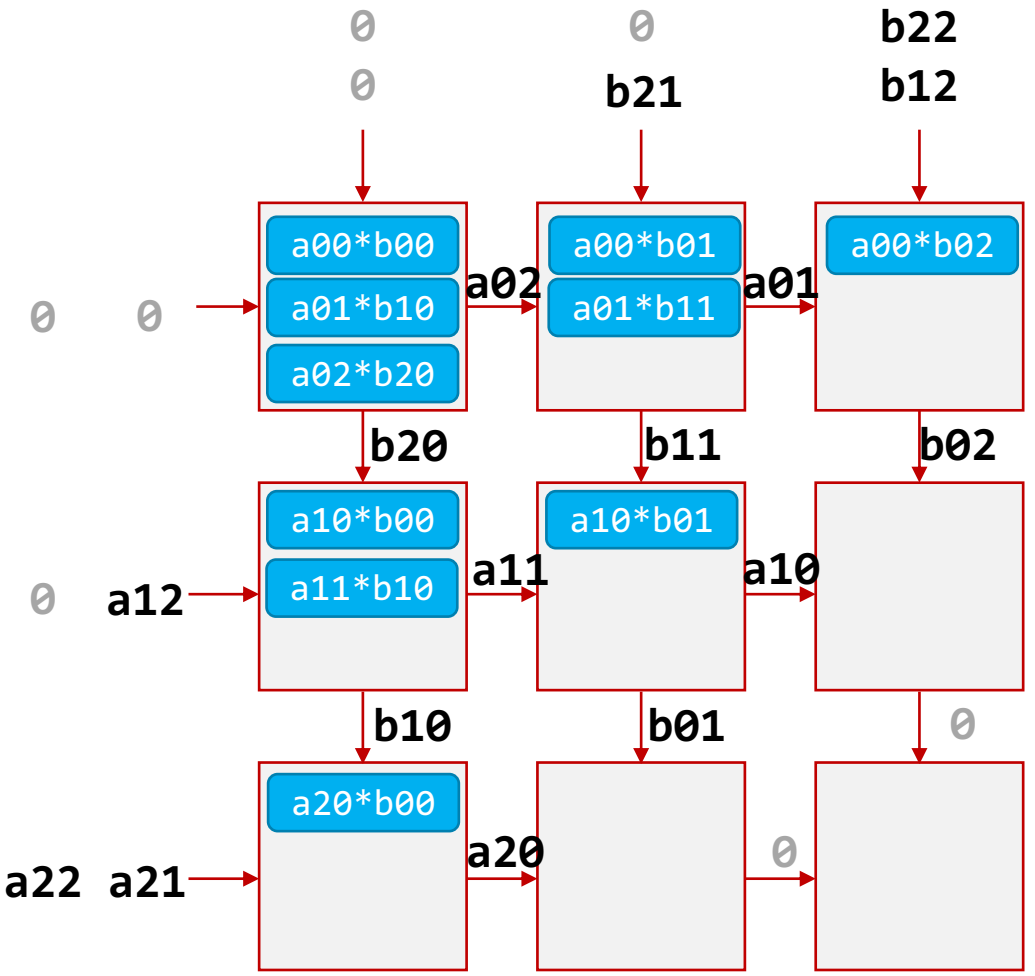
R = R + M*N

Figure 1: A systolic array processing element

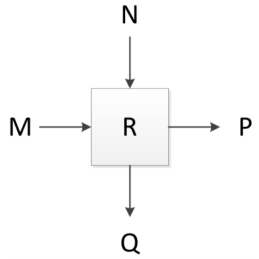
- 3x3 矩阵乘法

$$\begin{bmatrix} a00 & a01 & a02 \\ a10 & a11 & a12 \\ a20 & a21 & a22 \end{bmatrix} \begin{bmatrix} b00 & b01 & b02 \\ b10 & b11 & b12 \\ b20 & b21 & b22 \end{bmatrix} = \begin{bmatrix} c00 & c01 & c02 \\ c10 & c11 & c12 \\ c20 & c21 & c22 \end{bmatrix}$$

```
c00= a00*b00 + a01*b10 + a02*b20
c01= a00*b01 + a01*b11 + a02*b21
c02= a00*b02 + a01*b12 + a02*b22

c10= a10*b00 + a11*b10 + a12*b20
c11= a10*b01 + a11*b11 + a12*b21
c12= a10*b02 + a11*b12 + a12*b22

c20= a20*b00 + a21*b10 + a22*b20
c21= a20*b01 + a21*b11 + a22*b21
c22= a20*b02 + a21*b12 + a22*b22
```



• 设计硬件单元：



P = M

Q = N

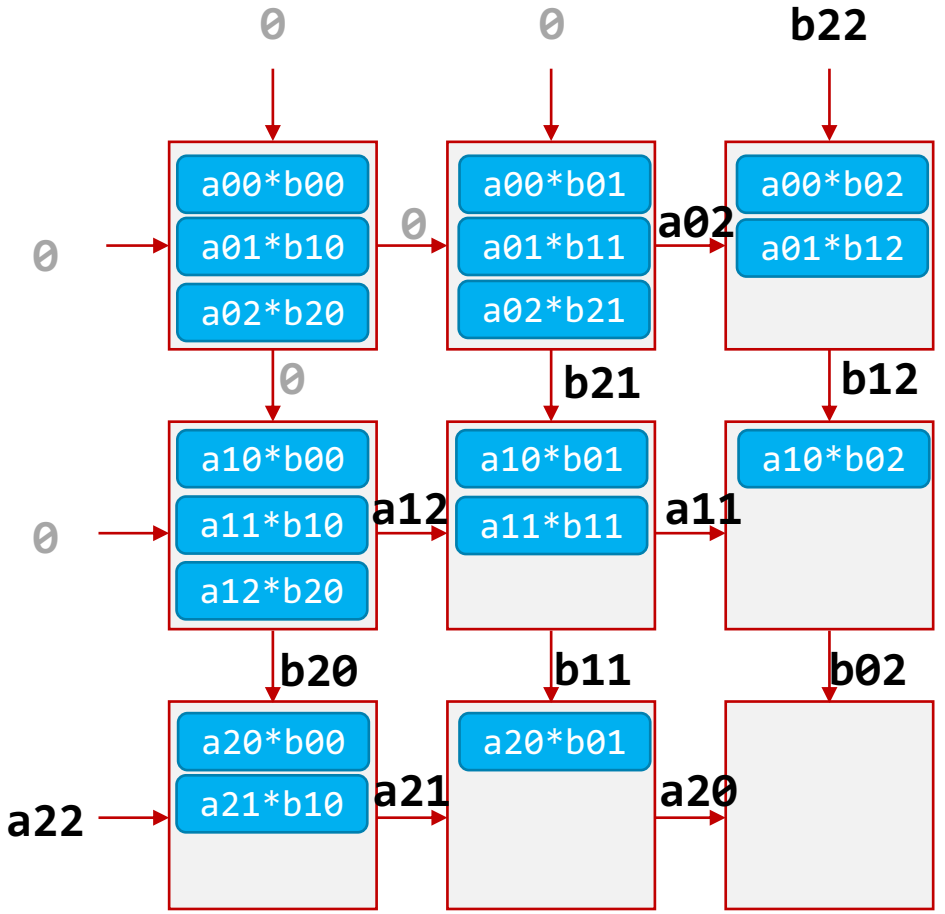
R = R + M*N

Figure 1: A systolic array processing element

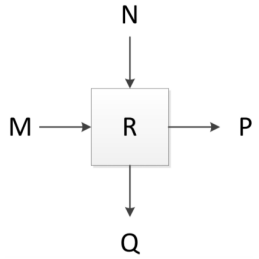
- 3x3 矩阵乘法

$$\begin{bmatrix} a00 & a01 & a02 \\ a10 & a11 & a12 \\ a20 & a21 & a22 \end{bmatrix} \begin{bmatrix} b00 & b01 & b02 \\ b10 & b11 & b12 \\ b20 & b21 & b22 \end{bmatrix} = \begin{bmatrix} c00 & c01 & c02 \\ c10 & c11 & c12 \\ c20 & c21 & c22 \end{bmatrix}$$

```
c00= a00*b00 + a01*b10 + a02*b20
c01= a00*b01 + a01*b11 + a02*b21
c02= a00*b02 + a01*b12 + a02*b22

c10= a10*b00 + a11*b10 + a12*b20
c11= a10*b01 + a11*b11 + a12*b21
c12= a10*b02 + a11*b12 + a12*b22

c20= a20*b00 + a21*b10 + a22*b20
c21= a20*b01 + a21*b11 + a22*b21
c22= a20*b02 + a21*b12 + a22*b22
```



