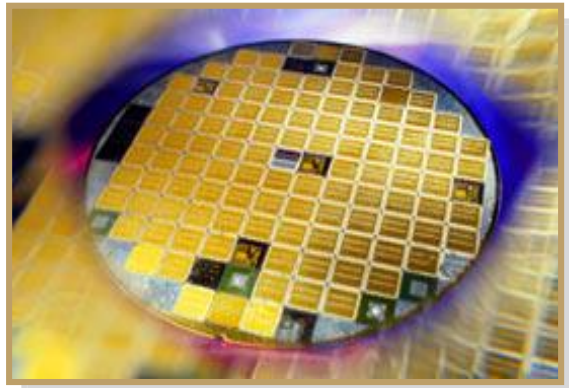
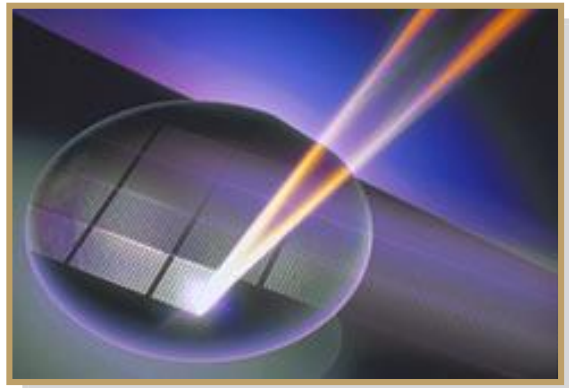
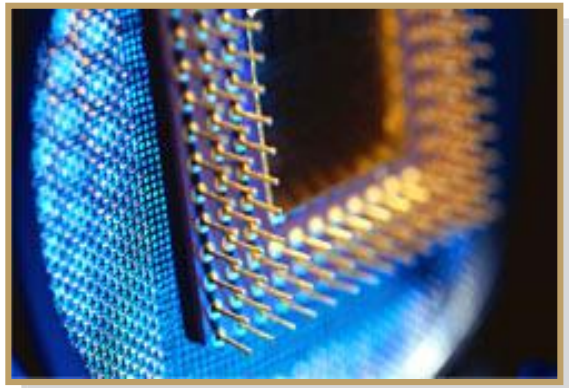
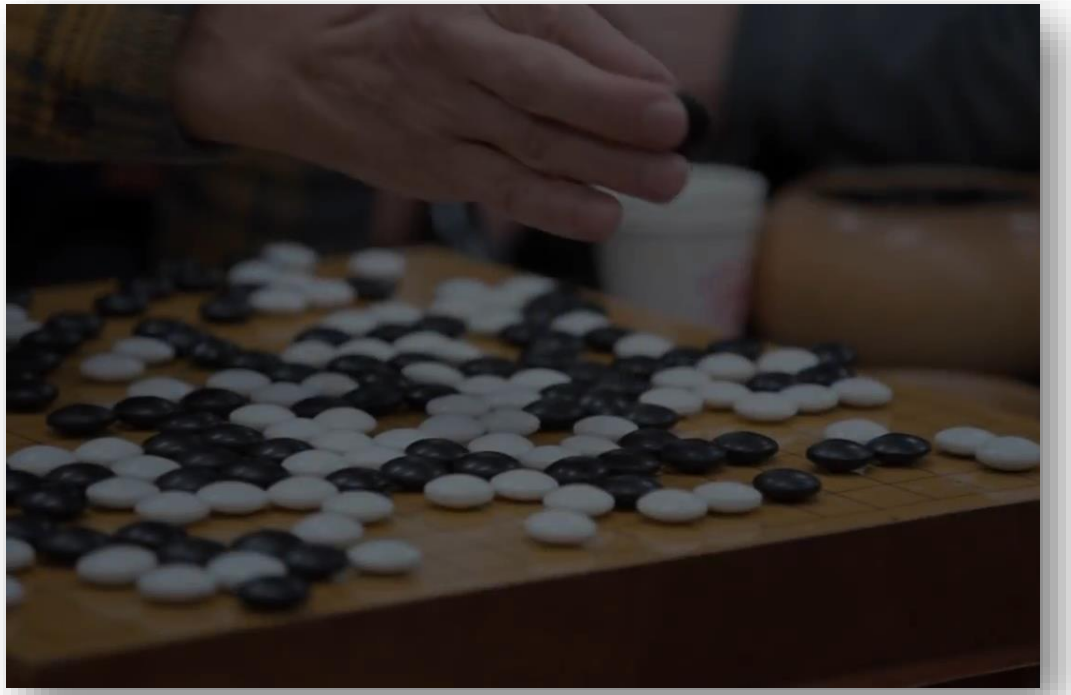
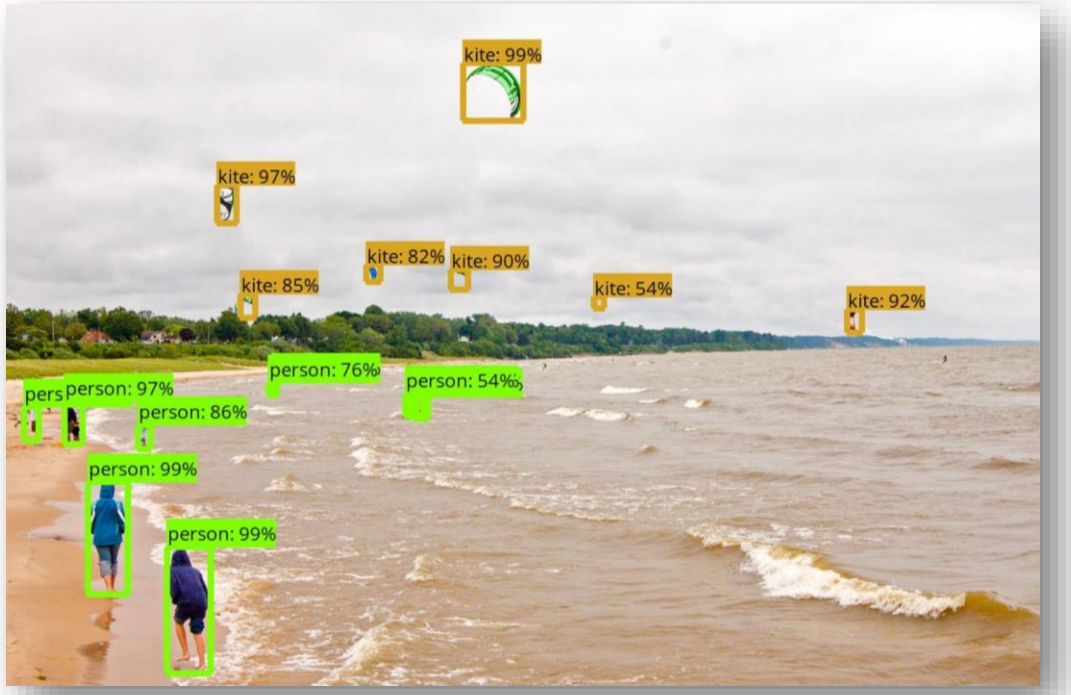
第七章：脉动架构设计

