Embedded Intelligent System and Novel Computer Architecture

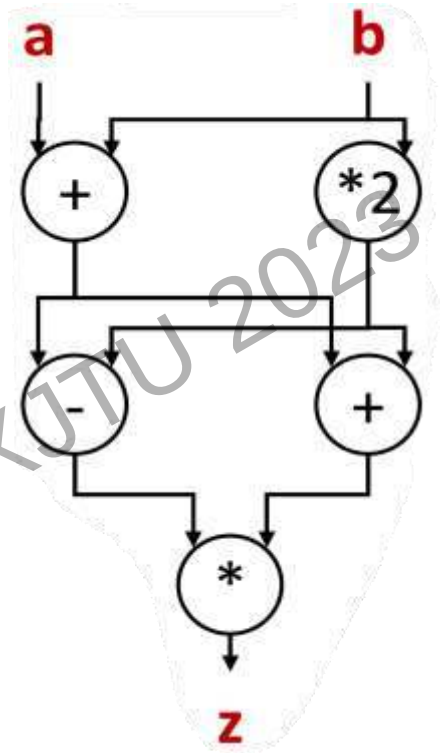
Lecture 05 – Systolic Array

Pengju Ren
Institute of Artificial Intelligence and Robotics
Xi'an Jiaotong University

http://gr.xjtu.edu.cn/web/pengjuren

Very Different Architectures Exist

```
v = a+b;
w= b*2;
x = v-w;
y = v+w;
z = x*y ;
```

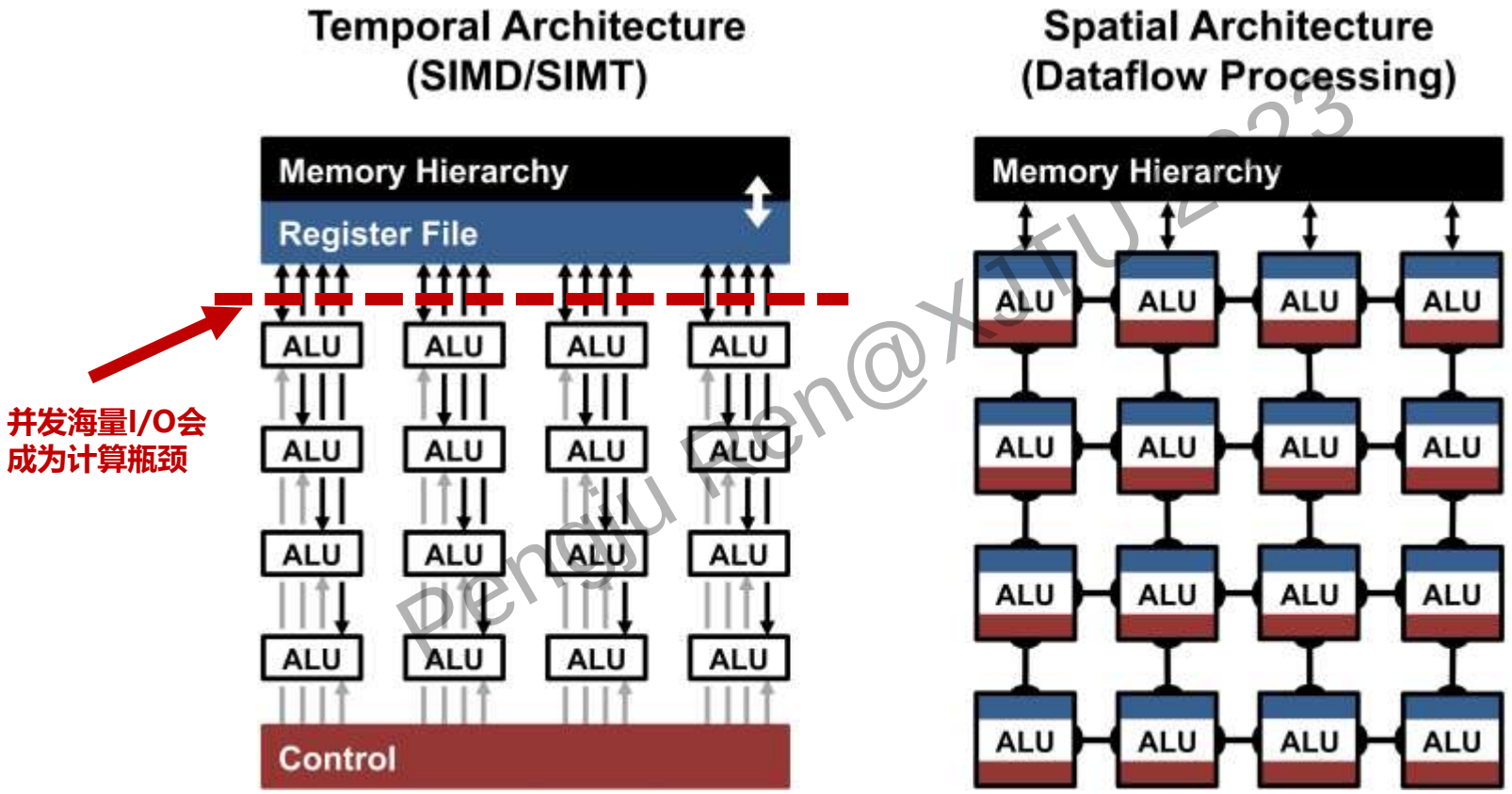


Consider a von Neumann program

– what is the significance of the instruction order?