• 设计硬件单元：



P = M

Q = N

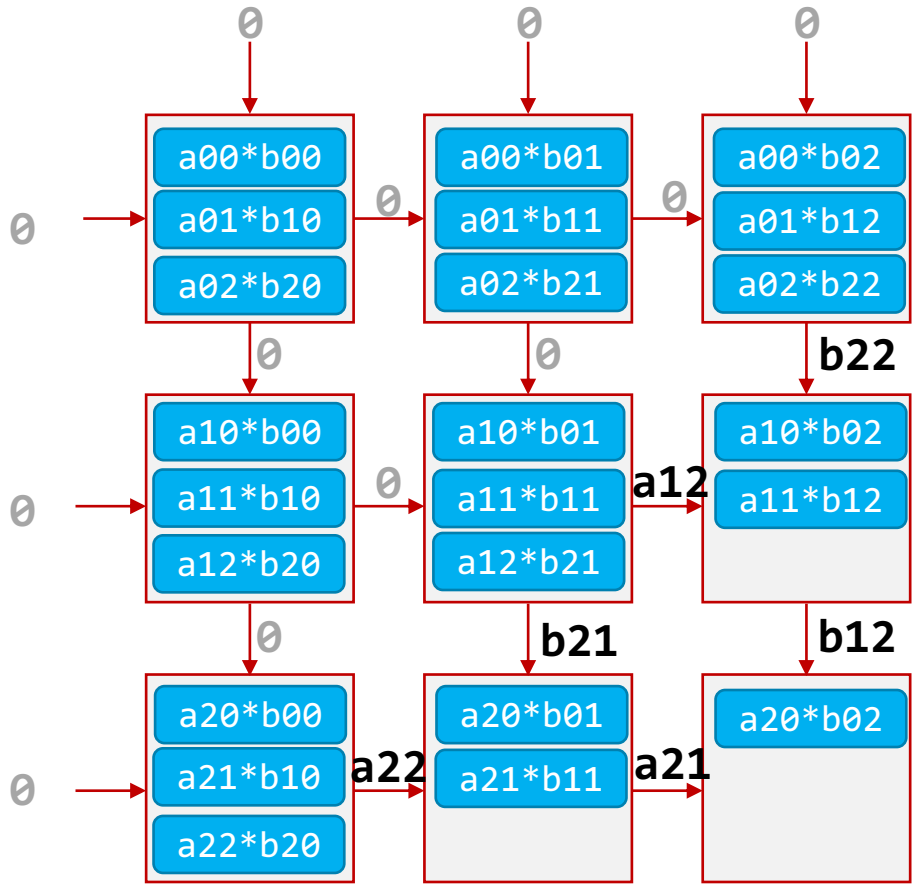
R = R + M*N

Figure 1: A systolic array processing element

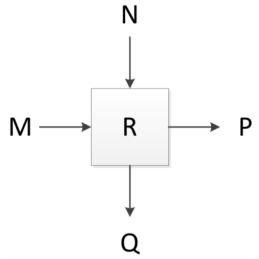
- 3x3 矩阵乘法

$$\begin{bmatrix} a00 & a01 & a02 \\ a10 & a11 & a12 \\ a20 & a21 & a22 \end{bmatrix} \begin{bmatrix} b00 & b01 & b02 \\ b10 & b11 & b12 \\ b20 & b21 & b22 \end{bmatrix} = \begin{bmatrix} c00 & c01 & c02 \\ c10 & c11 & c12 \\ c20 & c21 & c22 \end{bmatrix}$$

```
c00= a00*b00 + a01*b10 + a02*b20
c01= a00*b01 + a01*b11 + a02*b21
c02= a00*b02 + a01*b12 + a02*b22

c10= a10*b00 + a11*b10 + a12*b20
c11= a10*b01 + a11*b11 + a12*b21
c12= a10*b02 + a11*b12 + a12*b22

c20= a20*b00 + a21*b10 + a22*b20
c21= a20*b01 + a21*b11 + a22*b21
c22= a20*b02 + a21*b12 + a22*b22
```



• 设计硬件单元：



P = M

Q = N

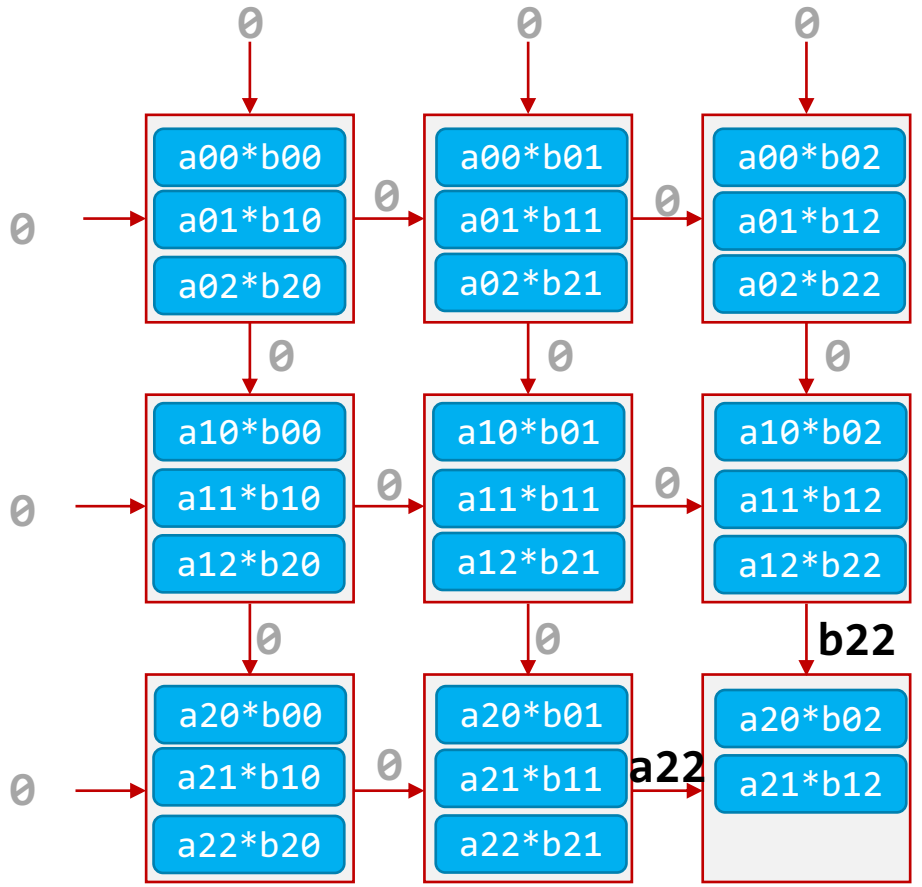
R = R + M*N

Figure 1: A systolic array processing element

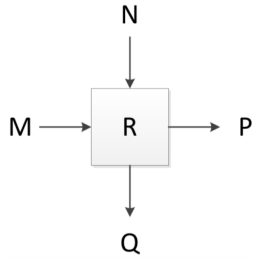
- 3x3 矩阵乘法

$$\begin{bmatrix} a00 & a01 & a02 \\ a10 & a11 & a12 \\ a20 & a21 & a22 \end{bmatrix} \begin{bmatrix} b00 & b01 & b02 \\ b10 & b11 & b12 \\ b20 & b21 & b22 \end{bmatrix} = \begin{bmatrix} c00 & c01 & c02 \\ c10 & c11 & c12 \\ c20 & c21 & c22 \end{bmatrix}$$

```
c00= a00*b00 + a01*b10 + a02*b20
c01= a00*b01 + a01*b11 + a02*b21
c02= a00*b02 + a01*b12 + a02*b22

c10= a10*b00 + a11*b10 + a12*b20
c11= a10*b01 + a11*b11 + a12*b21
c12= a10*b02 + a11*b12 + a12*b22

c20= a20*b00 + a21*b10 + a22*b20
c21= a20*b01 + a21*b11 + a22*b21
c22= a20*b02 + a21*b12 + a22*b22
```