人工智能飞速发展

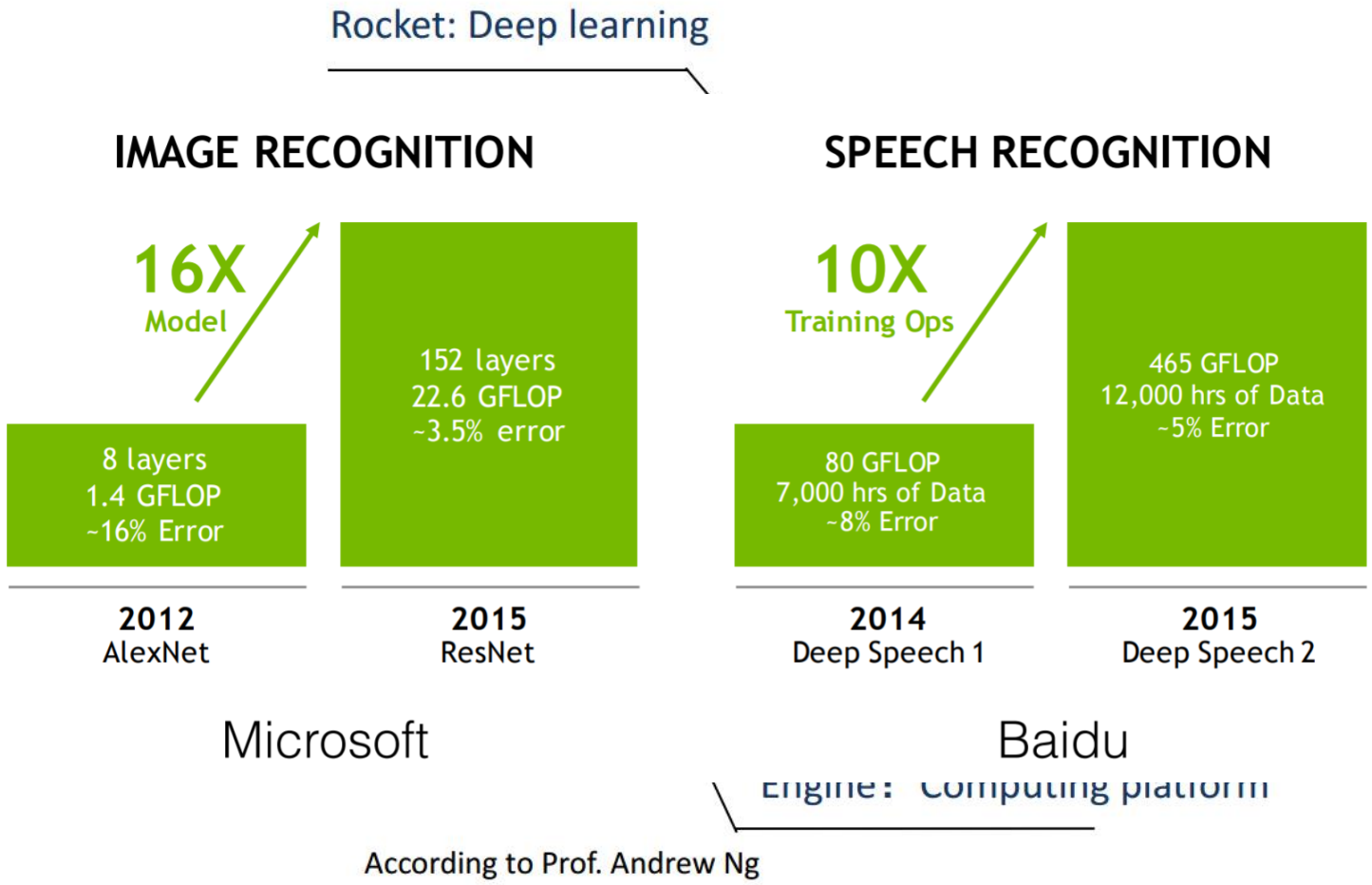


AlphaGo

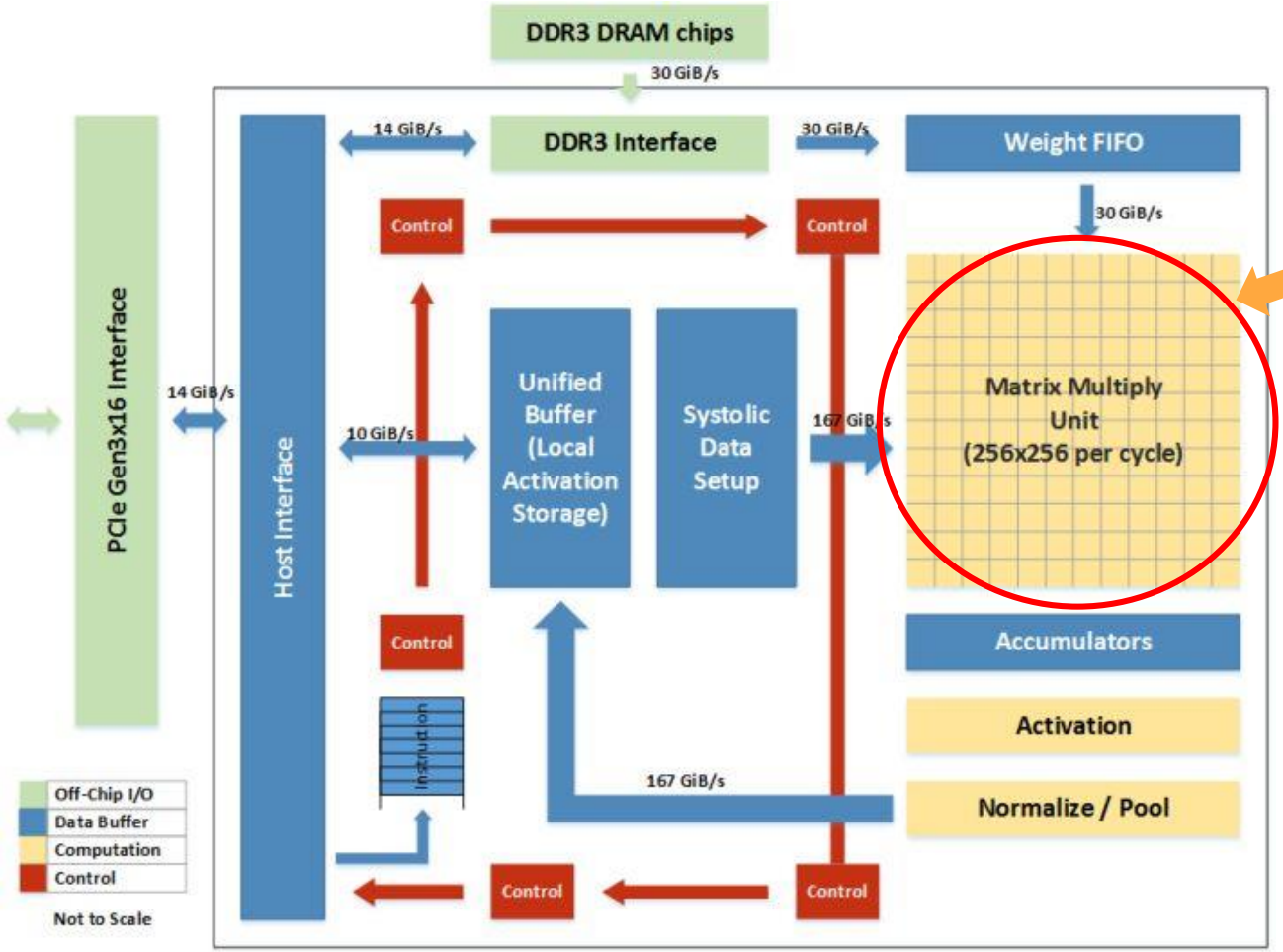


Object Detection

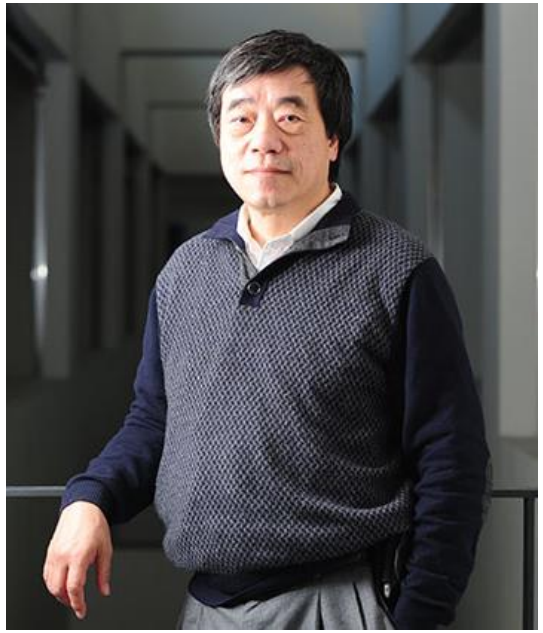
计算性能需求


Rocket: Deep learning
IMAGE RECOGNITION
16X
Model
8 layers
1.4 GFLOP
~16% Error
152 layers
22.6 GFLOP
~3.5% error
2012
AlexNet
2015
ResNet
Microsoft
SPEECH RECOGNITION
10X
Training Ops
80 GFLOP
7,000 hrs of Data
~8% Error
465 GFLOP
12,000 hrs of Data
~5% Error
2014
Deep Speech 1
2015
Deep Speech 2
Baidu
Engine: Computing platform
According to Prof. Andrew Ng

TPU架构



脉动阵列



Google TPU架构

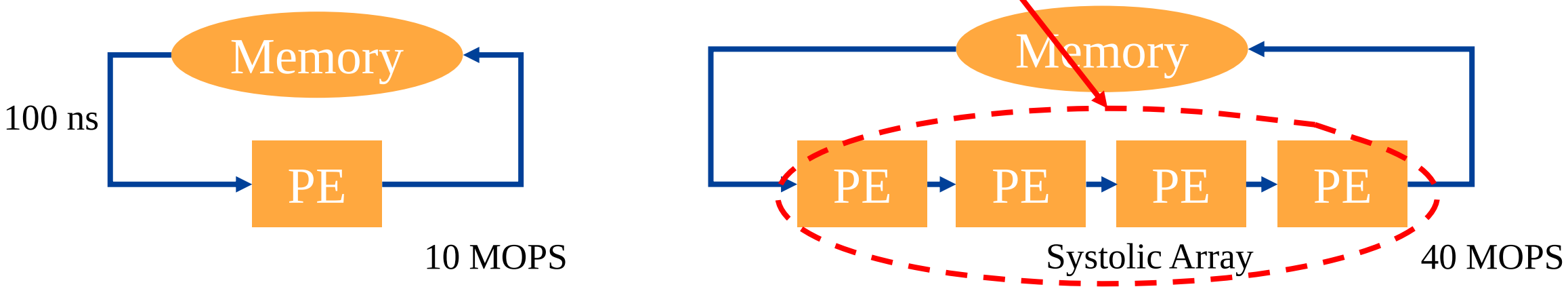
孔祥重（H.T. Kung）教授
1978年提出

目录

01.脉动架构的基本概念

脉动架构：也称脉动阵列 (Systolic Array)

- 有节奏地计算、传输数据的处理单元(PE)网络
- 类似于血液在心脏泵力下通过动脉血管的流动，因此命名为“脉动”(systolic)



- 每一个节点，都是相同的；
- 每个PE只与其相邻的PE进行通信，也就是说PE之间的通信具有局部性，而且通信是规则的；
- 每个PE都有其局部的存储器。

脉动架构特点

典型情况下，脉动阵列的所有PE是相同、且全流水的，即PE的所有通信边沿都包含延迟单元，且整个系统通常只包含局部互连

PE规则地泵入、泵出数据，以维持规则的数据流

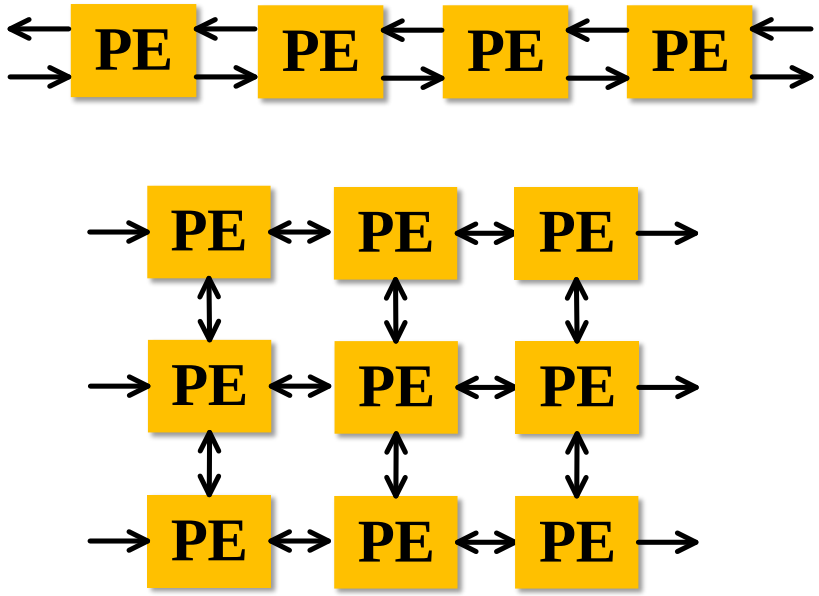
是单指令多数据(SIMD)系统，特别适于对大量规则数据流进行单一计算、处理的场合，通过全流水大大提高吞吐率，减少数据带宽

架构特征是模块化和规则化，特别适于VLSI实现

01.脉动架构的基本概念——脉动架构的种类与应用

脉动架构的种类

- 一维(上)；二维(中)；多维(下)



脉动架构的应用

信号、图像处理	IIR、FIR滤波器、卷积、相关 DFT、DCT、DWT、内插、媒体滤波器
矩阵运算	矩阵 - 矢量、矩阵 - 矩阵乘； 矩阵三角化； 矩阵分解
非数字应用	图论算法实现 数据结构 语言、字符识别 编解码器 关系数据库操作

目录

脉动架构的设计方法


绘制算法的规则依赖图 （Regular DG）
添加投影、 PE空间矢量和调度矢量
（空间-时间表示）
边缘映射
构造最终脉动阵列电路

02.脉动架构的设计方法——依赖图(Dependence graph)

依赖图(DG)是描述算法中运算关系的有向图。

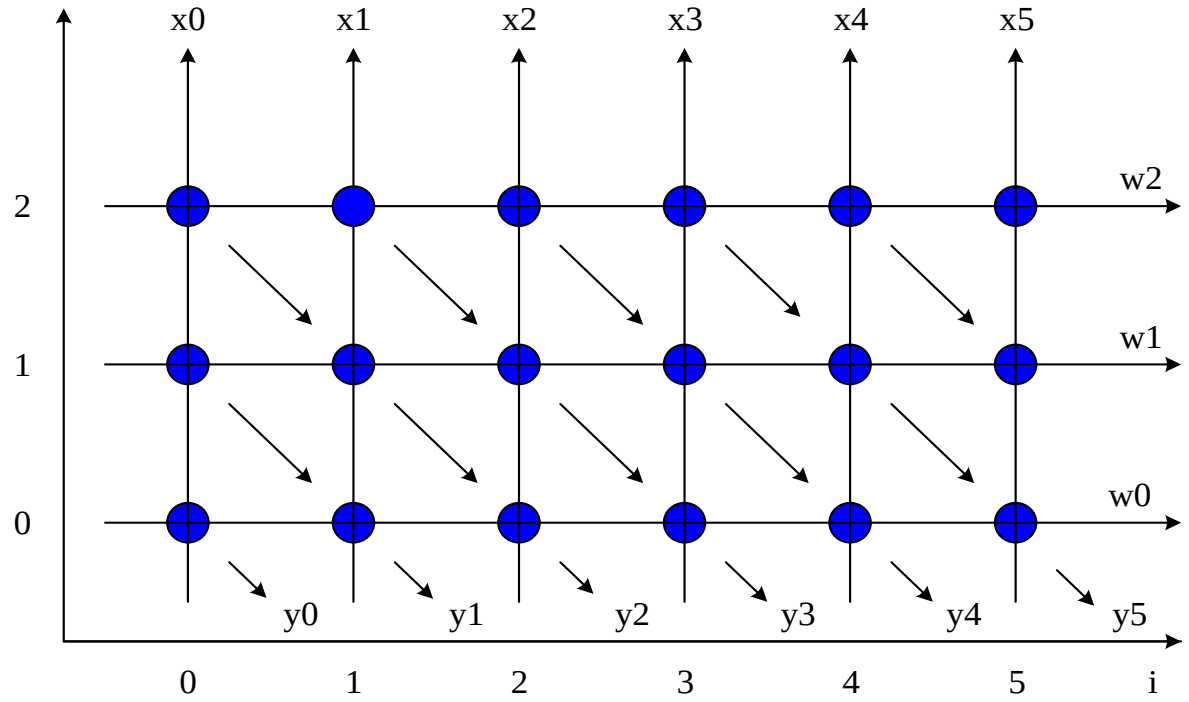
- **DG中的节点代表计算**
- **DG中边代表节点间的优先约束**

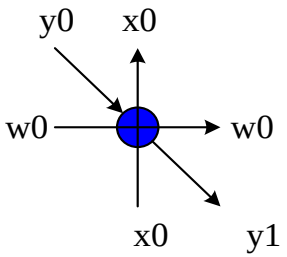
依赖图(DG)	数据流图(DFG)
DG包含所有循环的计算	DFG只包含一个循环的计算
DG不包含延时单元	DFG包含延时单元

- **DG用于脉动阵列设计**

02.脉动架构的设计方法——依赖图(Dependence graph)

$$y(n)=\omega_0 x(n)+\omega_1 x(n-1)+\omega_2 x(n-2)$$





y1=y0+w0x0

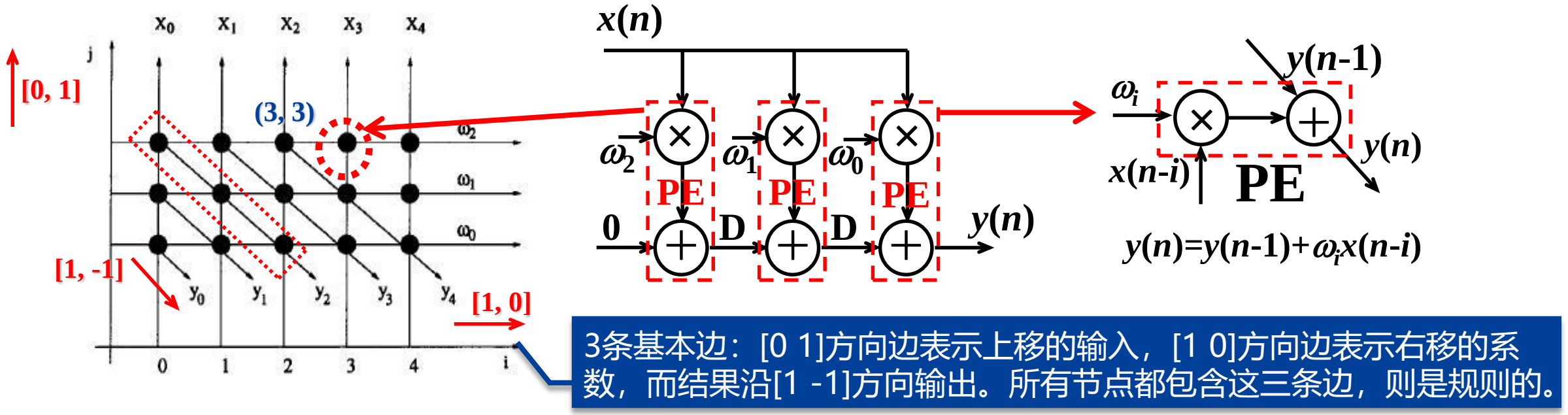
- **规则DG：若在DG中的任意节点沿某方向的一个边代表DG中所有节点沿相同方向的边，称为规则DG。2维坐标系中，节点标号为(i, j)，[1,0]-x方向；[0,1]-y方向。**