– what is the significance of the storage locations?

Temporal v.s Spatial Architecture

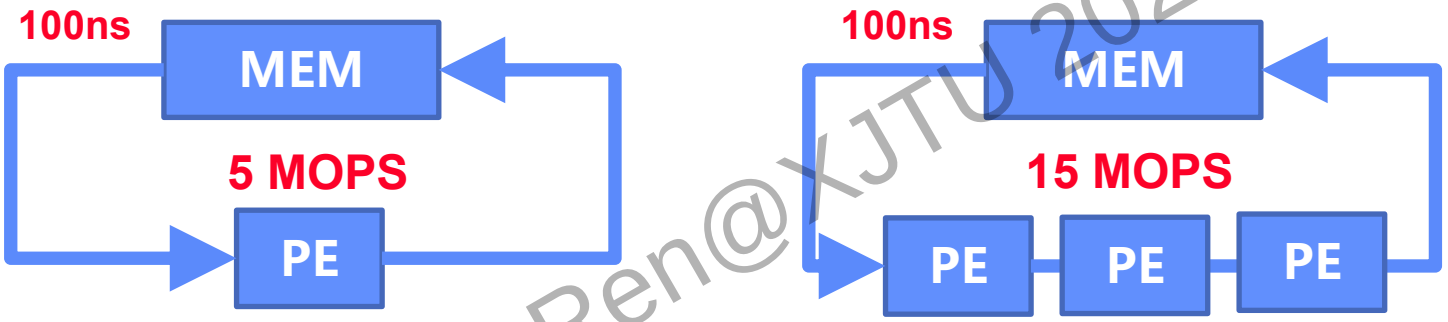


Systolic Array

$$A[i] = aA^2[i] + b$$

What is the performance (MOPS) of these two Architectures ?

Note: MOPS = Millions of Operations Per Second



Systolic Computers are a pipelined array architecture, by replace single processor with an array of regular processing elements(PEs), and orchesteate data flow for high throughput with less memory access.

- **Synchrony**
- **Modularity**
- **Regularity**
- **Spatial Locality**
- **Temporal Locality**
- **Pipelinability**
- **Parallel Computing**

Applications of Systolic Arrays

The various applications of systolic arrays are:

1. **Matrix Inversion and Decomposition.**
2. **Polynomial Evaluation.**
3. **Convolution.**
4. **Matrix multiplication.**
5. **Image Processing.**
6. **Systolic lattice filters used for speech and seismic signal processing.**
7. **Artificial neural network**

Case Studies:

- **DFT**
- **1D - CONV**
- **Pattern Matching**
- **Top-K elements of an input data-stream**
- **VM and MM Mul**

DFT (Discrete Fourier Transform)

For a given input array: $a = (a_0, a_1, a_2, \ldots, a_{n-1})$

y=DFT(a) = $(y_0, y_1, y_2, \ldots, y_{n-1})$ where:

$$y_j = \sum_{k=0}^{n-1} a_k w^{kj}, 0 \le j \le n-1$$

$$w = e^{2\pi i/n}, i = \sqrt{-1}$$

$$y(x) = \sum_{k=0}^{n-1} a_k x^k \,|_{x=w^j}$$

DFT (Discrete Fourier Transform)

$$y(x) = \sum_{k=0}^{n-1} a_k x^k \,|_{x=w^j}$$

⇩ A five elements DFT is:

$$y(x = w^j) = a_4(w^j)^4 + a_3(w^j)^3 + a_2(w^j)^2 + a_1(w^j)^1 + a_0$$

⇩

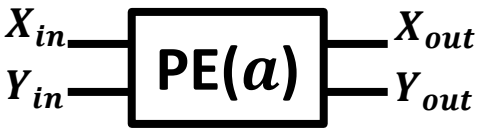
$$\begin{aligned}
y(x = w^0) &= a_4(w^0)^4 + a_3(w^0)^3 + a_2(w^0)^2 + a_1(w^0)^1 + a_0 \\
y(x = w^1) &= a_4(w^1)^4 + a_3(w^1)^3 + a_2(w^1)^2 + a_1(w^1)^1 + a_0 \\
y(x = w^2) &= a_4(w^2)^4 + a_3(w^2)^3 + a_2(w^2)^2 + a_1(w^2)^1 + a_0 \\
y(x = w^3) &= a_4(w^3)^4 + a_3(w^3)^3 + a_2(w^3)^2 + a_1(w^3)^1 + a_0 \\
y(x = w^4) &= a_4(w^4)^4 + a_3(w^4)^3 + a_2(w^4)^2 + a_1(w^4)^1 + a_0
\end{aligned}$$