• 设计硬件单元：



P = M

Q = N

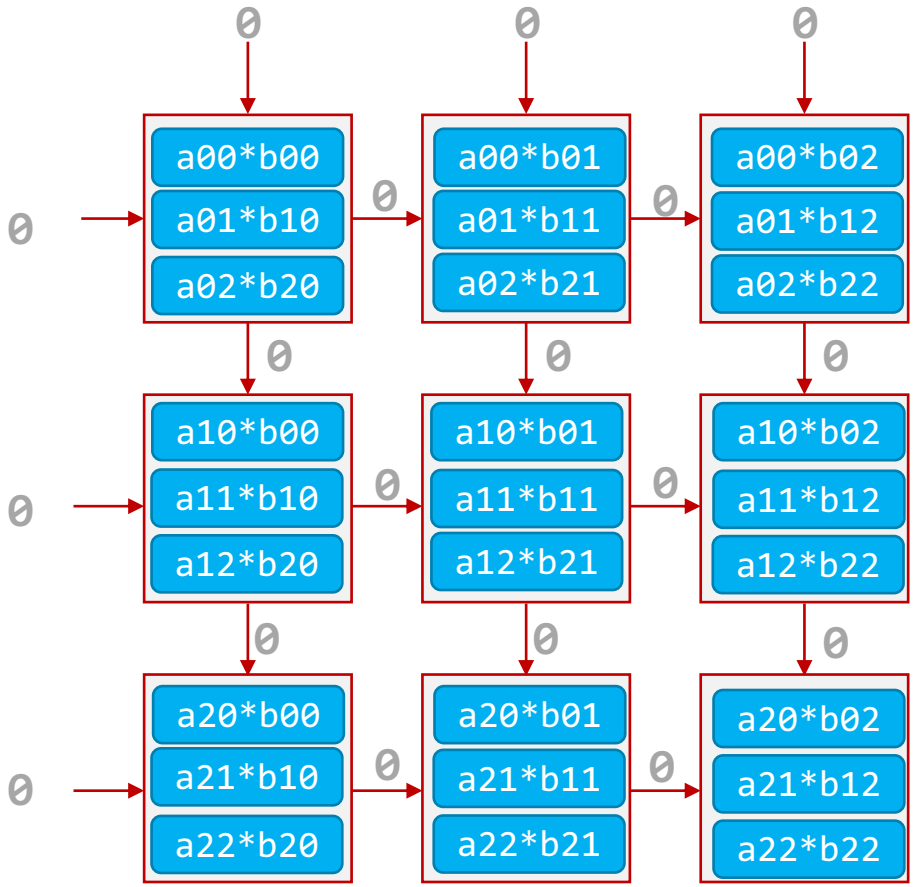
R = R + M*N

Figure 1: A systolic array processing element

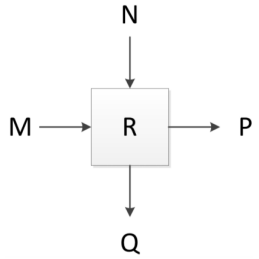
- 3x3 矩阵乘法

$$\begin{bmatrix} a00 & a01 & a02 \\ a10 & a11 & a12 \\ a20 & a21 & a22 \end{bmatrix} \begin{bmatrix} b00 & b01 & b02 \\ b10 & b11 & b12 \\ b20 & b21 & b22 \end{bmatrix} = \begin{bmatrix} c00 & c01 & c02 \\ c10 & c11 & c12 \\ c20 & c21 & c22 \end{bmatrix}$$

```
c00= a00*b00 + a01*b10 + a02*b20
c01= a00*b01 + a01*b11 + a02*b21
c02= a00*b02 + a01*b12 + a02*b22

c10= a10*b00 + a11*b10 + a12*b20
c11= a10*b01 + a11*b11 + a12*b21
c12= a10*b02 + a11*b12 + a12*b22

c20= a20*b00 + a21*b10 + a22*b20
c21= a20*b01 + a21*b11 + a22*b21
c22= a20*b02 + a21*b12 + a22*b22
```



• 设计硬件单元：



P = M

Q = N

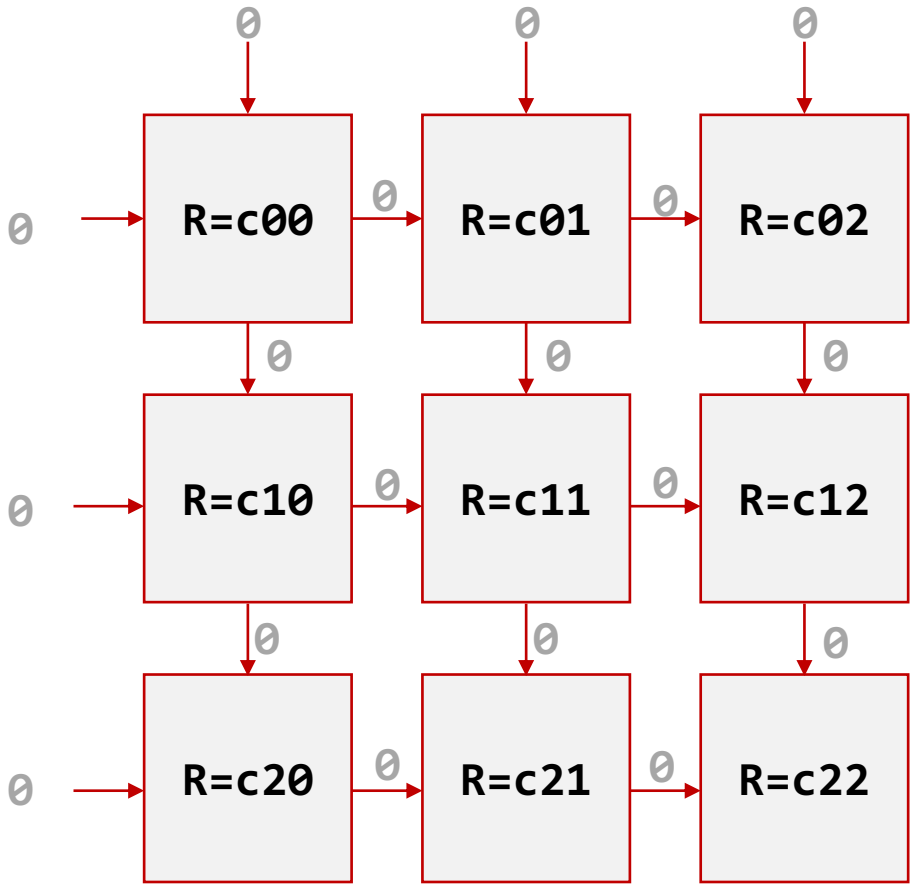
R = R + M*N

Figure 1: A systolic array processing element

- 3x3 矩阵乘法

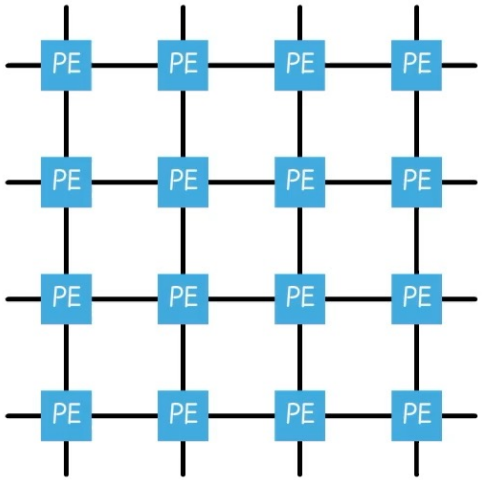
$$\begin{bmatrix} a00 & a01 & a02 \\ a10 & a11 & a12 \\ a20 & a21 & a22 \end{bmatrix} \begin{bmatrix} b00 & b01 & b02 \\ b10 & b11 & b12 \\ b20 & b21 & b22 \end{bmatrix} = \begin{bmatrix} c00 & c01 & c02 \\ c10 & c11 & c12 \\ c20 & c21 & c22 \end{bmatrix}$$

```
c00= a00*b00 + a01*b10 + a02*b20
c01= a00*b01 + a01*b11 + a02*b21
c02= a00*b02 + a01*b12 + a02*b22

c10= a10*b00 + a11*b10 + a12*b20
c11= a10*b01 + a11*b11 + a12*b21
c12= a10*b02 + a11*b12 + a12*b22

c20= a20*b00 + a21*b10 + a22*b20
c21= a20*b01 + a21*b11 + a22*b21
c22= a20*b02 + a21*b12 + a22*b22
```



2维脉动阵列计算 示例

Systolic Array



脉动阵列思想特点：

- **规则电路单元 -> 高并发度**
- **数据流动“节奏”需要专门设计**

回到问题：概念区分

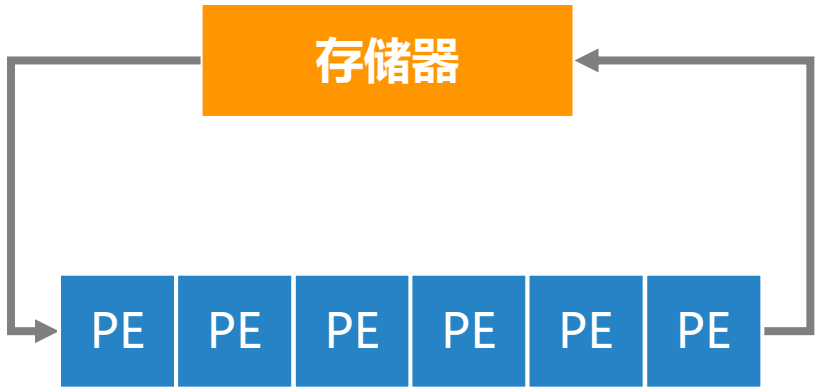
PE: Processing Element

北京大学 PEKING UNIVERSITY

脉动阵列与多核处理器流水线有什么区别?