02.脉动架构的设计方法

脉动架构是在规则依赖图(Regular DG)上，利用线性映射(Linear Mapping) 设计的。

例：3阶FIR滤波器，DFG示于下图中间，确定PE为下图右，DG为下图左

$$y(n) = \omega_0 x(n) + \omega_1 x(n-1) + \omega_2 x(n-2)$$

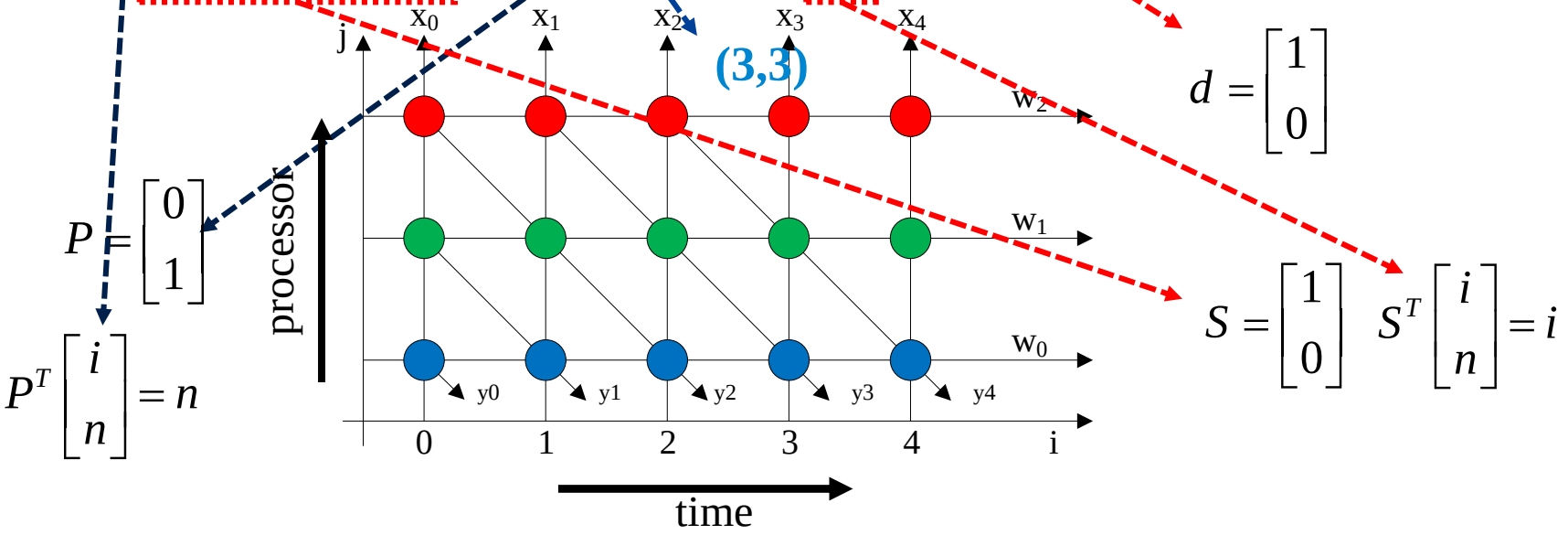


3条基本边：[0 1]方向边表示上移的输入，[1 0]方向边表示右移的系数，而结果沿[1 -1]方向输出。所有节点都包含这三条边，则是规则的。

02.脉动架构的设计方法——定义

矢量定义

- 节点标号矢量：$I^T = (i, j)$
- 投影矢量(也称迭代矢量)：$d^T = (d_1, d_2)$ 给出投影方向，长度为该方向上节点间距
- PE空间矢量(也称处理器矢量)：$p^T = (p_1\ p_2)$，节点$I^T = (i, j)$被分配到标号为$p^T I = (p_1 i + p_2 j)$的PE执行
- 调度矢量：$s^T = (s_1, s_2)$，节点I在时刻$s^T I$被PE执行。



02.脉动架构的设计方法——定义

硬件利用效率Hardware Utilization Efficiency (HUE)：

- Let x and y be computations with index vectors $I(x)$, $I(y)$ that satisfy $I(x) = I(y) + d$. Then they are executed on the same PE. Let S_x be the time at which x is scheduled and S_y be the time at which y is scheduled. Then $S_x = S_y + s^T d$.

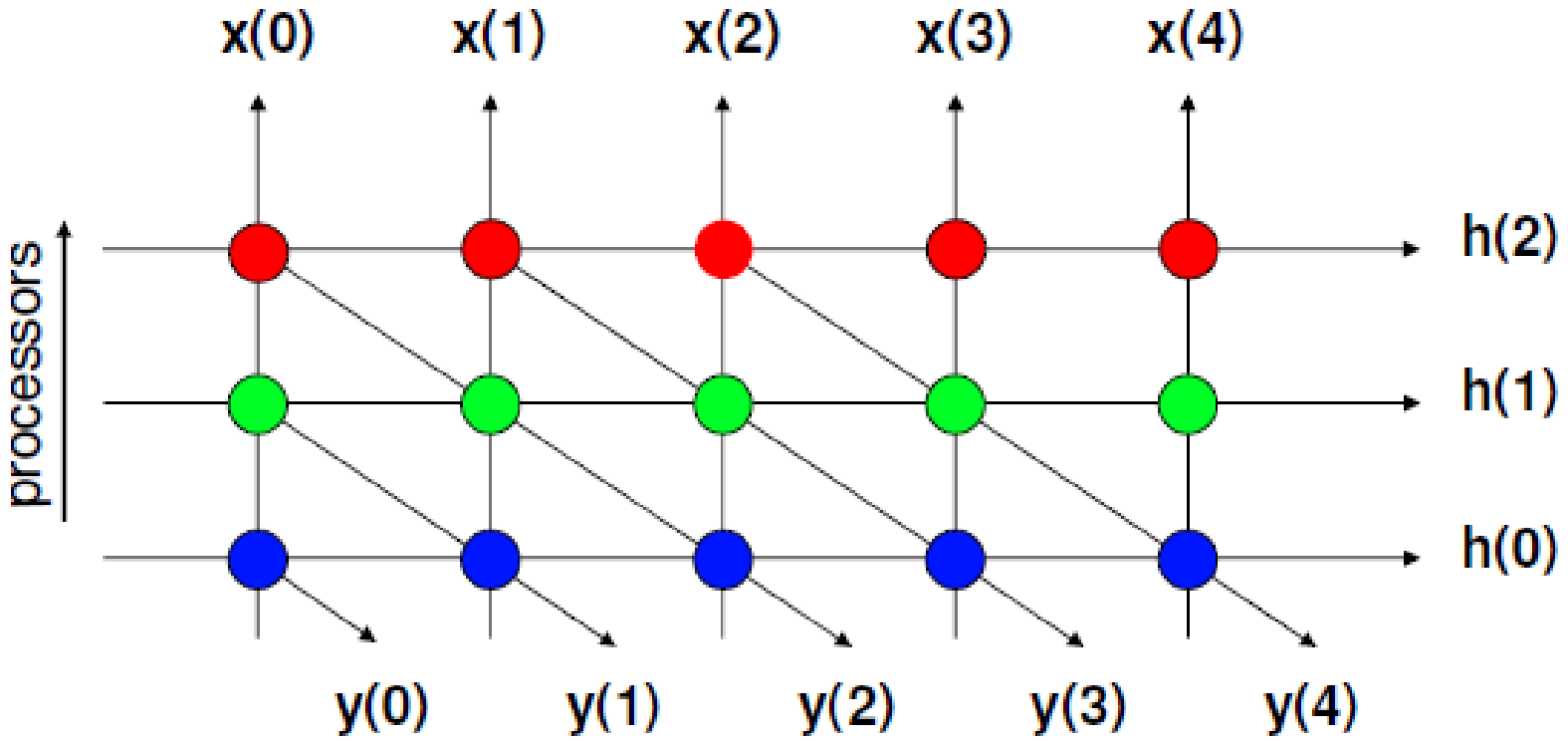
- Hence, any PE executes at most 1 computation per $s^T d$ time slots. So

$$HUE = 1/|s^T d|$$

相邻两个任务被同一PE执行的时间间隔$|s^T d|$越大，表明PE的利用率越低

02.脉动架构的设计方法

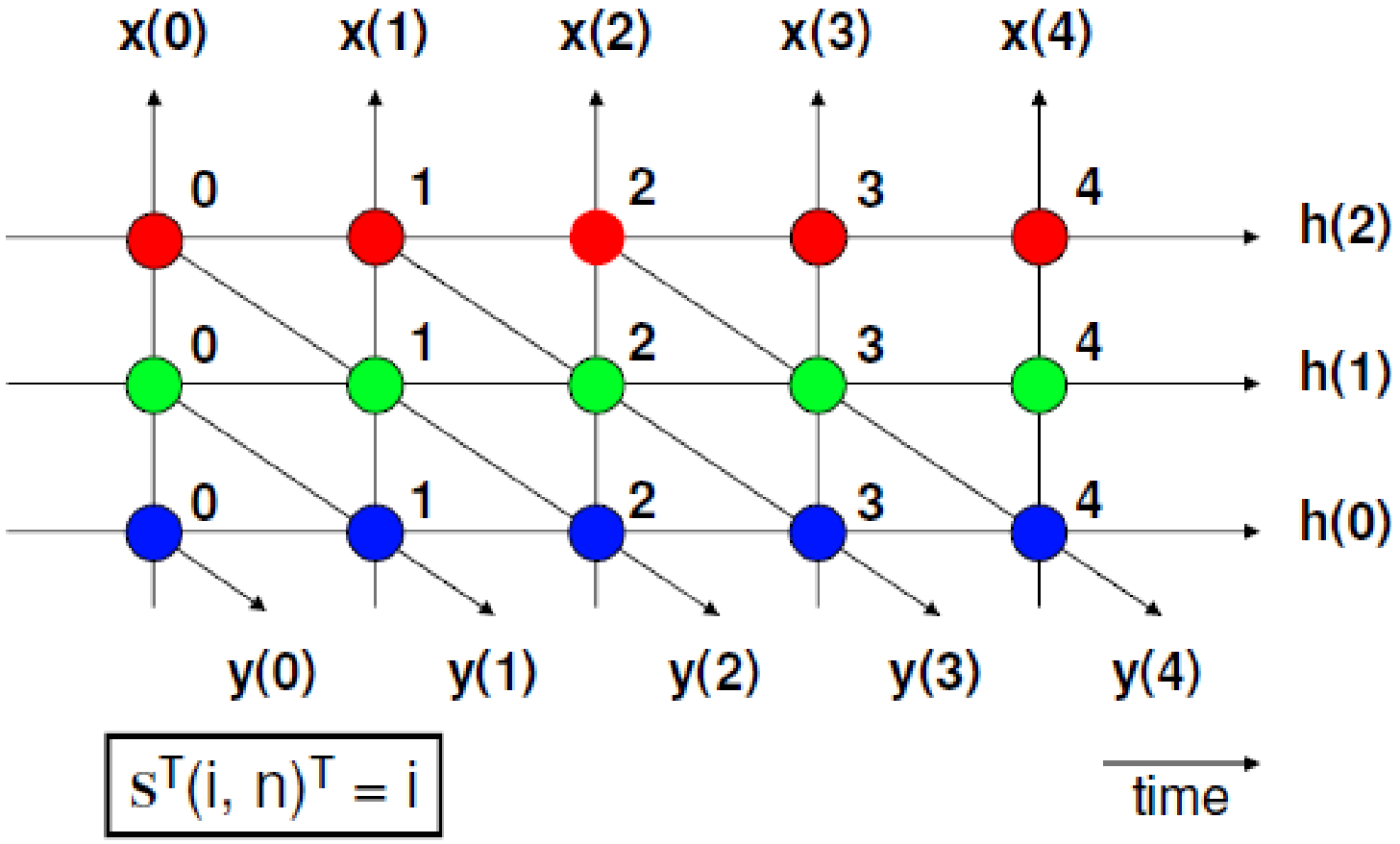
Processor Allocation $p^T = (0, 1),\ d^T = (1, 0)$