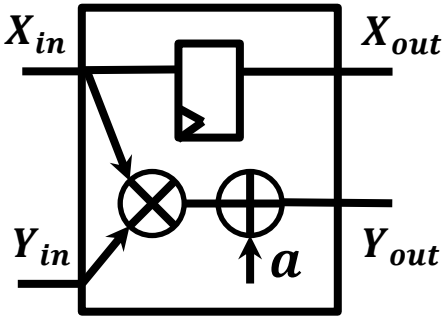
DFT (Discrete Fourier Transform)

$$
\begin{aligned}
y(x=w^0) &= a_4(w^0)^4 + a_3(w^0)^3 + a_2(w^0)^2 + a_1(w^0)^1 + a_0 \\
y(x=w^1) &= a_4(w^1)^4 + a_3(w^1)^3 + a_2(w^1)^2 + a_1(w^1)^1 + a_0 \\
y(x=w^2) &= a_4(w^2)^4 + a_3(w^2)^3 + a_2(w^2)^2 + a_1(w^2)^1 + a_0 \\
y(x=w^3) &= a_4(w^3)^4 + a_3(w^3)^3 + a_2(w^3)^2 + a_1(w^3)^1 + a_0 \\
y(x=w^4) &= a_4(w^4)^4 + a_3(w^4)^3 + a_2(w^4)^2 + a_1(w^4)^1 + a_0
\end{aligned}
$$

$$
\begin{aligned}
y(w^0) = y_0 &= \Big(\big((a_4(w^0)+a_3)w^0 + a_2\big)w^0 + a_1\Big)w^0 + a_0 \\
y(w^1) = y_1 &= \Big(\big((a_4(w^1)+a_3)w^1 + a_2\big)w^1 + a_1\Big)w^1 + a_0 \\
y(w^2) = y_2 &= \Big(\big((a_4(w^2)+a_3)w^2 + a_2\big)w^2 + a_1\Big)w^2 + a_0 \\
y(w^3) = y_3 &= \Big(\big((a_4(w^3)+a_3)w^3 + a_2\big)w^3 + a_1\Big)w^3 + a_0 \\
y(w^4) = y_4 &= \Big(\big((a_4(w^4)+a_3)w^4 + a_2\big)w^4 + a_1\Big)w^4 + a_0
\end{aligned}
$$

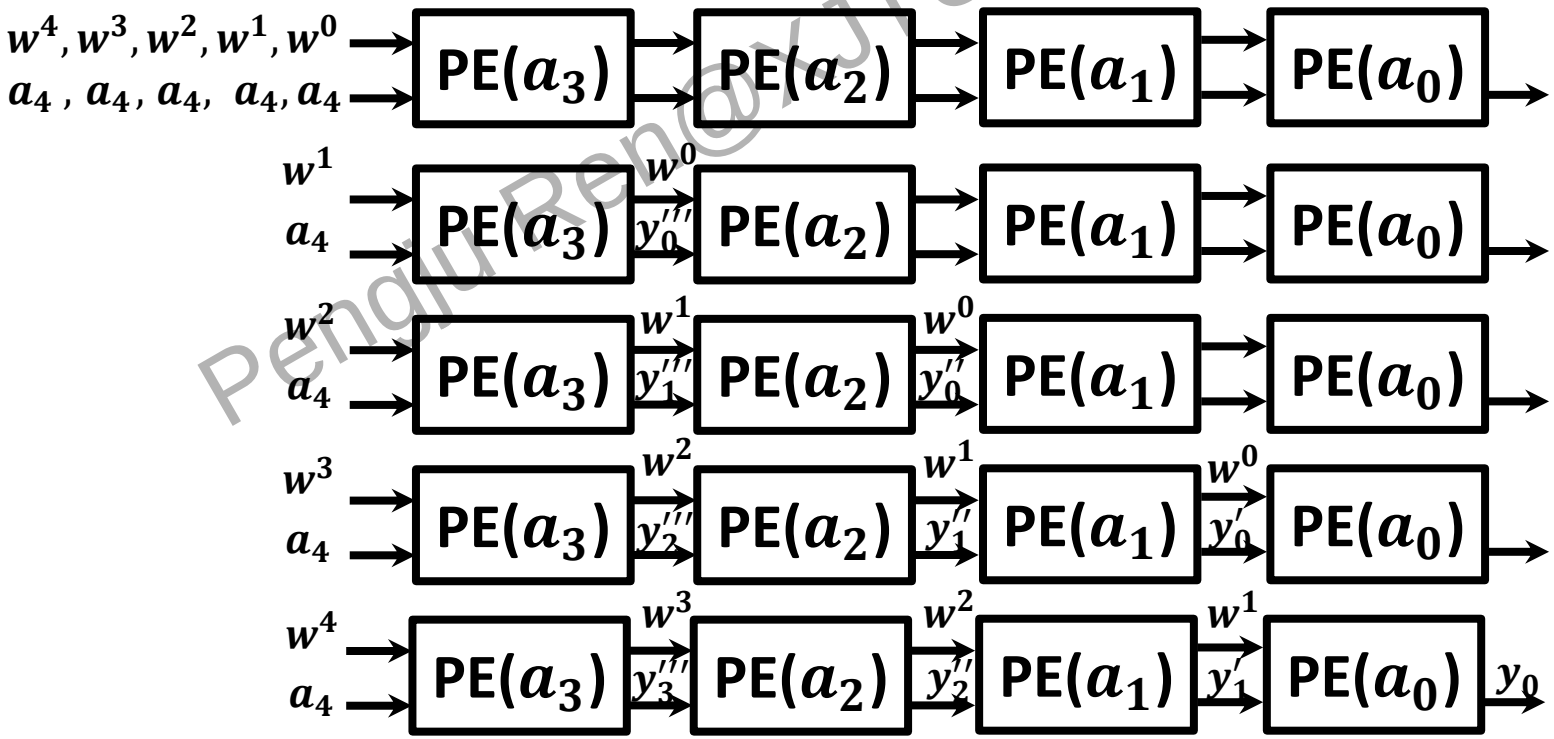
DFT Hardware implementation





$$X_{out} \leftarrow X_{in}$$

$$Y_{out} \leftarrow Y_{in}X_{in} + a$$



CONV (Convolution Operation)