流水线：

脉动阵列：

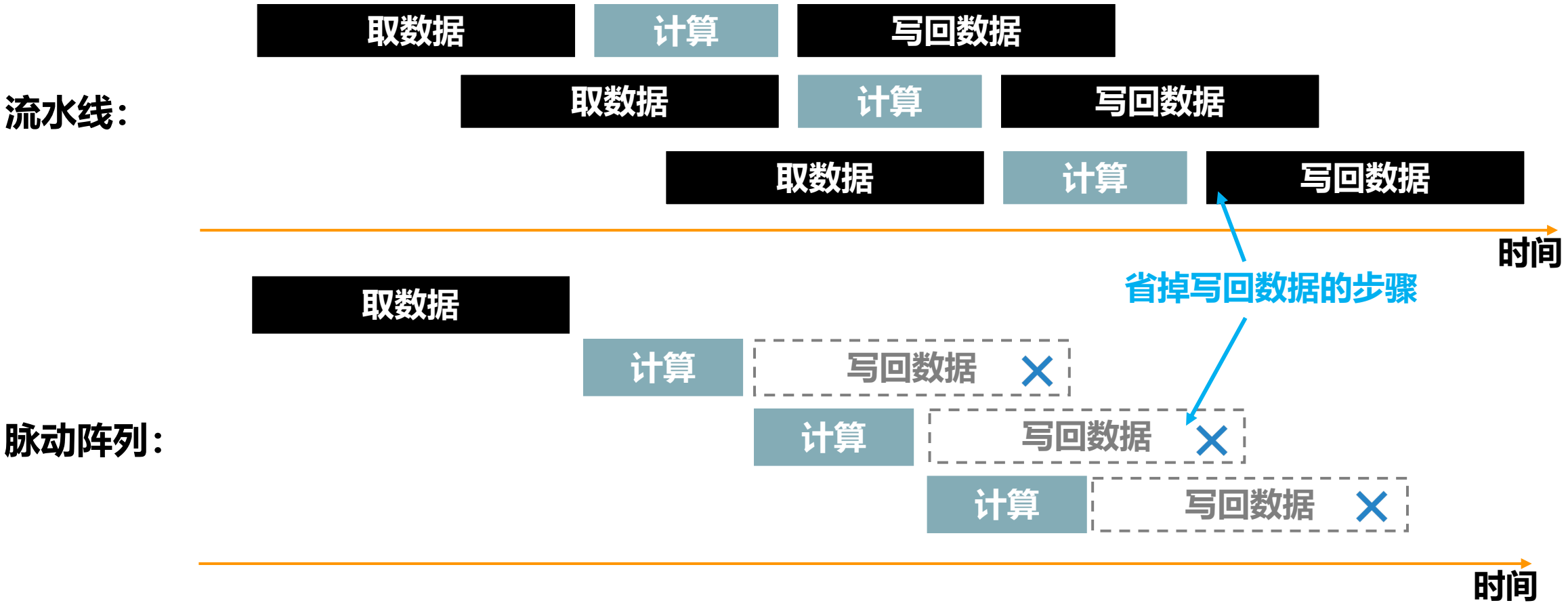


回到问题：概念区分

PE: Processing Element

北京大学 PEKING UNIVERSITY

脉动阵列与多核处理器流水线有什么区别?



Two-Dimensional Systolic Arrays

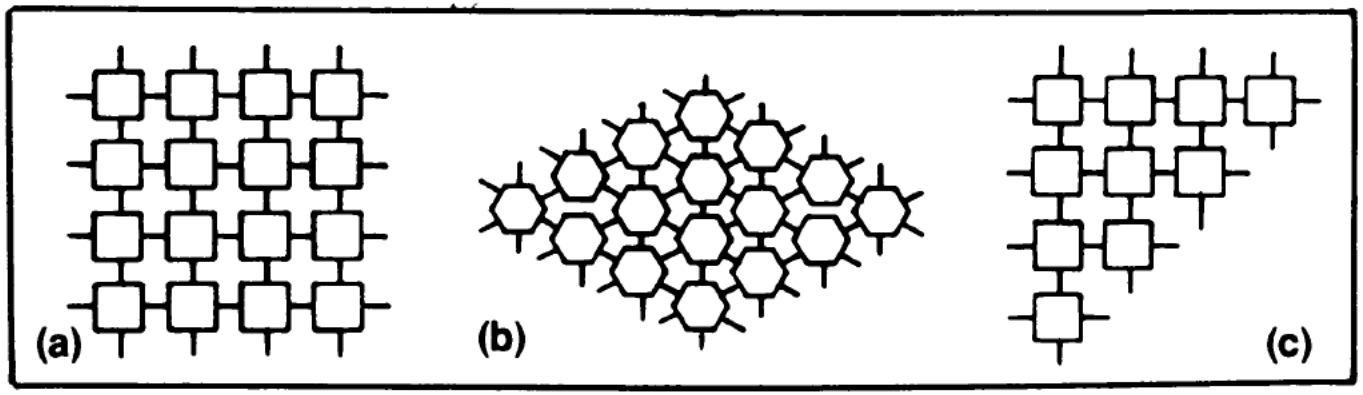


Figure 11. Two-dimensional systolic arrays: (a) type R, (b) type H, and (c) type T.

To a given problem there could be both one- and two-dimensional systolic array solutions. For example, two-dimensional convolution can be performed by a one-dimensional systolic array[24,25] or a two-dimensional systolic array.[6] When the memory speed is more than cell speed, two-dimensional systolic arrays such as those depicted in Figure 11 should be used. At each cell cycle, all the I/O ports on the array boundaries can input or output data items to or from the memory; as a result, the available memory bandwidth can be fully utilized. Thus, the choice of a one- or two-dimensional scheme is very dependent on how cells and memories will be implemented.

Combinations

- Systolic arrays can be chained together to form powerful systems
- This systolic array is capable of producing on-the-fly least-squares fit to all the data that has arrived up to any given moment

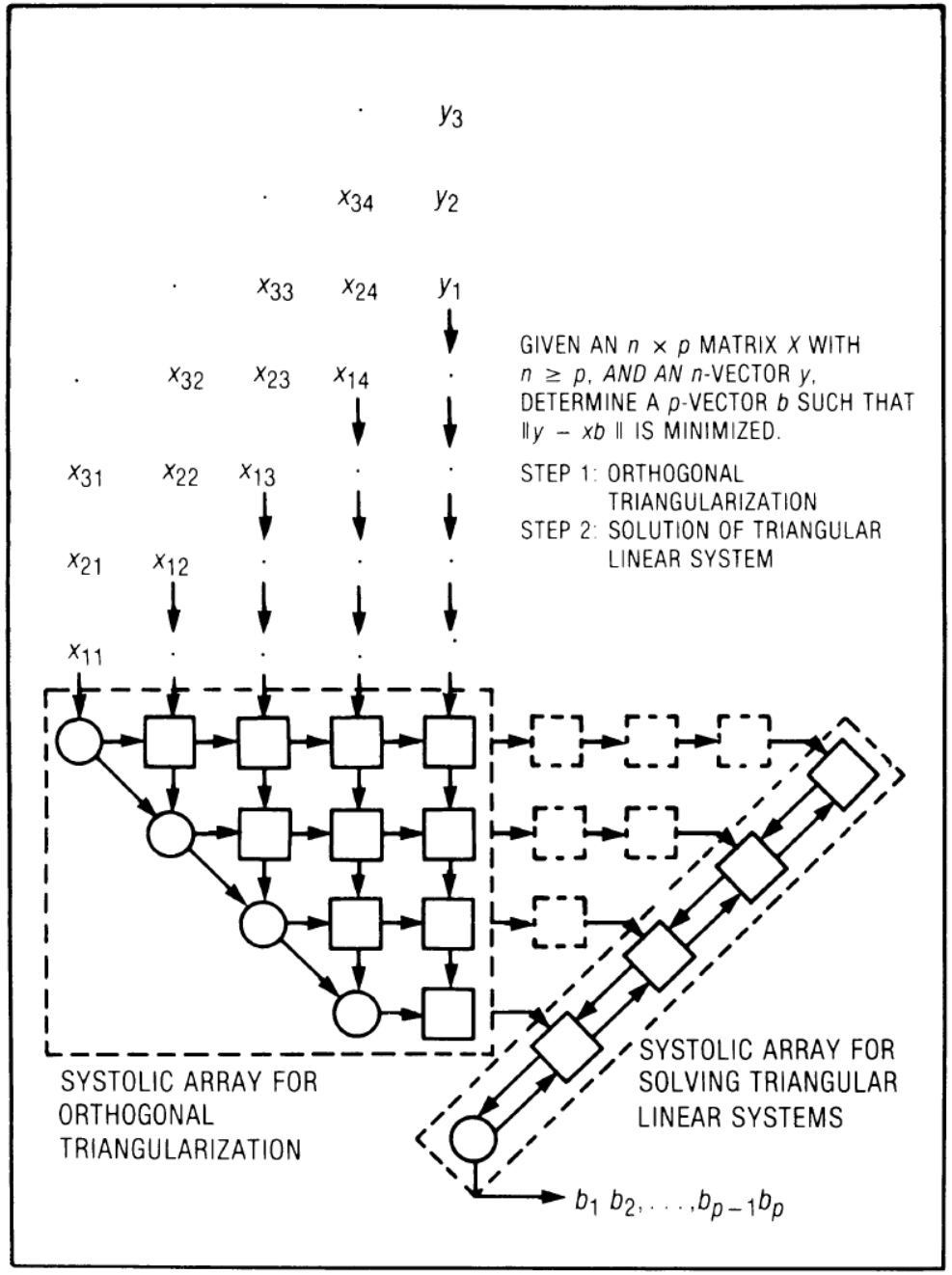


Figure 12. On-the-fly least-squares solutions using one- and two-dimensional systolic arrays, with $p = 4$.