$\mathbf{p}^T (i, n)^T = n$

02.脉动架构的设计方法

Scheduling $S^T = (1, 0),\ d^T = (1, 0)$



02.脉动架构的设计方法——映射的可行性约束

- $P^T d = 0$：PE空间矢量P^T和投影矢量d必须互相正交

- $s^T d \neq 0$：映射到同一PE的节点A和B，不能同时执行，即$s^T\ IA \neq s^T\ IB$，或s^T与P^T不平行

- **边的映射**：若DG(空间表示)中存在边e，则在脉动阵列相应PE间的边沿$(p^T e)$方向，延迟为$s^T e \geq 0$(延迟为0的变量为广播变量)

02.脉动架构的设计方法

把空间表示DG变换为空间 - 时间表示的方法

- 把DG中的一个空间维翻译为时间维
- 对2维DG，一般的变换描述为$i' = t = 0$， $j' = p^T I$， $t' = s^T I$，即：

$$\begin{bmatrix} i' \\ j' \\ t' \end{bmatrix} = \mathrm{T} \begin{bmatrix} i \\ j \\ t \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ P^T & & 0 \\ S^T & & 0 \end{bmatrix} \begin{bmatrix} i \\ j \\ t \end{bmatrix}$$

- j': PE 的标号；t':被调度的时刻

脉动架构的设计方法


绘制算法的规则依赖图 （Regular DG）
添加投影、 PE空间矢量和调度矢量
（空间-时间表示）
边缘映射
构造最终脉动阵列电路

02.脉动架构的设计方法——添加空间和调度矢量

Applying Processor and Scheduling

Processor Vector
$p^T = [0\ 1]$

Projection Vector
$d^T = [1\ 0]$

Scheduling Vector
$s^T = [1\ 0]$

apply

Part of DG:

$[\begin{smallmatrix}0\\2\end{smallmatrix}]$ $p^T I = [0\ 1][\begin{smallmatrix}0\\2\end{smallmatrix}]=2$ $s^T I = [1\ 0][\begin{smallmatrix}0\\2\end{smallmatrix}]=0$

$[\begin{smallmatrix}1\\2\end{smallmatrix}]$ $p^T I = [0\ 1][\begin{smallmatrix}1\\2\end{smallmatrix}]=2$ $s^T I = [1\ 0][\begin{smallmatrix}1\\2\end{smallmatrix}]=1$

$[\begin{smallmatrix}0\\1\end{smallmatrix}]$ $p^T I = [0\ 1][\begin{smallmatrix}0\\1\end{smallmatrix}]=1$ $s^T I = [1\ 0][\begin{smallmatrix}0\\1\end{smallmatrix}]=0$

$[\begin{smallmatrix}1\\1\end{smallmatrix}]$ $p^T I = [0\ 1][\begin{smallmatrix}1\\1\end{smallmatrix}]=1$ $s^T I = [1\ 0][\begin{smallmatrix}1\\1\end{smallmatrix}]=1$

$[\begin{smallmatrix}0\\0\end{smallmatrix}]$ $p^T I = [0\ 1][\begin{smallmatrix}0\\0\end{smallmatrix}]=0$ $s^T I = [1\ 0][\begin{smallmatrix}0\\0\end{smallmatrix}]=0$

$[\begin{smallmatrix}1\\0\end{smallmatrix}]$ $p^T I = [0\ 1][\begin{smallmatrix}1\\0\end{smallmatrix}]=0$ $s^T I = [1\ 0][\begin{smallmatrix}1\\0\end{smallmatrix}]=1$

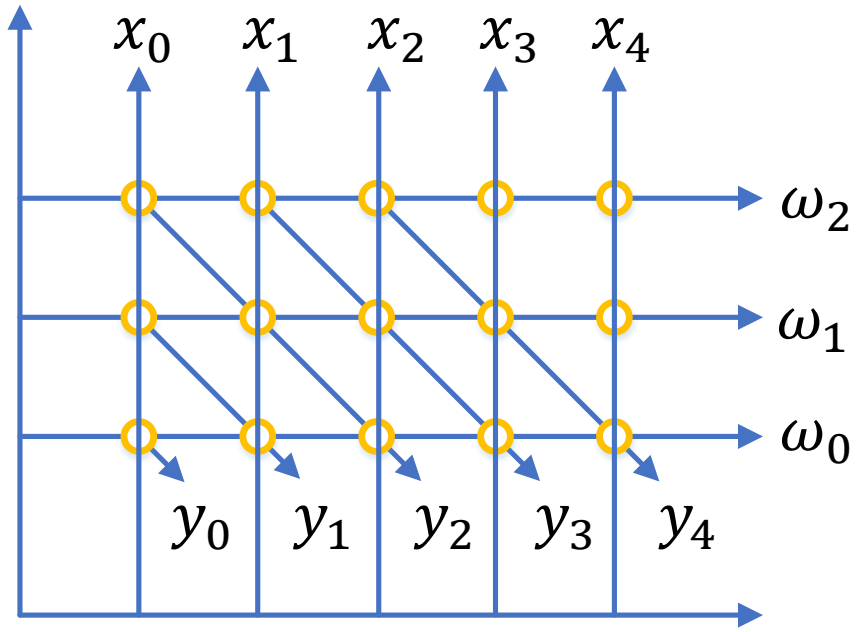
PE2
Processor 2

PE1
Processor 1

PE0
Processor 0

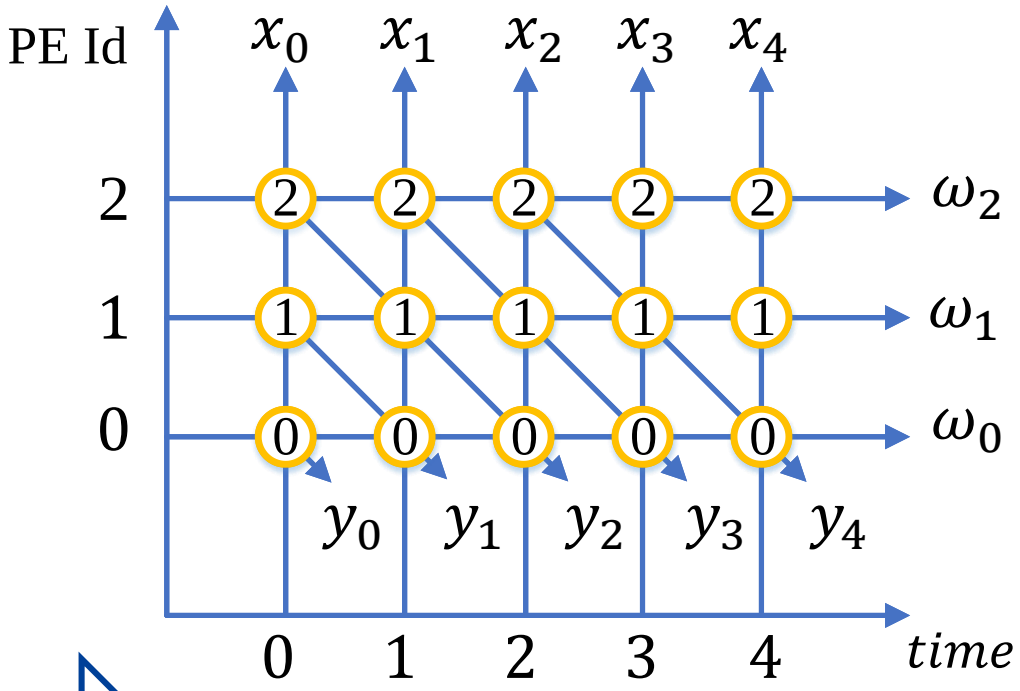
02.脉动架构的设计方法——添加空间和调度矢量

Applying Processor and Scheduling



Dependence Graph

Applying projection and scheduling



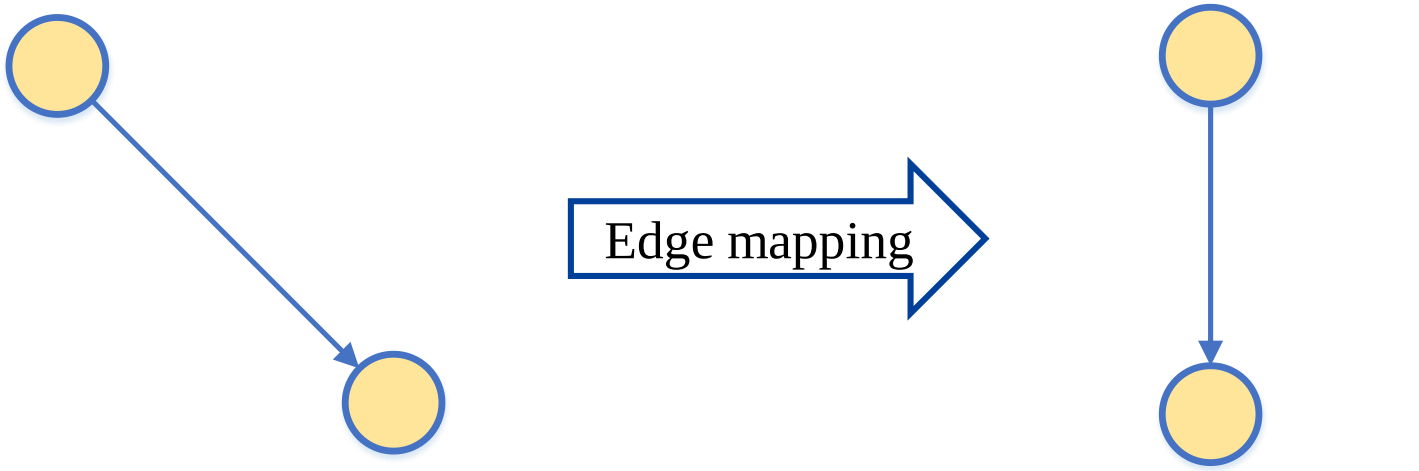
Space-time representation

脉动架构的设计方法


绘制算法的规则依赖图 （Regular DG）
添加投影、 PE空间矢量和调度矢量
（空间-时间表示）
边缘映射
构造最终脉动阵列电路

02.脉动架构的设计方法

Edge Mapping



$$e = \begin{bmatrix} i \\ j \end{bmatrix}$$

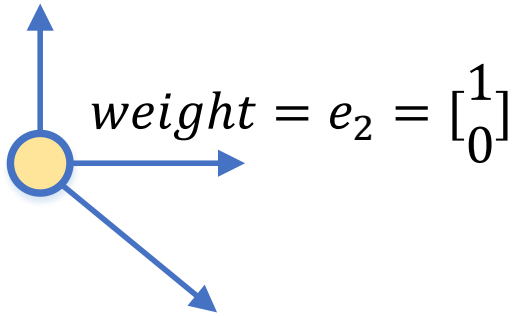
$$e' = p^T e$$

$$\text{Delay} = s^T e$$

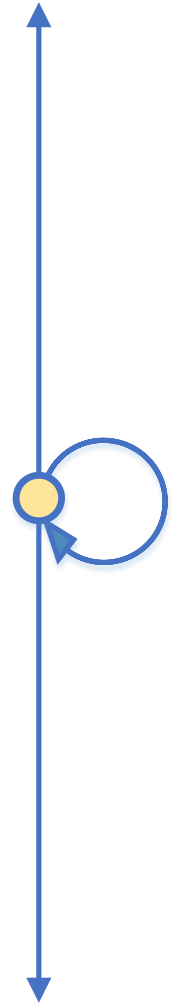
Edge Mapping

Example:

$input = e_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$



Edge mapping



$e_1' = p^T e_1 = [0 \ \ 1] \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 1$

$delay_{e1} = s^T e_1 = [1 \ \ 0] \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 0$

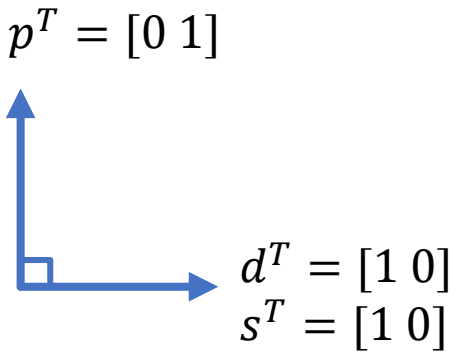
$e_2' = p^T e_2 = [0 \ \ 1] \begin{bmatrix} 1 \\ 0 \end{bmatrix} = 0$

$delay_{e2} = s^T e_2 = [1 \ \ 0] \begin{bmatrix} 1 \\ 0 \end{bmatrix} = 1$

$e_3' = p^T e_3 = [0 \ \ 1] \begin{bmatrix} 1 \\ -1 \end{bmatrix} = -1$

$delay_{e3} = s^T e_3 = [1 \ \ 0] \begin{bmatrix} 1 \\ -1 \end{bmatrix} = 1$

Edge Mapping



e	$p^T e$	$s^T e$
Input $[0\ 1]^T$	1	0
Weight $[1\ 0]^T$	0	1
Output $[1\ -1]^T$	-1	1