Given a Weights Array: $w = (w_1, w_2, \dots, w_n)$

and the Inputs Array: $x = (x_1, x_2, \dots, x_m)$,

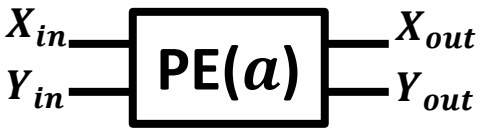
The Output Array: $y = (y_1, y_2, \dots, y_{m-n+1})$, Where y_j is as following:

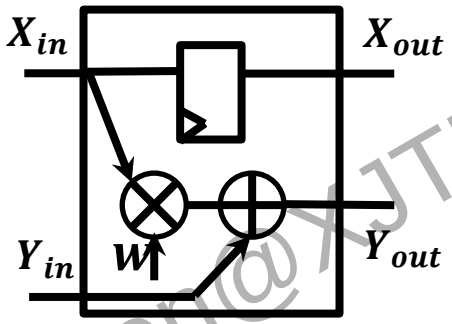
$$y_j = \sum_{i=1}^{n} w_i x_{i+j-1}, 1 \le j \le m - n - 1$$

If n=4, m=6 then: $y = (y_1, y_2, y_3)$, as following:

$$y_1 = x_1 w_1 + x_2 w_2 + x_3 w_3 + x_4 w_4$$
$$y_2 = x_2 w_1 + x_3 w_2 + x_4 w_3 + x_5 w_4$$
$$y_3 = x_3 w_1 + x_4 w_2 + x_5 w_3 + x_6 w_4$$