Systolic Array: Advantages & Disadvantages

- Advantages
  - Makes multiple uses of each data item → reduced need for fetching/refetching → better use of memory bandwidth
  - High concurrency
  - Regular design (both data and control flow)

- Disadvantages
  - Not good at exploiting irregular parallelism
  - Relatively special purpose → need software, programmer support to be a general purpose model

Example Systolic Array: The WARP Computer

- HT Kung, CMU, 1984-1988

- Linear array of 10 cells, each cell a 10 Mflop programmable processor
- Attached to a general purpose host machine
- HLL and optimizing compiler to program the systolic array
- Used extensively to accelerate vision and robotics tasks

- Annaratone et al., "Warp Architecture and Implementation," ISCA 1986.
- Annaratone et al., "The Warp Computer: Architecture, Implementation, and Performance," IEEE TC 1987.

The WARP Computer

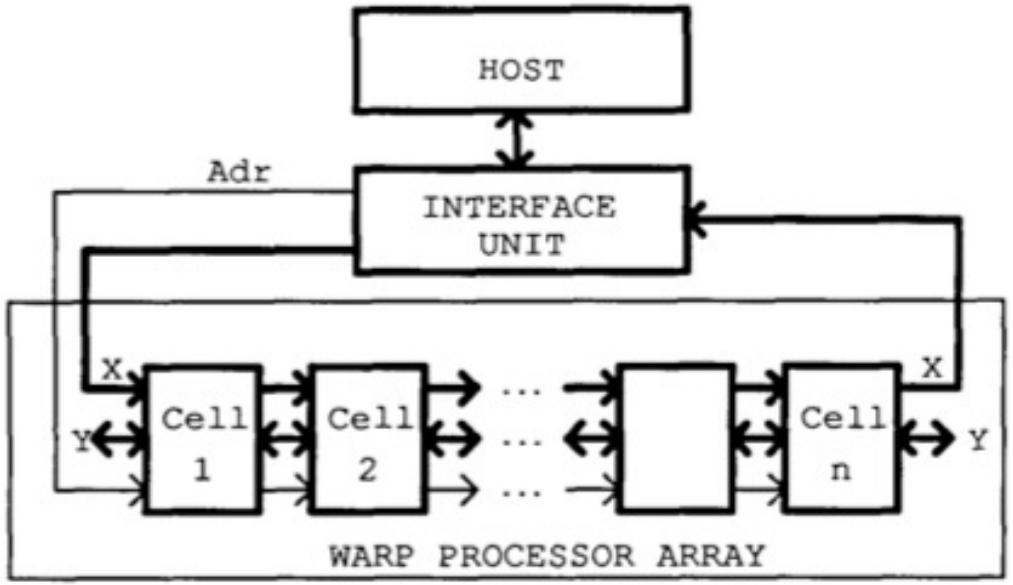


Figure 1: Warp system overview

参考："warp machine" 名字来源：织布机原理
https://www.bilibili.com/video/BV1eg41167fo

The WARP Cell

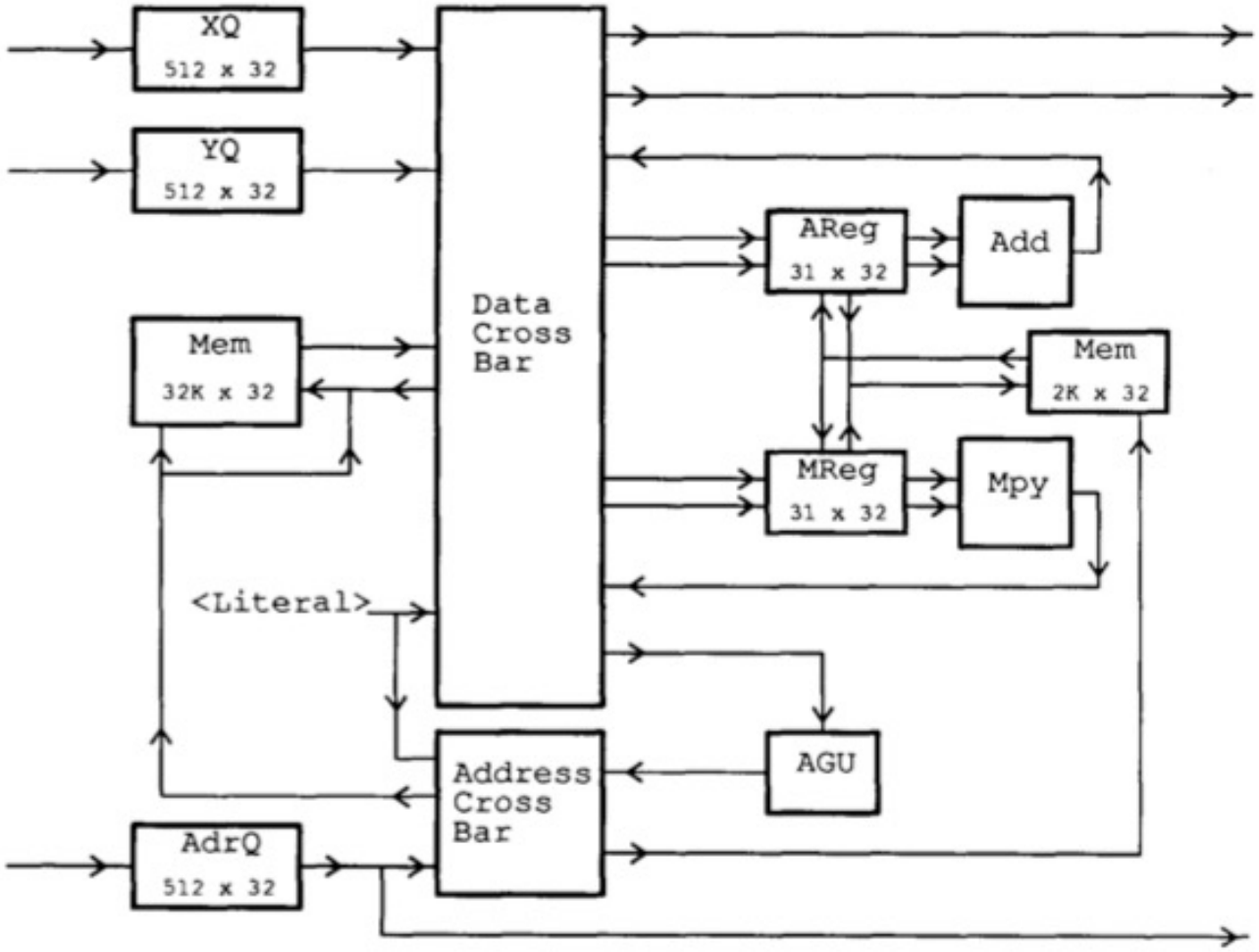


Figure 2: Warp cell data path

An Example Modern Systolic Array: TPU (I)

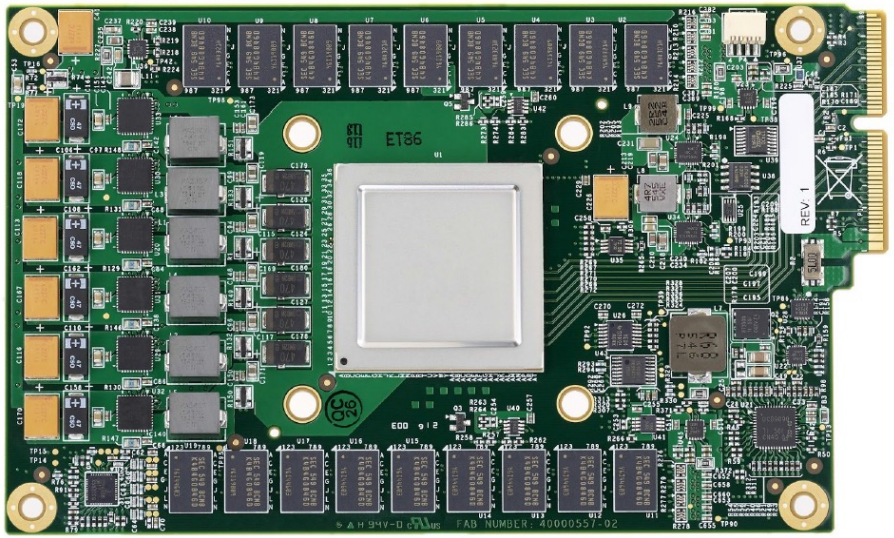


Figure 3. TPU Printed Circuit Board. It can be inserted in the slot for an SATA disk in a server, but the card uses PCIe Gen3 x16.