Edge mapping table

脉动架构的设计方法

绘制算法的规则依赖图 （Regular DG）

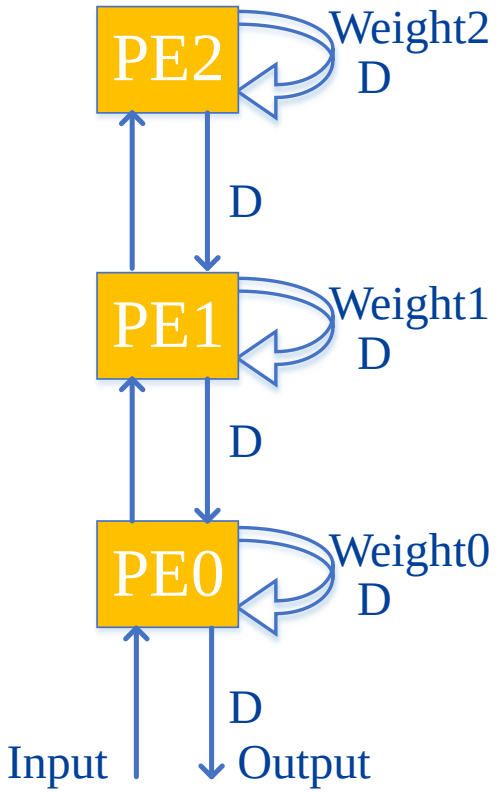
添加投影、 PE空间矢量和调度矢量

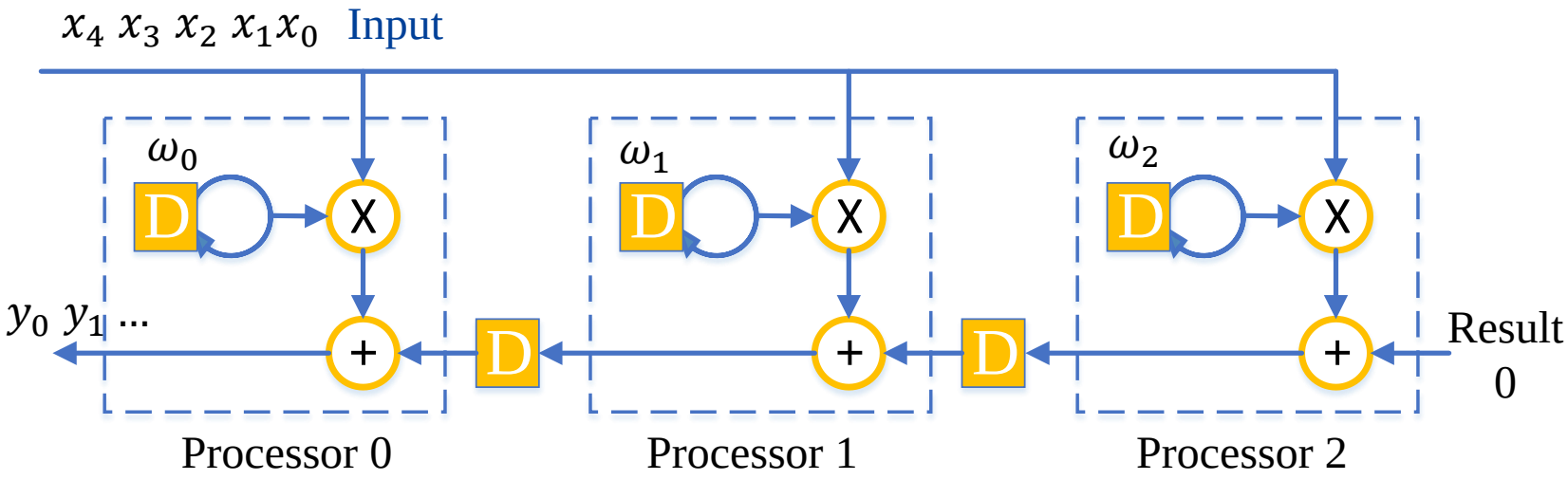
（空间-时间表示）

边缘映射

构造最终脉动阵列电路

Construct the Final Systolic Architecture





This is called B_1 design.

目录

03.脉动架构的应用——FIR systolic arrays-B_1

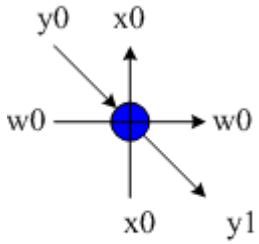
Design B_1 (Broadcast Inputs, Move Results, Weights Stay)

- Selecting $d^T = (1\ 0)$, $p^T = (0\ 1)$, $s^T = (1\ 0)$
- Any node with index $I^T = (i\ j)$
  - is mapped to processor $p^T I = j$. All nodes on horizontal line are mapped to the same processor.
  - is executed at time $s^T I = i$.
- Since $s^T d = 1$, we have $HUE = 1/|s^T d| = 1$.

03.脉动架构的应用——FIR systolic arrays-B_1

Edge mapping:

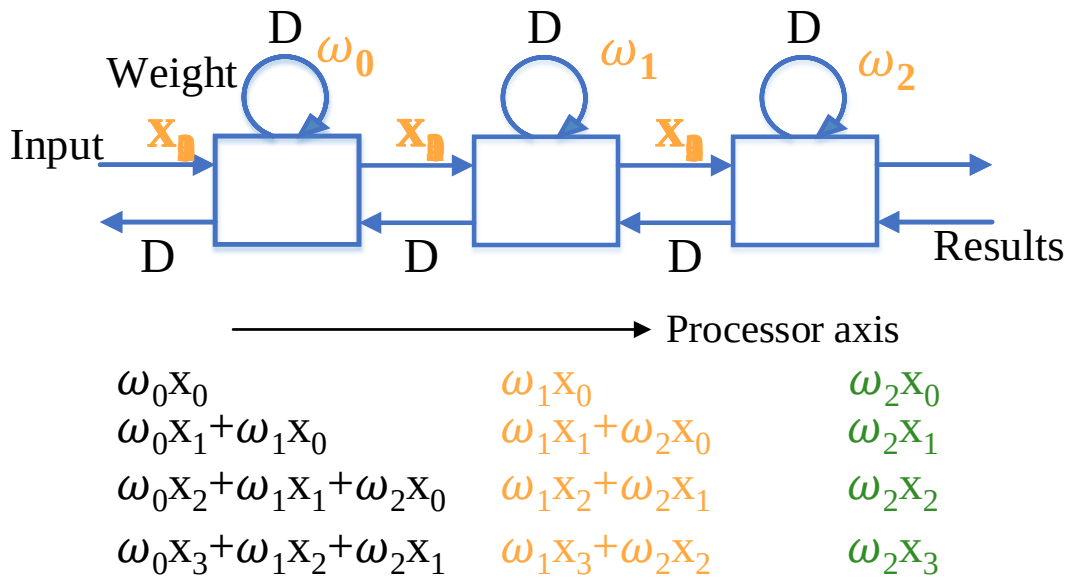
- **The 3 fundamental edges corresponding to weight, input, and result can be mapped to corresponding edges in the systolic array according to the following table:**

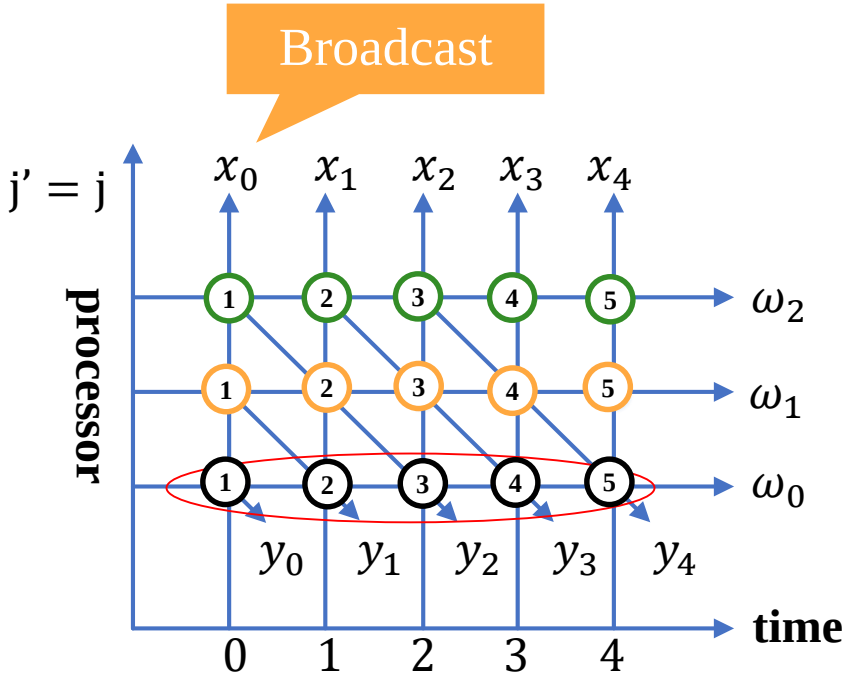


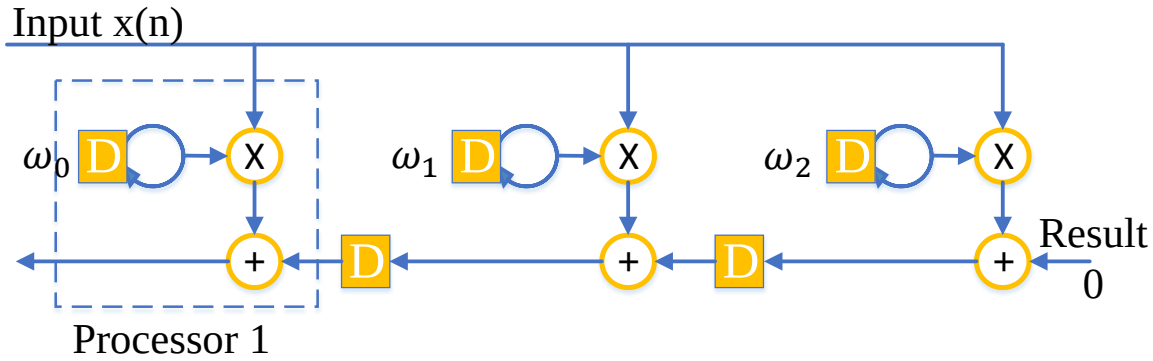
y1=y0+w0x0

e	$p^T e$	In block diagram	$s^T e$	In block diagram
$e_{weight}{}^T$=(1 0)	0	Weights stay	1	With delay
$e_{input}{}^T$=(0 1)	1	Input forward	0	Without delay
$e_{result}{}^T$=(1 -1)	-1	Result backward	1	With delay