CONV Hardware Implementation

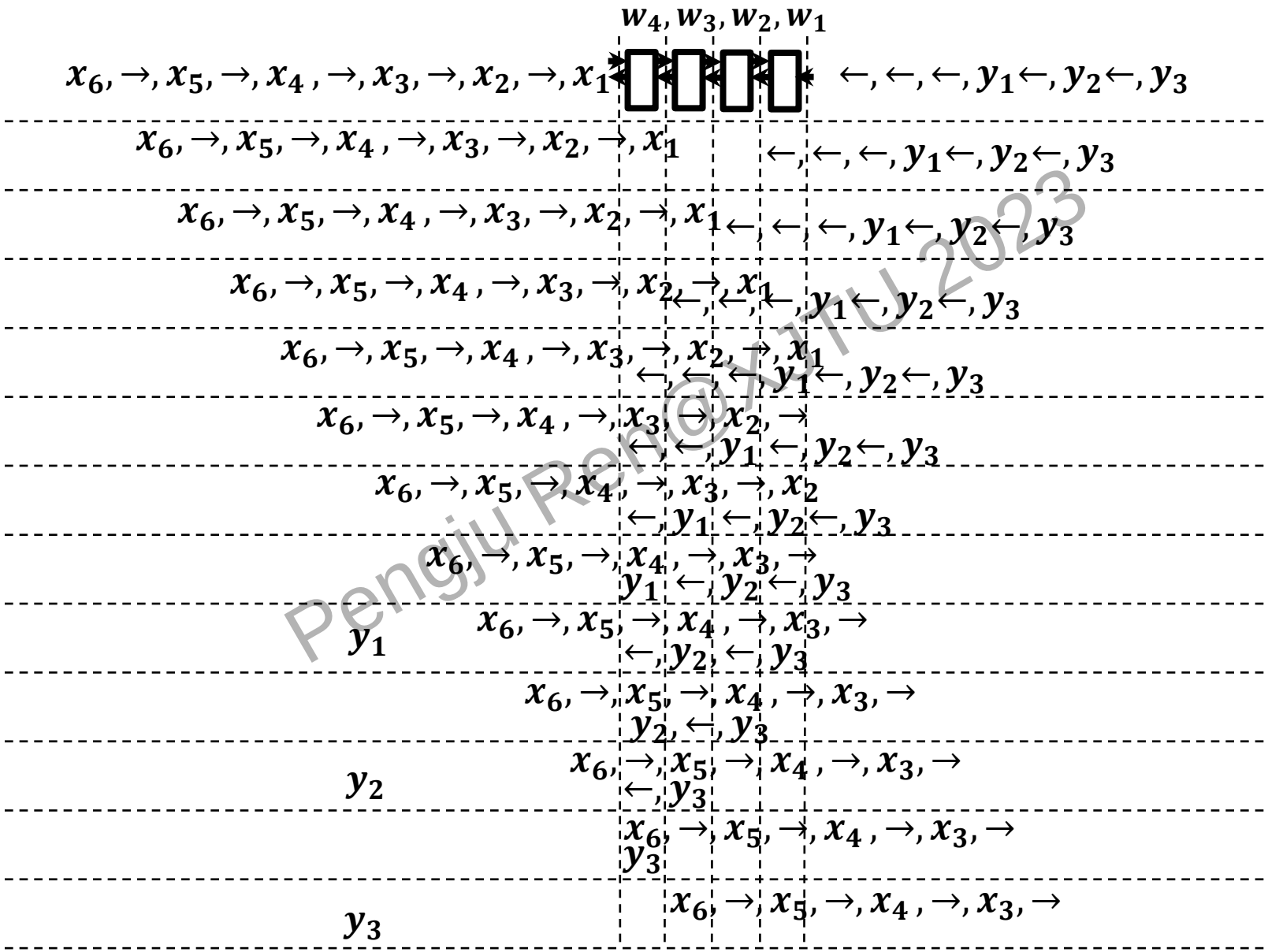




$$X_{out} \leftarrow X_{in}$$

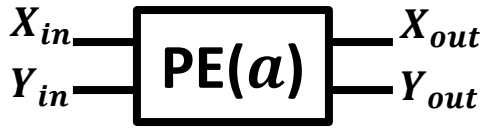
$$Y_{out} \leftarrow wX_{in} + Y_{in}$$

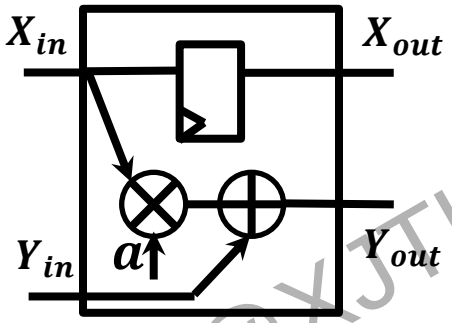
CONV Hardware Implementation



Combine CONV & DFT Hardware Implementation

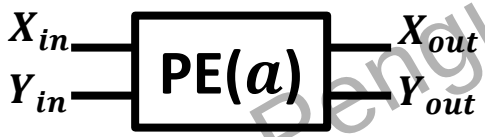
PE (CONV)

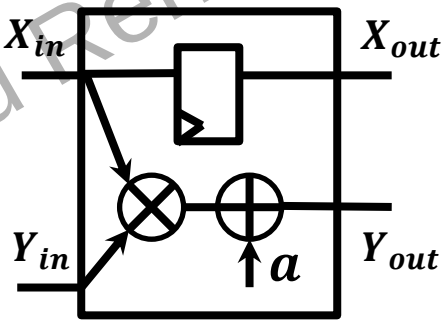




$$X_{out} \leftarrow X_{in}$$
$$Y_{out} \leftarrow aX_{in} + Y_{in}$$

PE (DFT)





$$X_{out} \leftarrow X_{in}$$
$$Y_{out} \leftarrow Y_{in}X_{in} + a$$

Pattern Matching can also using same Arch

CONV

Given a Weights Array: $w = (w_1, w_2, \dots, w_n)$

and the Inputs Array: $x = (x_1, x_2, \dots, x_m)$,

The Output Array: $y = (y_1, y_2, \dots, y_{m-n+1})$, Where y_j is as following:

$$y_j = \sum_{i=1}^{n} w_i x_{i+j-1}, \qquad 1 \le j \le m - n - 1$$

Pattern Matching

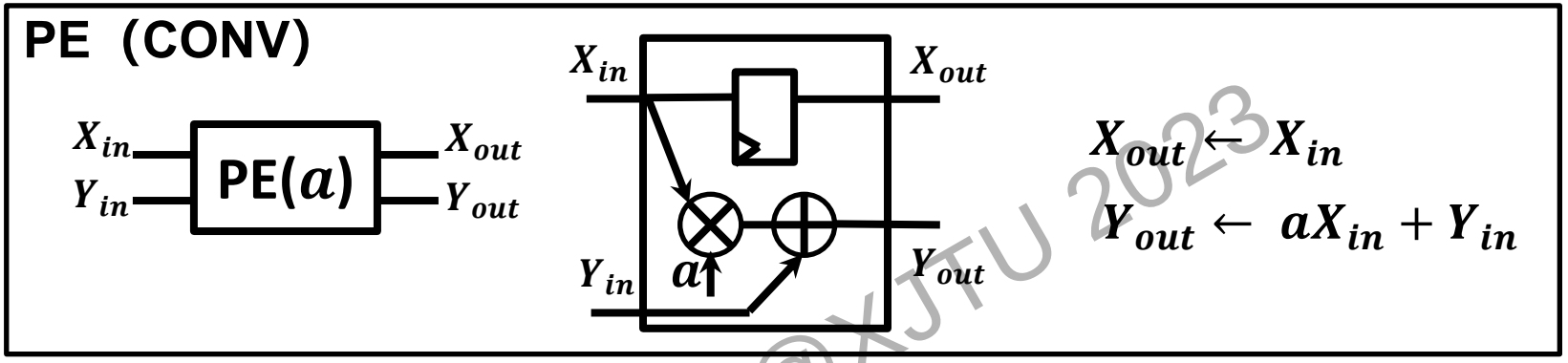
Given a Pattern : $w = (w_1, w_2, \dots, w_n)$