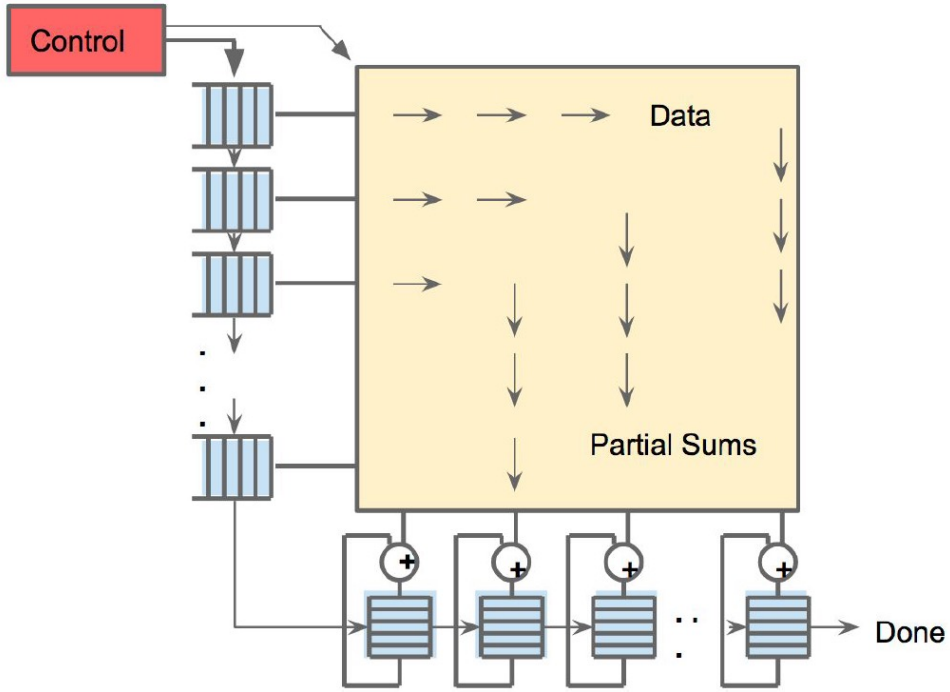


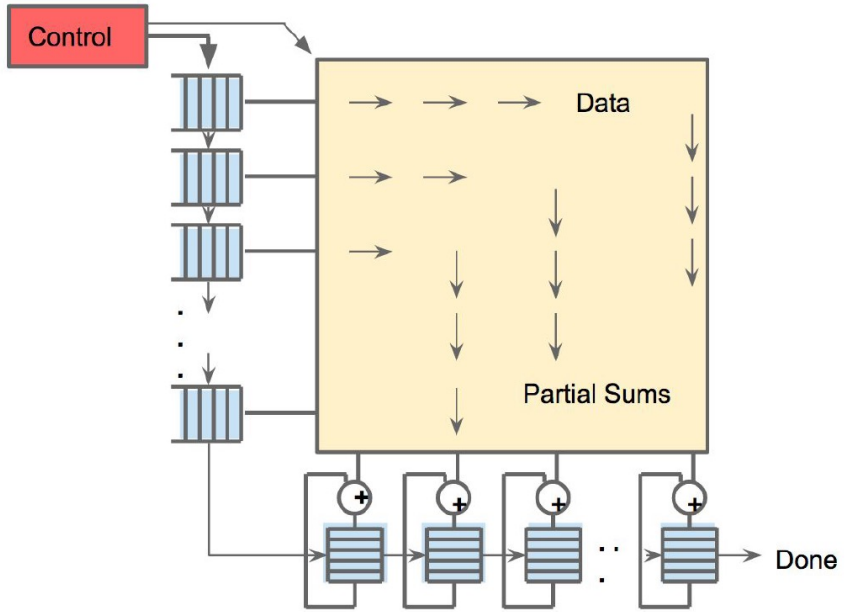
Figure 4. Systolic data flow of the Matrix Multiply Unit. Software has the illusion that each 256B input is read at once, and they instantly update one location of each of 256 accumulator RAMs.

Jouppi et al., "In-Datacenter Performance Analysis of a Tensor Processing Unit", ISCA 2017.

An Example Modern Systolic Array: TPU (II)

As reading a large SRAM uses much more power than arithmetic, the matrix unit uses systolic execution to save energy by reducing reads and writes of the Unified Buffer [Kun80][Ram91][Ovt15b]. Figure 4 shows that data flows in from the left, and the weights are loaded from the top. A given 256-element multiply-accumulate operation moves through the matrix as a diagonal wavefront. The weights are preloaded, and take effect with the advancing wave alongside the first data of a new block. Control and data are pipelined to give the illusion that the 256 inputs are read at once, and that they instantly update one location of each of 256 accumulators. From a correctness perspective, software is unaware of the systolic nature of the matrix unit, but for performance, it does worry about the latency of the unit.

Software are very important!



Jouppi et al., "In-Datacenter Performance Analysis of a Tensor Processing Unit", ISCA 2017.

Recall: Example 2D Systolic Array Computation

- Multiply two 3x3 matrices (inputs)
  - Keep the final result in PE accumulators

$$\begin{bmatrix} c_{00} & c_{01} & c_{02} \\ c_{10} & c_{11} & c_{12} \\ c_{20} & c_{21} & c_{22} \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} \times \begin{bmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \\ b_{20} & b_{21} & b_{22} \end{bmatrix}$$

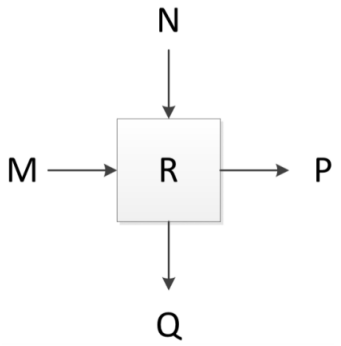
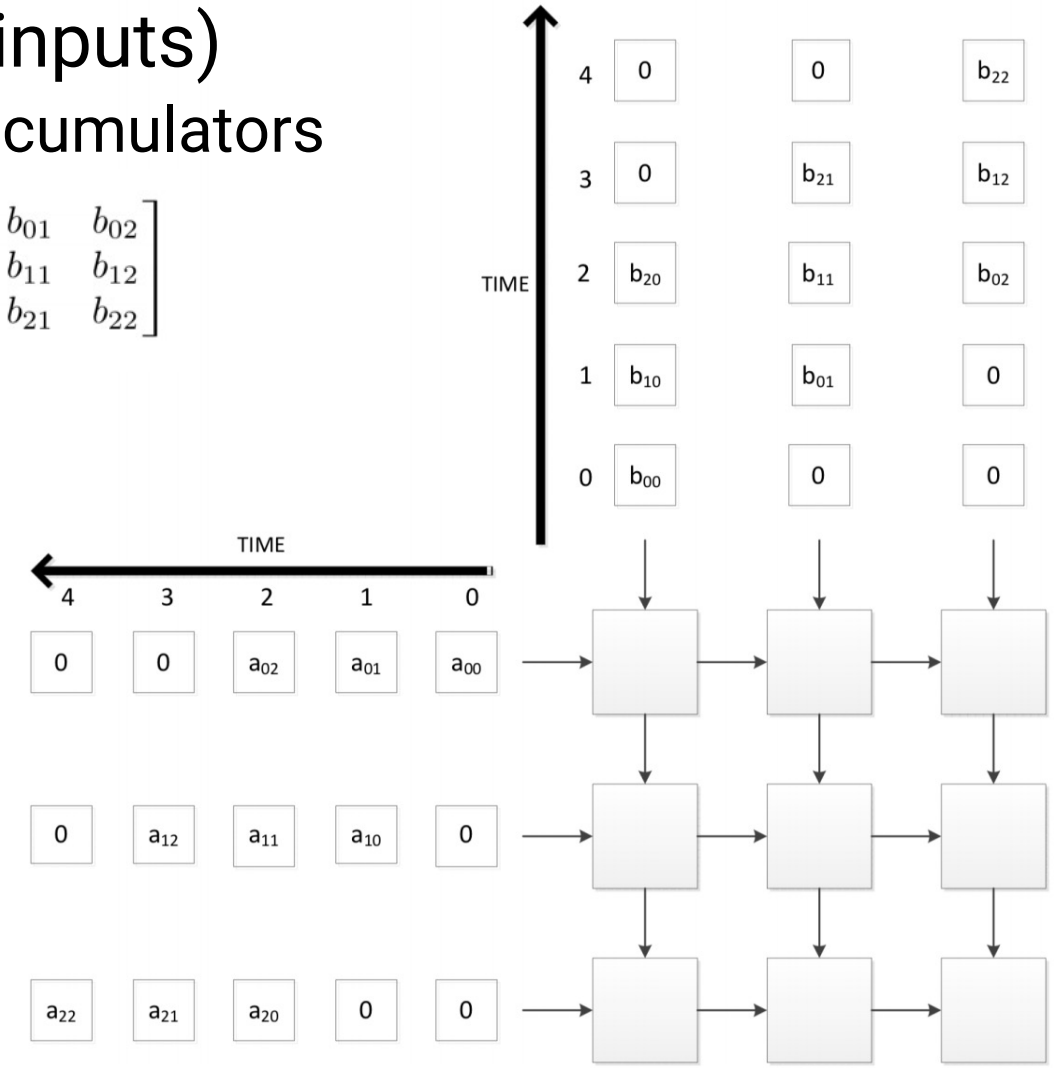