03.脉动架构的应用——FIR systolic arrays-B_1



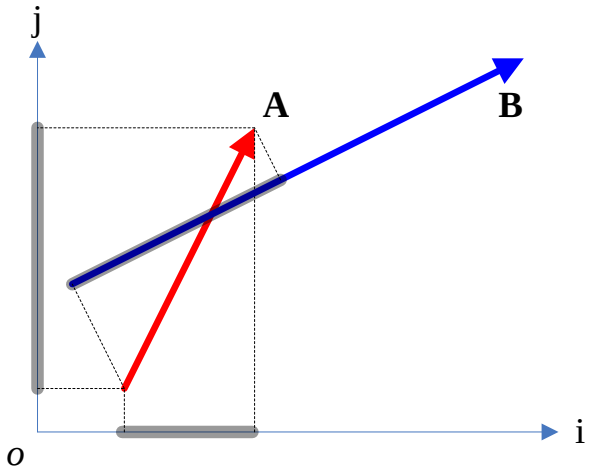


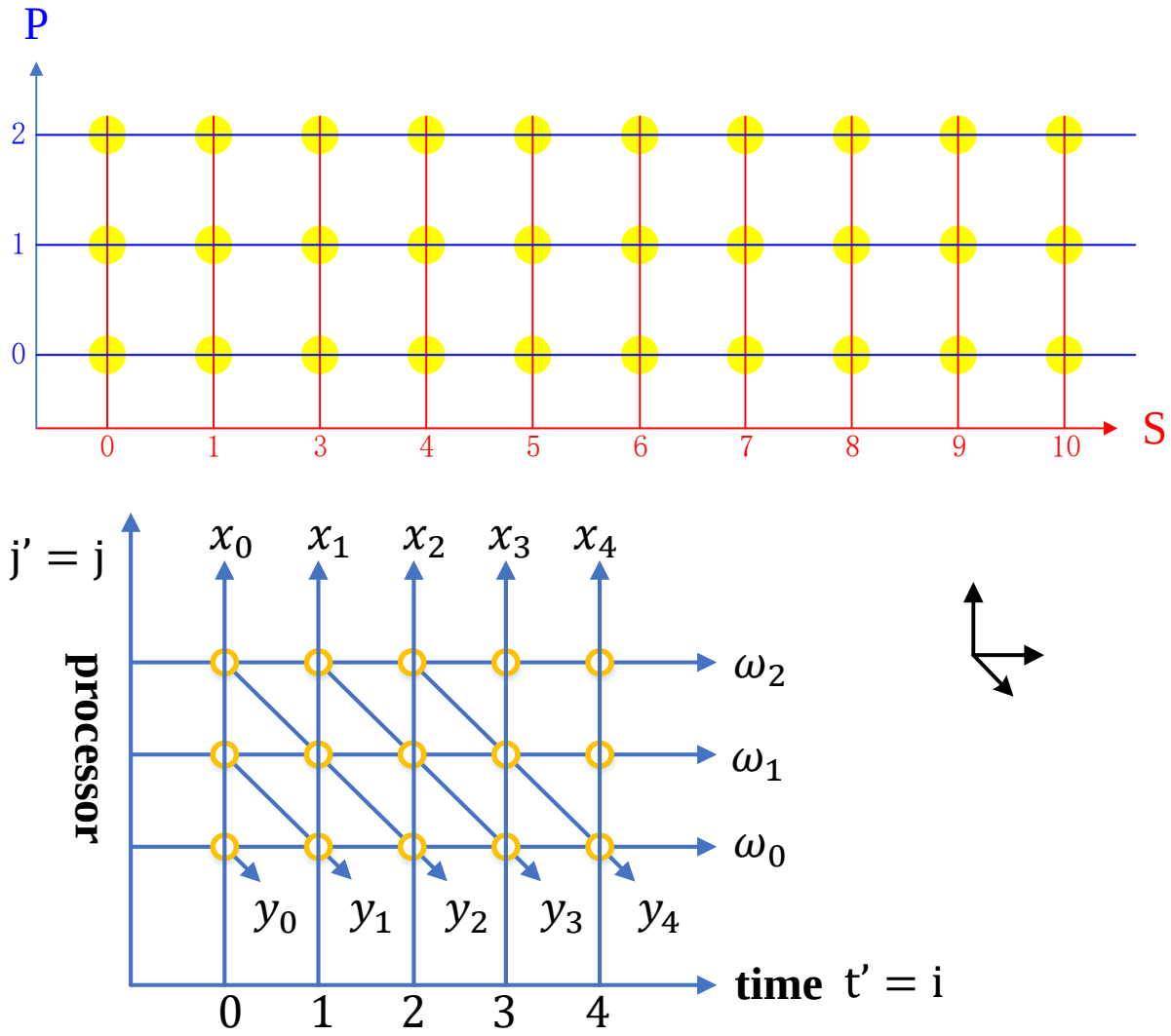


- Node I^T=(i j) is mapped to processor $p^T I = j$.
- Node I^T=(i j) is executed at time $s^T I = i$.

03.脉动架构的应用——FIR systolic arrays-B_1

Projection





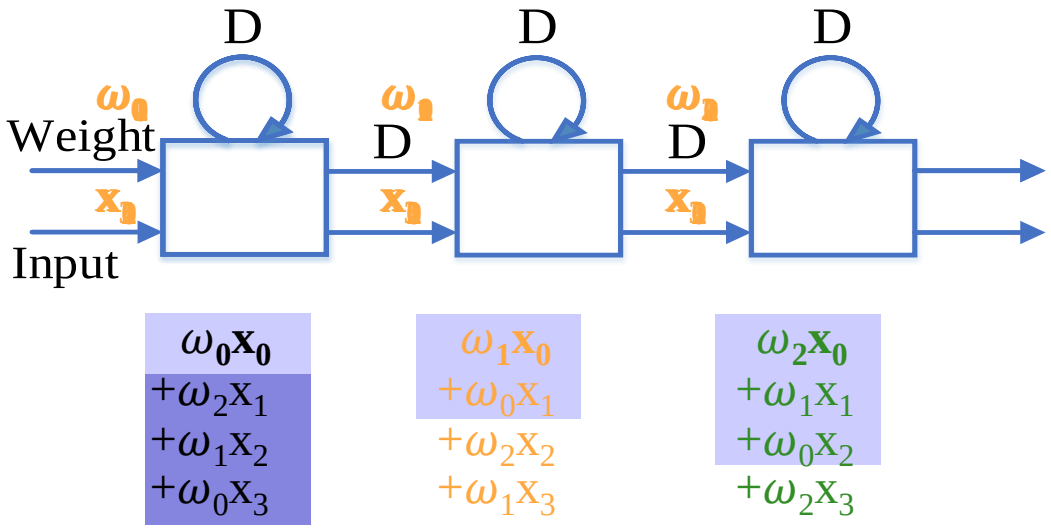
03.脉动架构的应用——FIR systolic arrays-B_2

Design B_2 (Broadcast Inputs, Move Weights, Results Stay)

- Selecting $d^T = (1 \quad -1)$, $pT = (1\ 1)$, $s^T = (1\ 0)$
- Any node with index $I^T = (i\ j)$
  - is mapped to processor $P^T I = i + j$.
  - is executed at time $s^T I = i$.
- Since $s^T d = 1$, we have $HUE = 1/|s^T d| = 1$.

03.脉动架构的应用——FIR systolic arrays-B_2

Edge mapping:

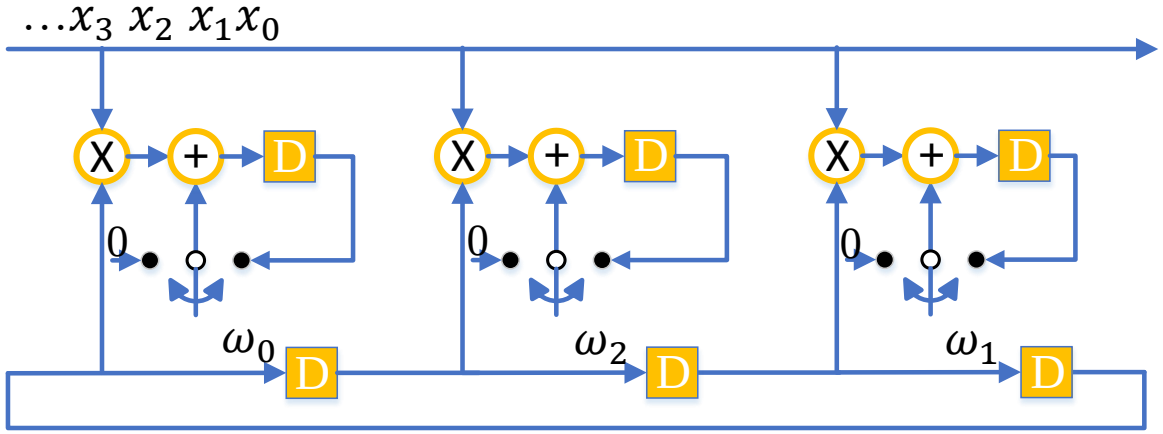


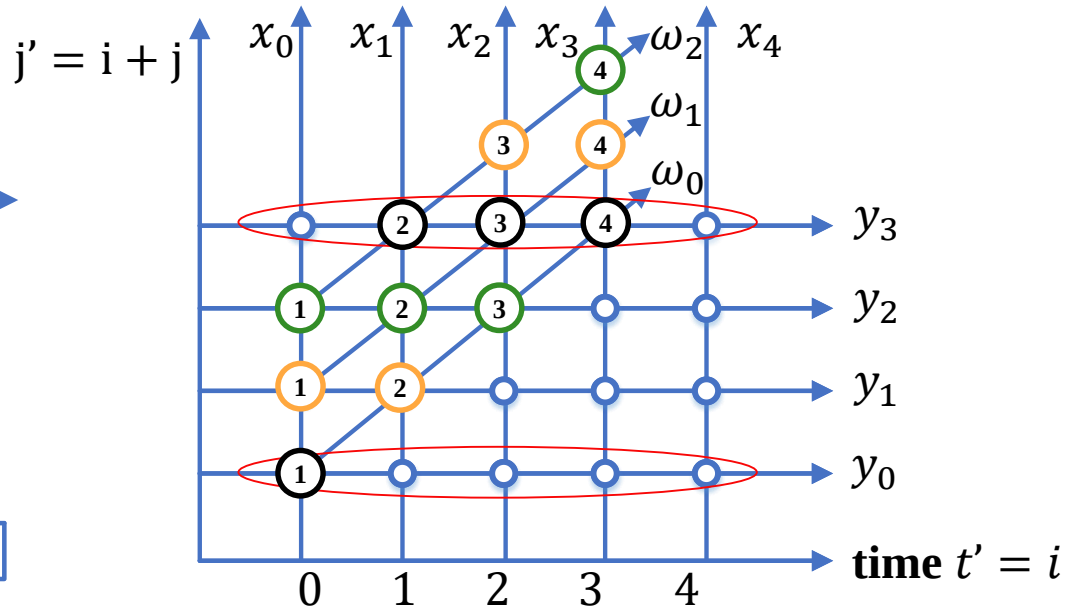
e	$p^T e$	$s^T e$
Weight $[1\ 0]^T$	1	1
i/p $[0\ 1]^T$	1	0
Result $[1\ -1]^T$	0	1

Applying space-time transformation, we get:

$j' = p^T(i\ j)^T = i + j, \qquad t' = s^T(i\ j)^T = i.$

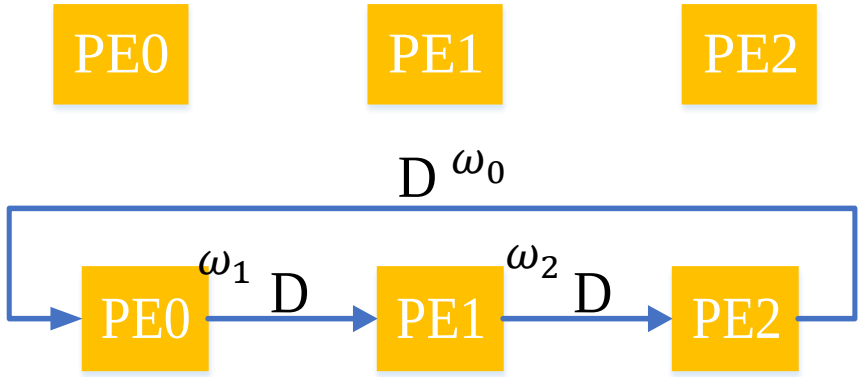
Time axis maintain, others rotate.



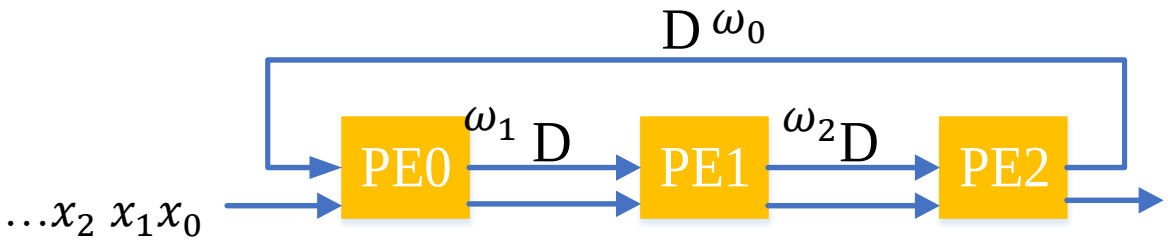


03.脉动架构的应用——FIR systolic arrays-B_2

Procedure



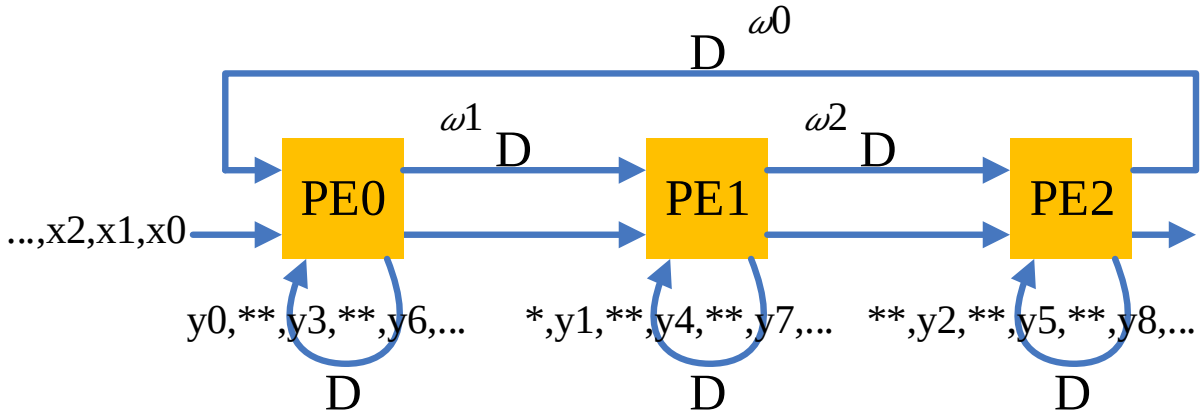
$$w = [1,0] \quad S \cdot \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}1\\0\end{bmatrix} \cdot \begin{bmatrix}1\\0\end{bmatrix} = 1 \quad P \cdot \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}1\\1\end{bmatrix} \cdot \begin{bmatrix}1\\0\end{bmatrix} = 1$$