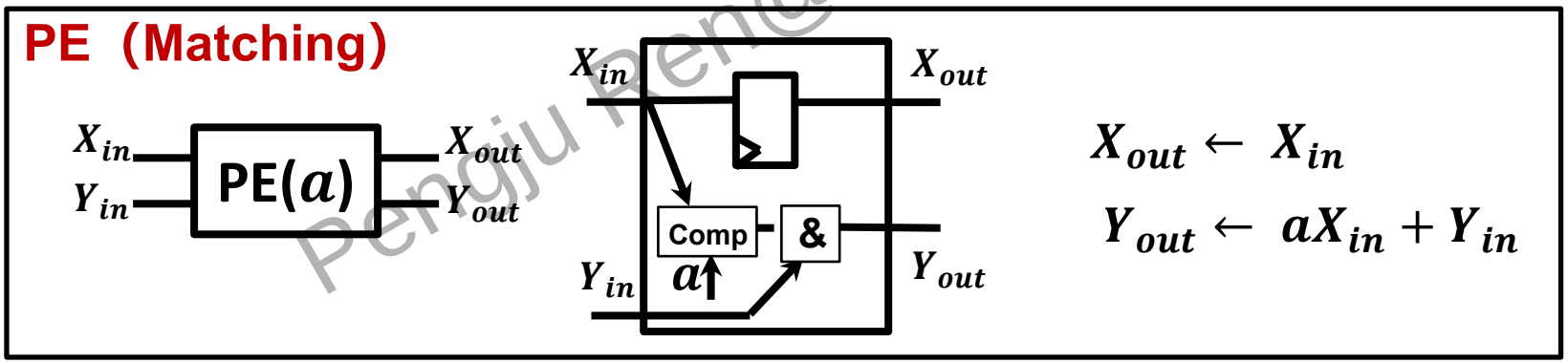
and the Inputs String: $x = (x_1, x_2, \dots, x_m)$,

The 1-bit Output Matching Result: $y = (y_1, y_2, \dots, y_{m-n+1})$, Where y_j is as following:

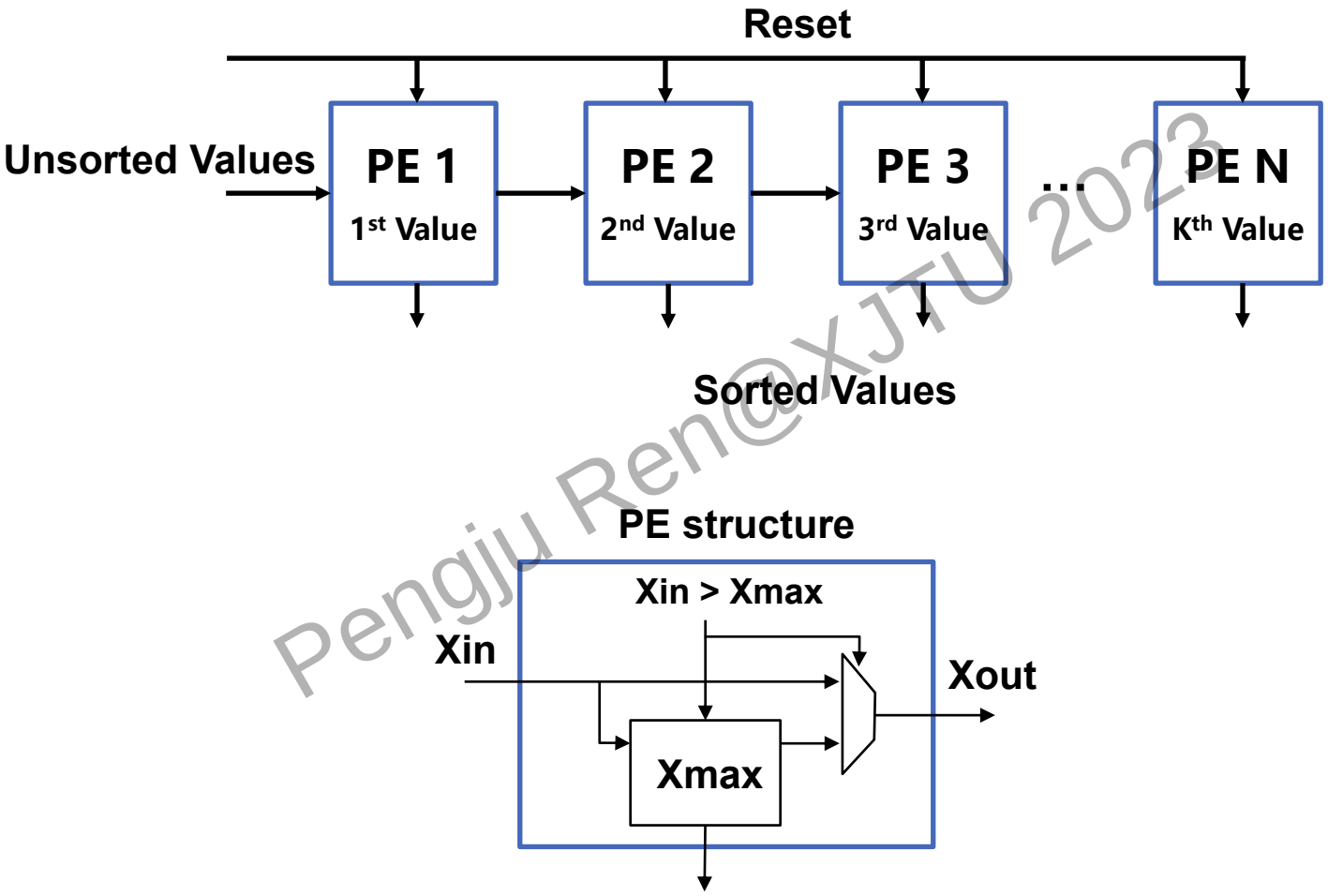
$$y_j = \prod_{i=1}^{n} Comp(w_i, x_{i+j-1}), \qquad 1 \le j \le m - n - 1$$