Figure 1: A systolic array processing element

P = M

Q = N

R = R + M*N



An Example Modern Systolic Array: TPU (III)

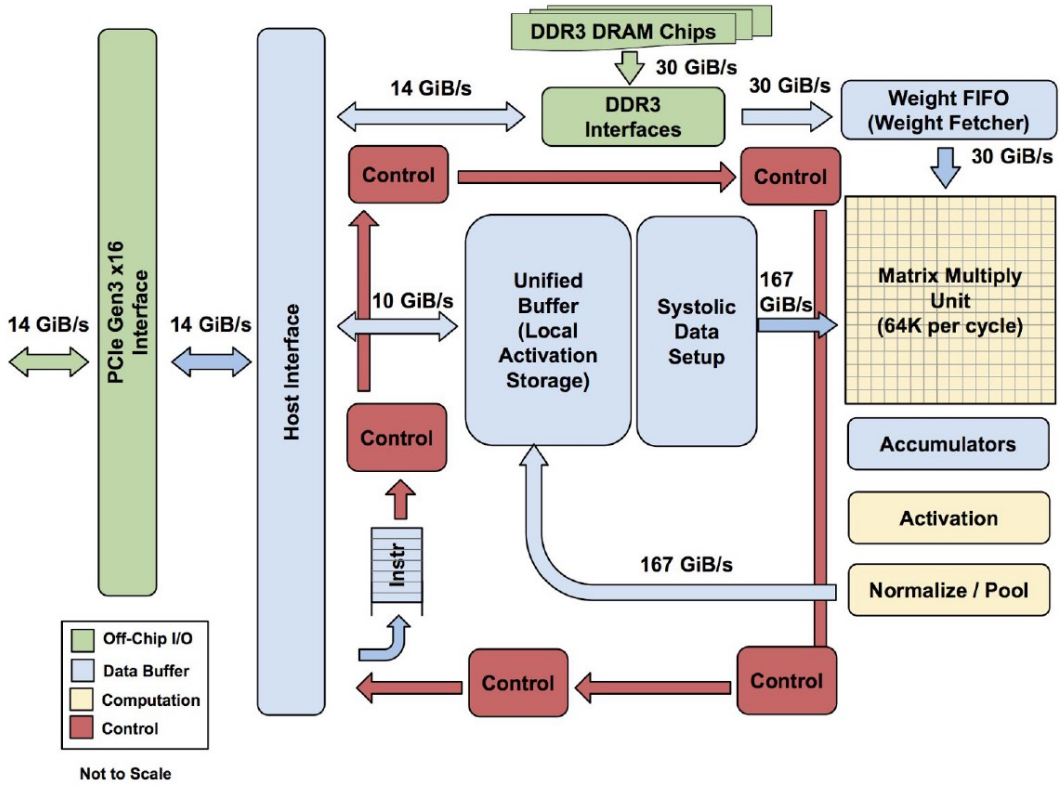
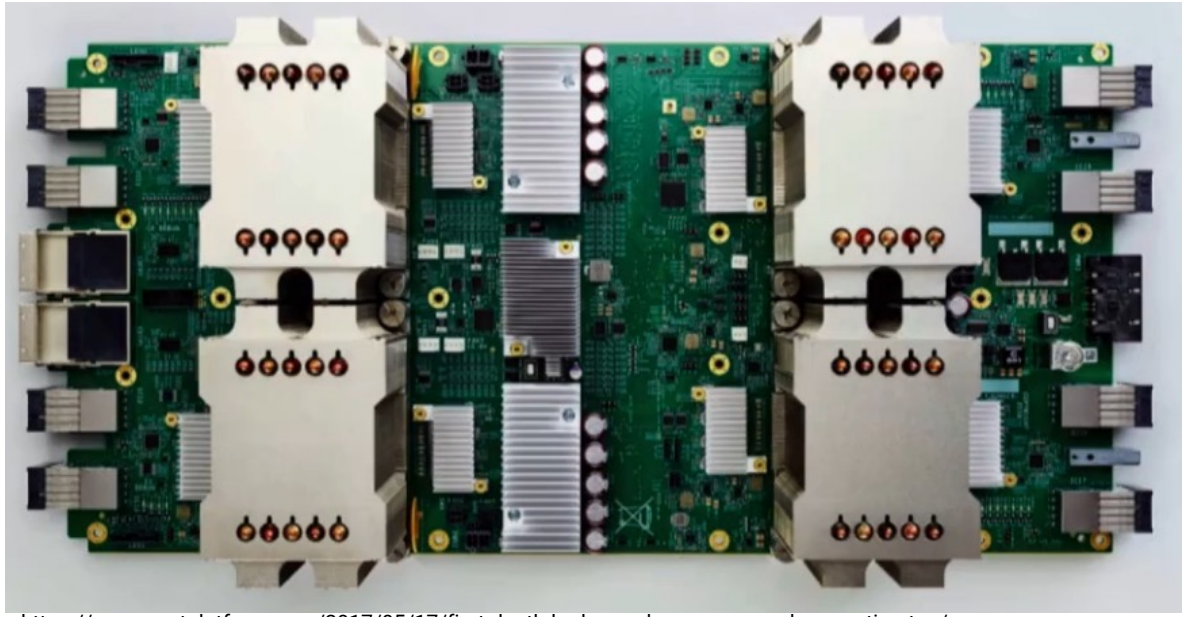


Figure 1. TPU Block Diagram. The main computation part is the yellow Matrix Multiply unit in the upper right hand corner. Its inputs are the blue Weight FIFO and the blue Unified Buffer (UB) and its output is the blue Accumulators (Acc). The yellow Activation Unit performs the nonlinear functions on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

An Example Modern Systolic Array: TPU2



https://www.nextplatform.com/2017/05/17/first-depth-look-googles-new-second-generation-tpu/