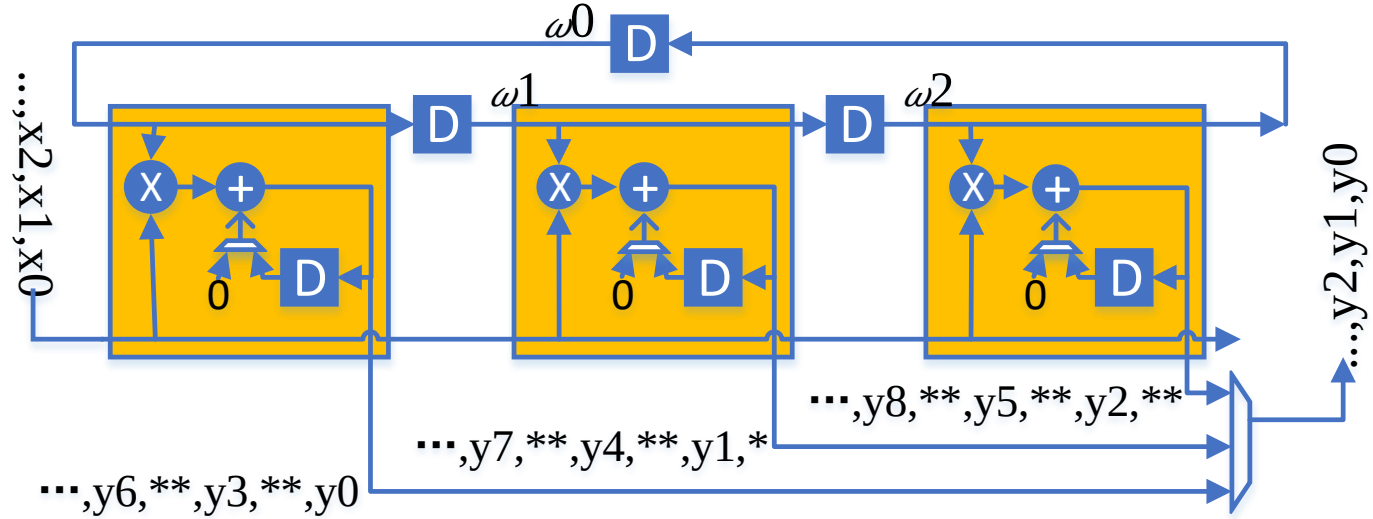
$$w = [0,1] \quad S \cdot \begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}1\\0\end{bmatrix} \cdot \begin{bmatrix}1\\1\end{bmatrix} = 0 \quad P \cdot \begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}1\\1\end{bmatrix} \cdot \begin{bmatrix}0\\1\end{bmatrix} = 1$$

03.脉动架构的应用——FIR systolic arrays-B_2

Procedure



$$y = [1, -1] \quad S \cdot \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ -1 \end{bmatrix} = 1 \quad P \cdot \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ -1 \end{bmatrix} = 0$$



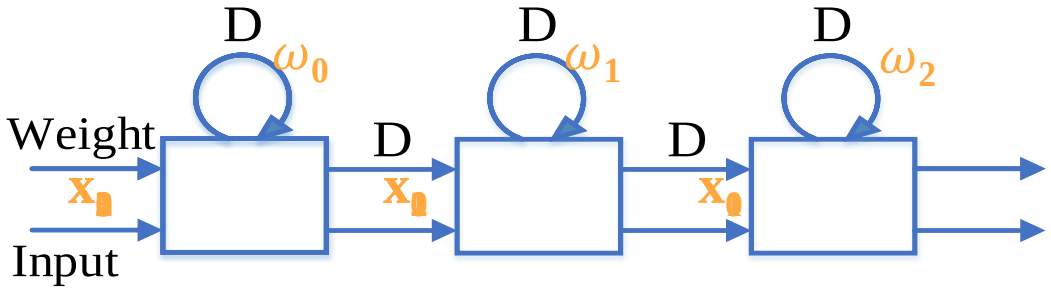
03.脉动架构的应用——FIR systolic arrays-F

Design F (Fan-In Results, Move Inputs, Weights Stay)

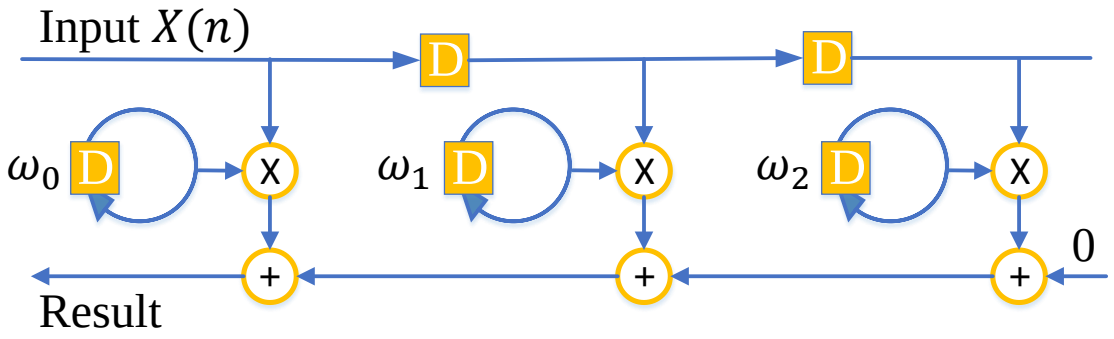
- Selecting $d^T = (1\ 0)$, $p^T = (0\ 1)$, $s^T = (1\ 1)$
- $HUE = 1/|s^T d| = 1.$

03.脉动架构的应用——FIR systolic arrays-F

Edge mapping:

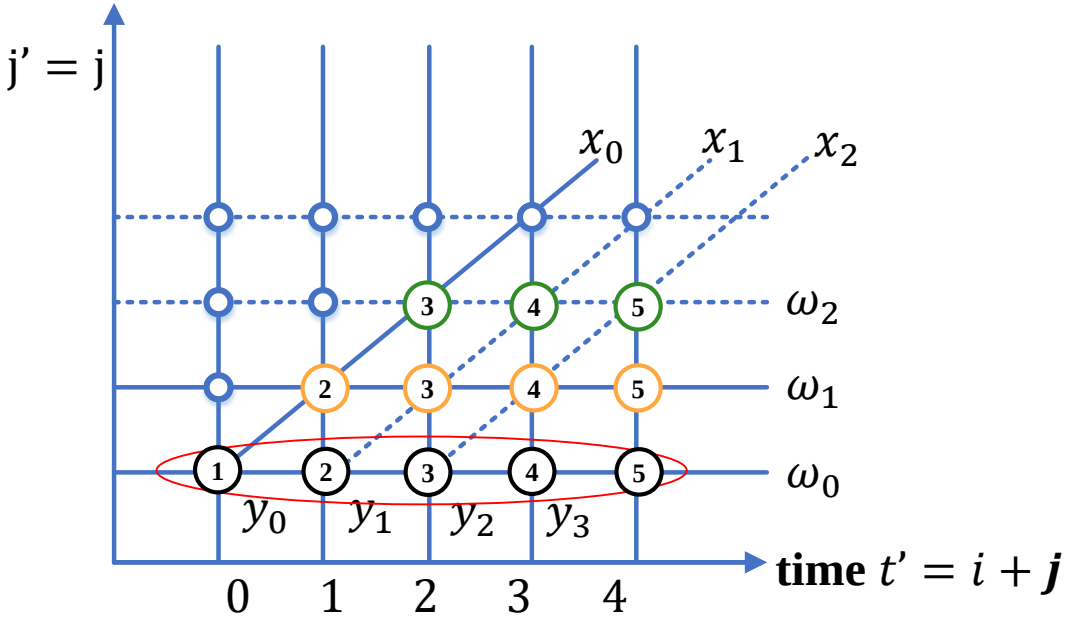


$\omega_0 x_0$
$\omega_0 x_1+\omega_1 x_0$ $\omega_1 x_0$
$\omega_0 x_2+\omega_1 x_1+\omega_2 x_0$ $\omega_1 x_1+\omega_2 x_0$ $\omega_2 x_0$
$\omega_0 x_3+\omega_1 x_2+\omega_2 x_1$ $\omega_1 x_2+\omega_2 x_1$ $\omega_2 x_1$

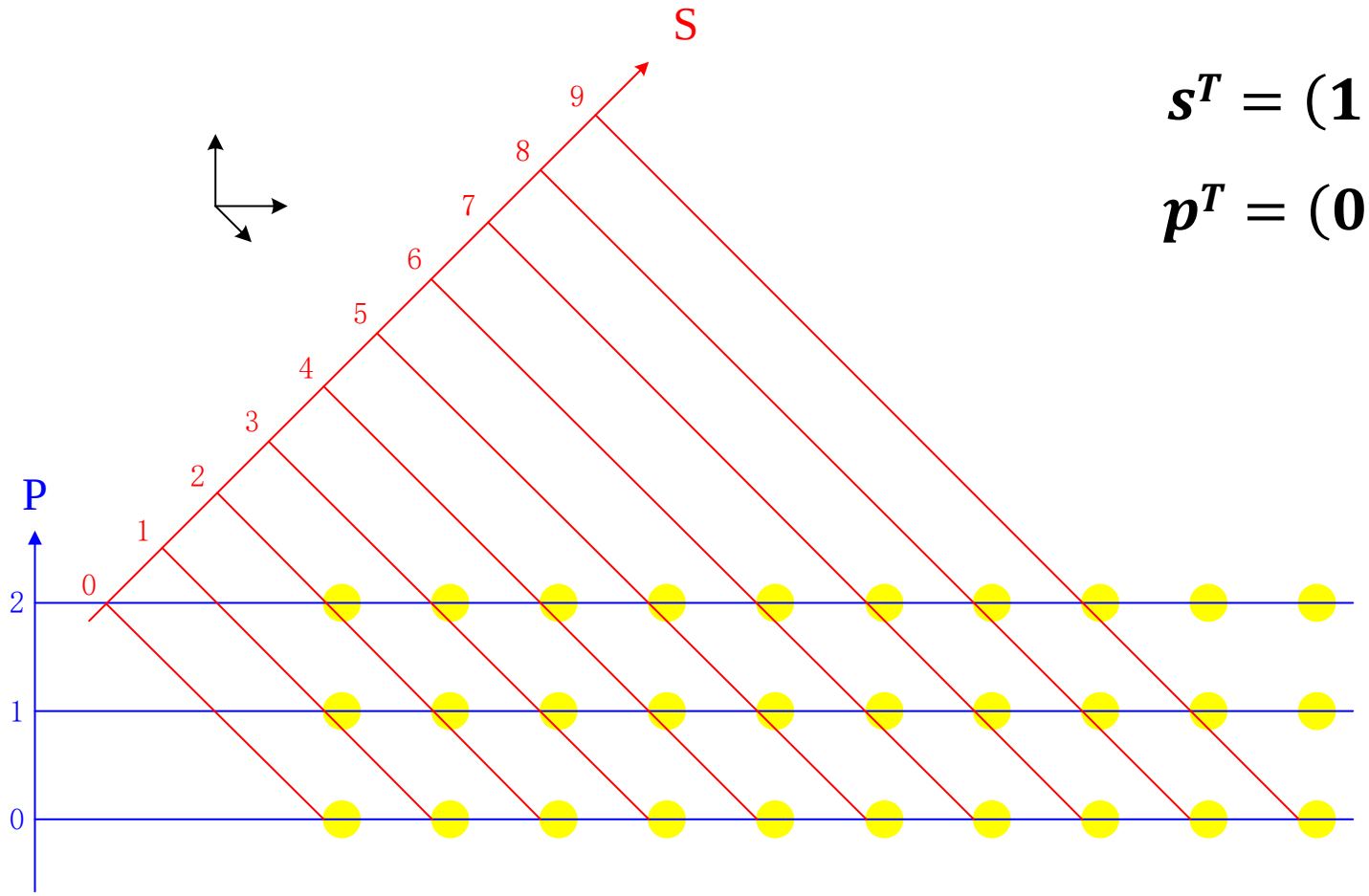


e	$p^T e$	$s^T e$
Weight $[1\ 0]^T$	0	1
i/p $[0\ 1]^T$	1	1
Result $[1\ -1]^T$	-1	0

Processor axis maintain, others rotate.