Pattern Matching can also using same Arch

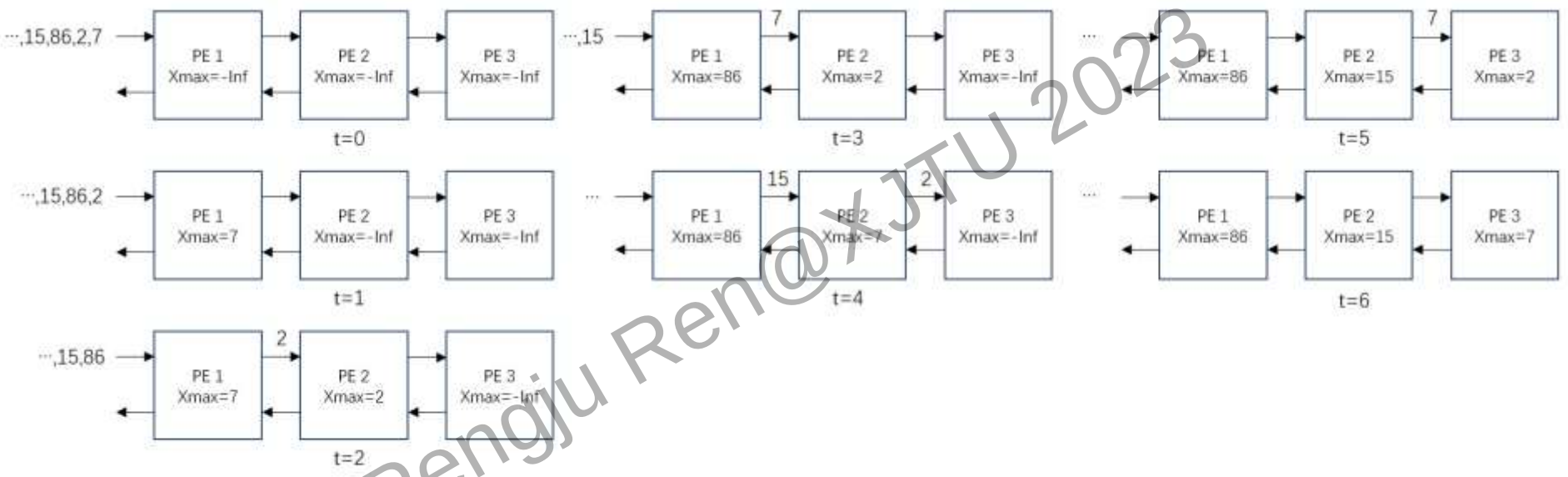




Top K elements of a input queue ?



Top K elements of a input queue ?

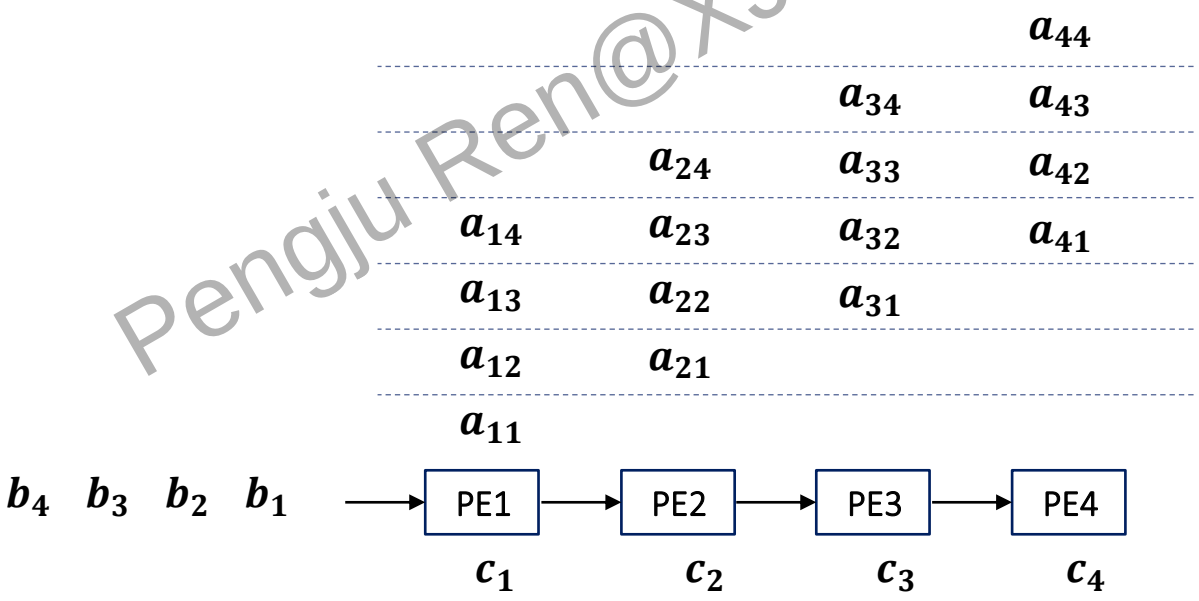


部分排序运行过程（N=3）

Other Examples : Vector Matrix Mul

C = A × B

$$\begin{bmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{41} \\ a_{21} & a_{22} & a_{23} & a_{42} \\ a_{31} & a_{32} & a_{33} & a_{43} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}$$