4 TPU chips
vs 1 chip in TPU1

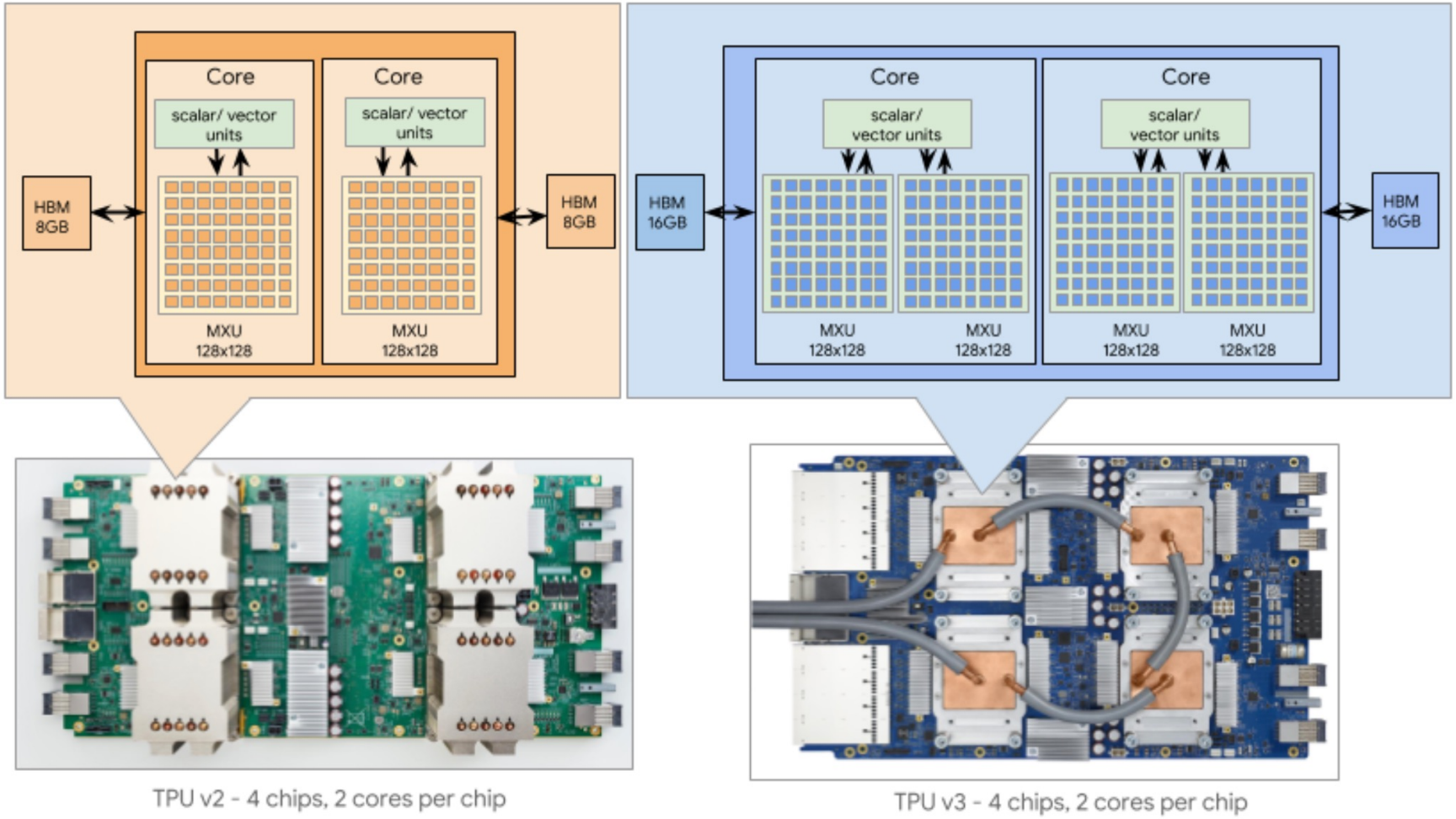
High Bandwidth Memory
vs DDR3

Floating point operations
vs FP16

45 TFLOPS per chip
vs 23 TOPS

Designed for training
and inference
vs only inference

An Example Modern Systolic Array: TPU3



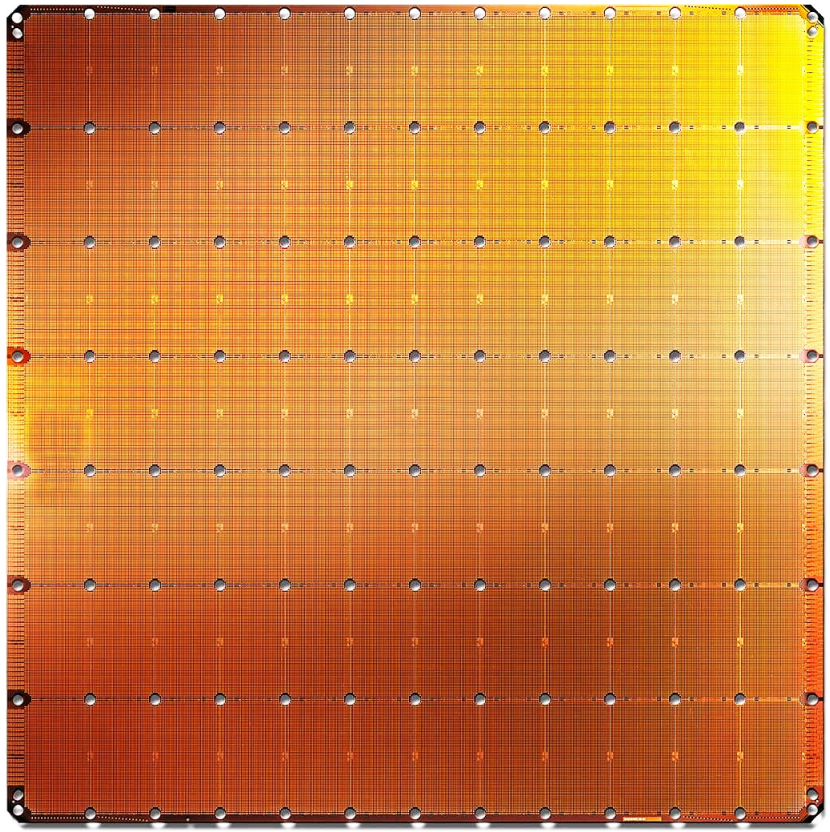
32GB HBM per chip
vs 16GB HBM in TPU2

4 Matrix Units per chip
vs 2 Matrix Units in TPU2

90 TFLOPS per chip
vs 45 TFLOPS in TPU2

https://cloud.google.com/tpu/docs/system-architecture

Cerebras's Wafer Scale Engine (2019)



- The largest ML accelerator chip
- 400,000 cores

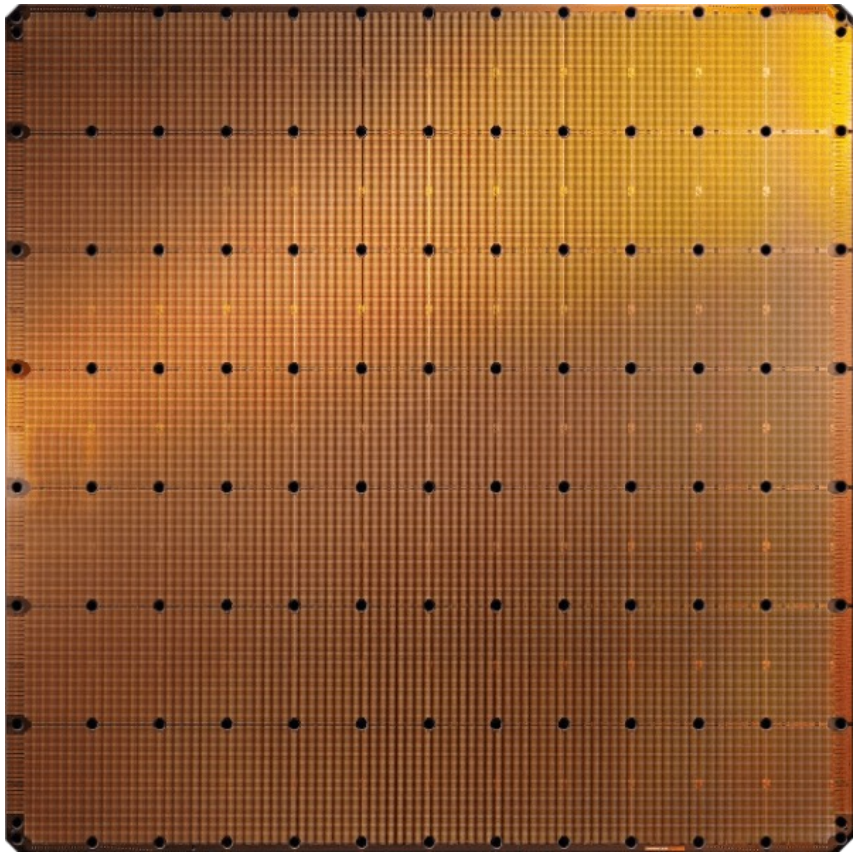
Cerebras WSE
1.2 Trillion transistors
46,225 mm^2

Largest GPU
21.1 Billion transistors
815 mm^2
NVIDIA TITAN V

https://www.anandtech.com/show/14758/hot-chips-31-live-blogs-cerebras-wafer-scale-deep-learning

https://www.cerebras.net/cerebras-wafer-scale-engine-why-we-need-big-chips-for-deep-learning/

Cerebras's Wafer Scale Engine-2 (2021)



- The largest ML accelerator chip
- 850,000 cores

Cerebras WSE-2
2.6 Trillion transistors
46,225 mm^2

Largest GPU
54.2 Billion transistors
826 mm^2
NVIDIA Ampere GA100

https://www.anandtech.com/show/14758/hot-chips-31-live-blogs-cerebras-wafer-scale-deep-learning

https://www.cerebras.net/cerebras-wafer-scale-engine-why-we-need-big-chips-for-deep-learning/

The End

北京大学 PEKING UNIVERSITY

- Summary
  - Systolic Array
  - Improvement of Programmability for Systolic Arrays
  - Latest AI Chips with Systolic Arrays