03.脉动架构的应用——FIR systolic arrays-F



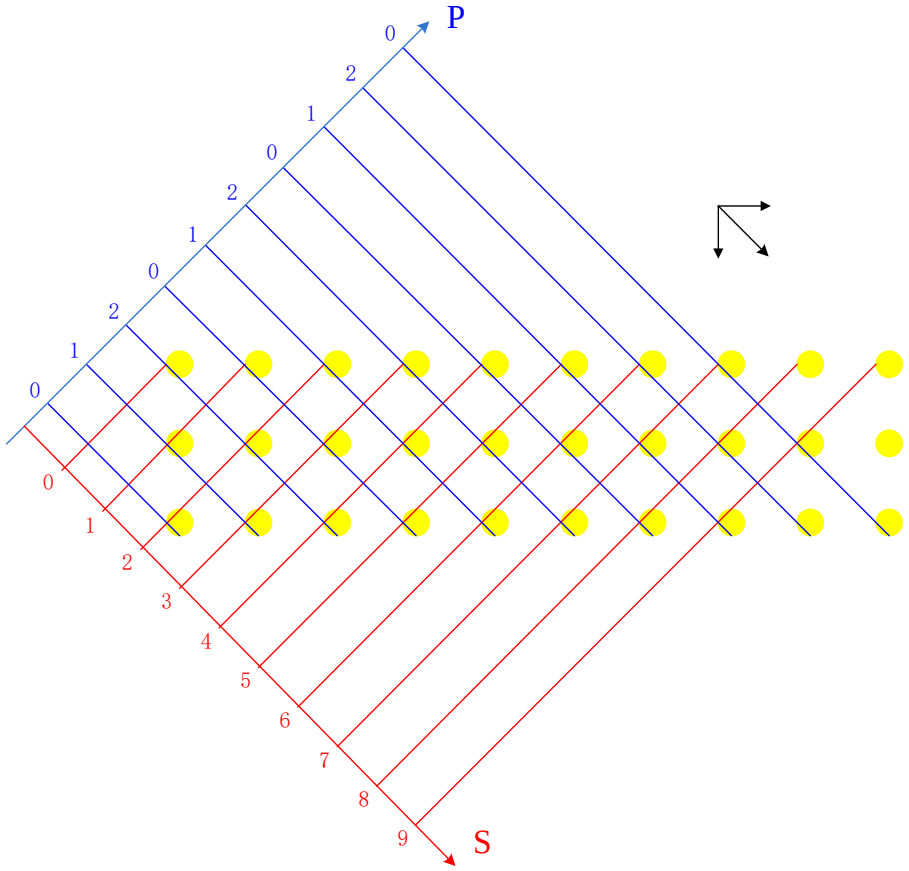
$$s^T = (1\ 1)$$

$$p^T = (0\ 1)$$

03.脉动架构的应用——FIR systolic arrays-R_1

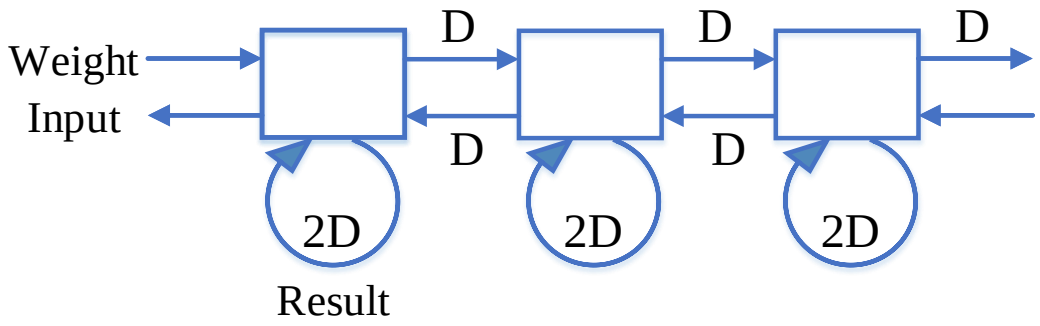
Design R_1 (Results Stay, Inputs and Weights Move in Opposite Direction)

- Selecting $d^T = (1 \quad -1)$, $p^T = (1\ 1)$, $s^T = (1 \quad -1)$
- Since $s^T d = 2$, we have $HUE = 1/|s^T d| = 1/2$.

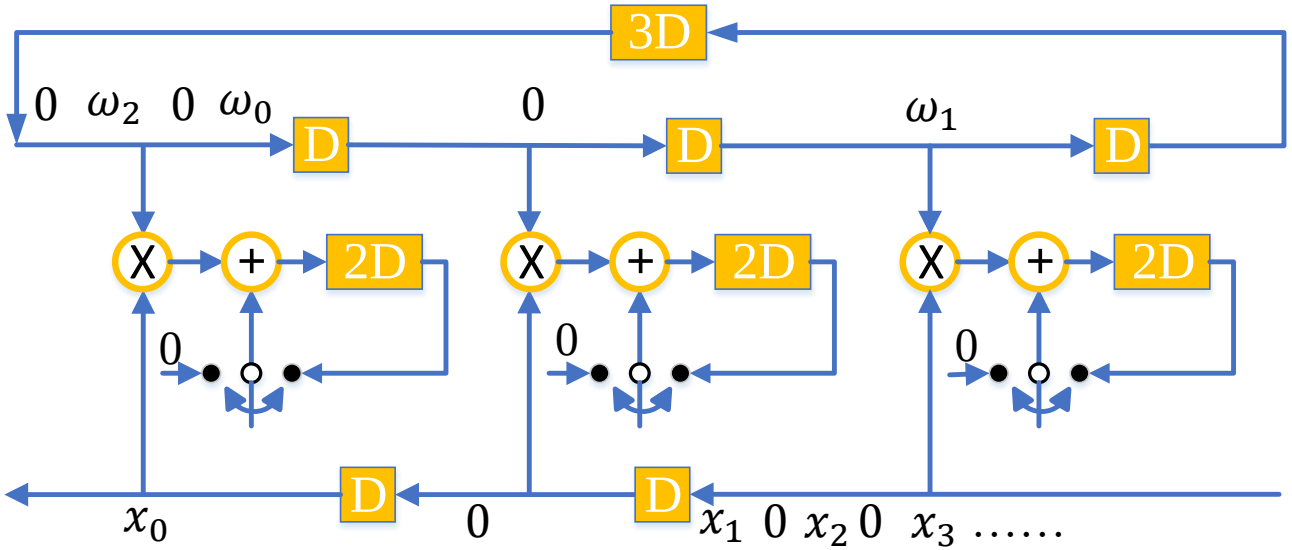


03.脉动架构的应用——FIR systolic arrays-R_1

Edge mapping:



e	$p^T e$	$s^T e$
Weight $[1\ 0]^T$	0	1
input $[0\ 1]^T$	1	-1
Result $[1\ -1]^T$	0	2

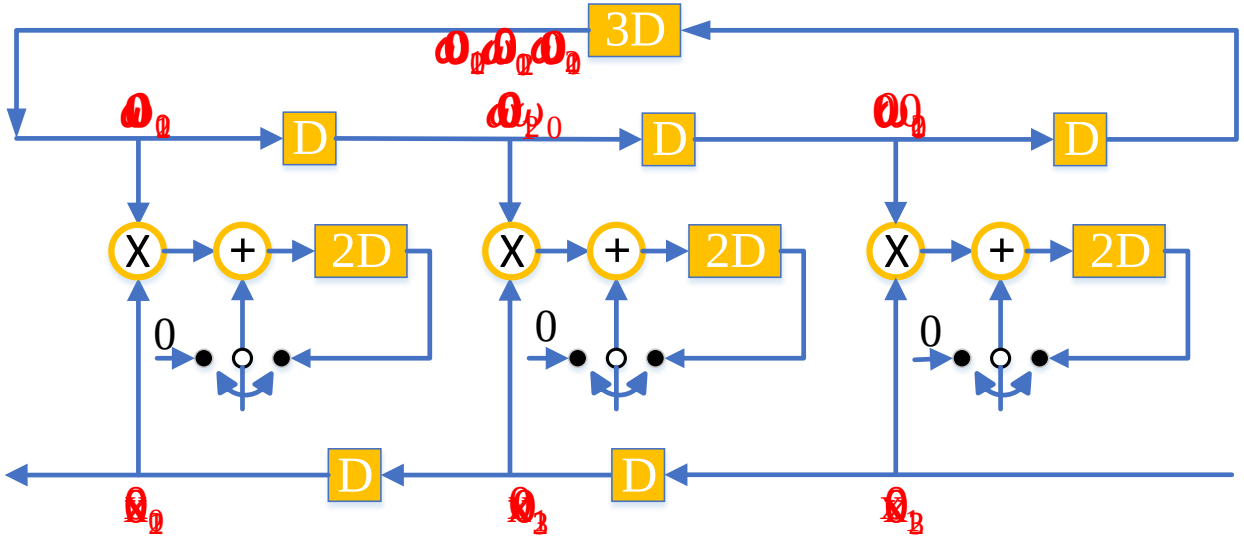


Note:

R_1 can be obtained from B_2 by 2-slow transformation and then retiming after changing the direction of signal x.

03.脉动架构的应用——FIR systolic arrays-R_1

Edge mapping:



$\omega_0 x_0$	0	$\omega_2 x_1$
0	$\omega_0 x_1$	0
$\omega_1 x_1$	0	$\omega_0 x_2$
0	$\omega_1 x_2$	0
$\omega_2 x_2$	0	$\omega_1 x_3$
0	$\omega_2 x_3$	0

04.本章总结

脉动架构在规则依赖图上，利用线性映射技术设计

线性映射技术

- 把N维DG映射到(N-1)维脉动架构的空间-时间表示
  - 每个节点映射到特定的PE
  - 在特定的时刻被调度
- 映射的可行性约束：对于给定问题选取d、P^T、 s^T的规则
  - $P^T d = 0$: P^T和d必须互相正交
  - $s^T d \neq 0$：映射e，在脉动阵到同一PE的节点A和B，不能同时执行
  - 边的映射：DG中的边列中相应于PE间的边沿($P^T e$)方向，延迟为$s^T d \geq 0$

    (延迟为0的变量为广播变量)

04.本章总结

线性映射技术

- 空间-时间表示：
  - $j' = p^T I$
  - $t' = s^T I$
- 步骤：
  - 选取d、P^T、s^T
  - 计算$P^T I$、 $s^T I$、边的映射表
  - 根据映射表画出脉动阵列架构和底层实现
- 脉动阵列可以与各种变换技术结合使用

阅读与思考

思考题

- 脉动阵列与并行相比，优缺点分别是什么?
- 了解Google公司的TPU中使用的脉动阵列

推荐阅读文献

In-Datacenter Performance Analysis of a Tensor Processing Unit

ABSTRACT

Many architects believe that major improvements in cost-energy-performance must now come from domain-specific hardware. This paper evaluates a custom ASIC—called a *Tensor Processing Unit (TPU)*— deployed in datacenters since 2015 that accelerates the inference phase of neural networks (NN). The heart of the TPU is a 65,536 8-bit MAC matrix multiply unit that offers a peak throughput of 92 TeraOps/second (TOPS) and a large (28 MiB) software-managed on-chip memory. The TPU's deterministic execution model is a better match to the 99th-percentile response-

KEYWORDS

DNN, MLP, CNN, RNN, LSTM, neural network, deep learning, domain-specific architecture, accelerator, TensorFlow, TPU, GPU

ACM Reference format:

Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, et al., Google, Inc., Mountain View, CA USA 2017. In-Datacenter Performance Analysis of a Tensor Processing Unit. In *Proceedings of ISCA '17, Toronto, ON, Canada, June 24-28, 2017*, 12 pages.
https://doi.org/10.1145/3079856.3080246

Jouppi et al. "In-datacenter performance analysis of a tensor processing unit. ISCA, 2017

课后作业

7.1 对于右图所示的依赖图

- 下面各组调度和投影中哪组是可行的?

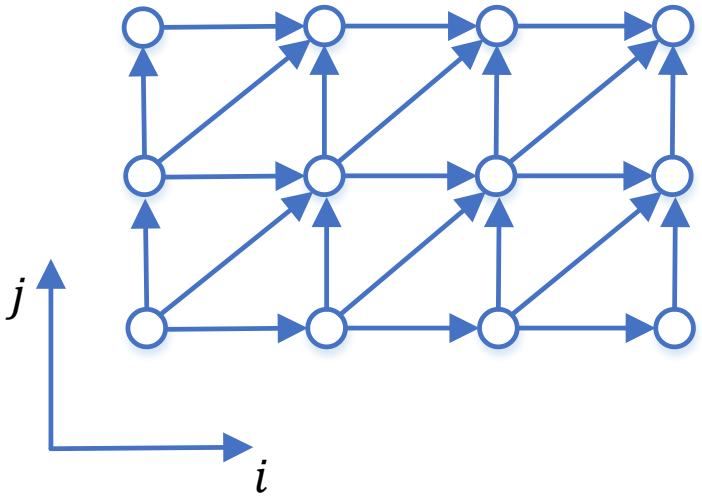
  a) $s^T = [1\ \ 0],\ d^T = [1\ \ 0]$

  b) $s^T = [1\ \ 2],\ d^T = [2\ \ -1]$

  c) $s^T = [1\ \ 1],\ d^T = [1\ \ 0]$

  d) $s^T = [1\ \ -2],\ d^T = [1\ \ 0]$

- 推导每个可行组的投影脉动阵列



谢谢！