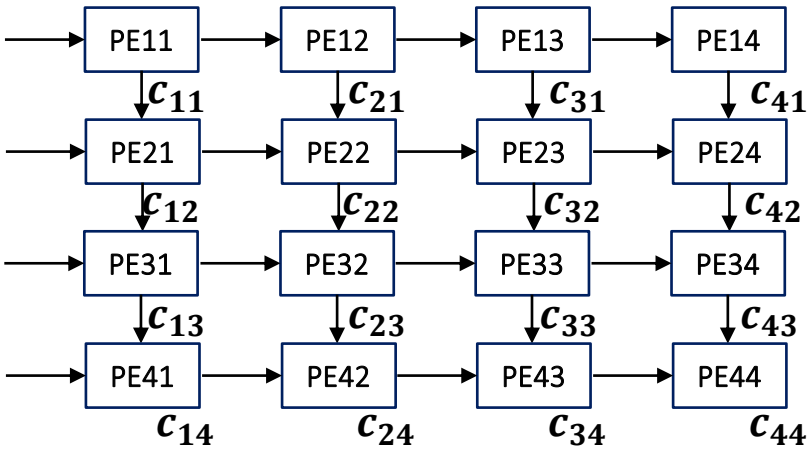
Other Examples : Matrix Matrix Mul

$$C = A \times B$$

$$\begin{bmatrix} c_{11} & \cdots & c_{14} \\ \vdots & \ddots & \vdots \\ c_{41} & \cdots & c_{44} \end{bmatrix} = \begin{bmatrix} a_{11} & \cdots & a_{14} \\ \vdots & \ddots & \vdots \\ a_{41} & \cdots & a_{44} \end{bmatrix} \begin{bmatrix} b_{11} & \cdots & b_{14} \\ \vdots & \ddots & \vdots \\ b_{41} & \cdots & b_{44} \end{bmatrix}$$

			a_{44}
		a_{34}	a_{43}
	a_{24}	a_{33}	a_{42}
a_{14}	a_{23}	a_{32}	a_{41}
a_{13}	a_{22}	a_{31}	
a_{12}	a_{21}		
a_{11}			

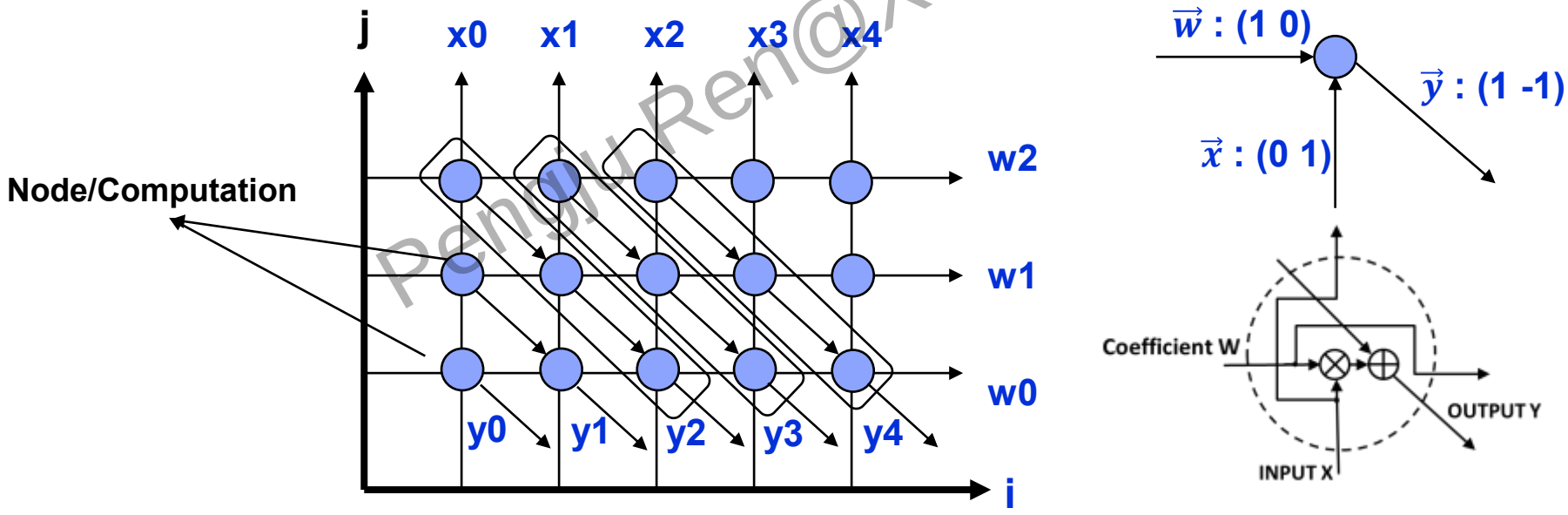
			b_{41}	b_{31}	b_{21}	b_{11}
		b_{42}	b_{32}	b_{22}	b_{12}	
	b_{43}	b_{33}	b_{23}	b_{13}		
b_{44}	b_{34}	b_{24}	b_{14}			



Systolic Array Design Methodology (1)

1. Represent the Algorithm as a Dependence Graph (DG)
2. Applying Processor, Scheduling and Projection Vectors (Space-Time Representation)
3. Edge Mapping
4. Construct the Final Systolic Architecture

$$FIR\ filter: y(n) = w_0 x(n) + w_1 x(n-1) + w_2 x(n-2)$$



Mapping from (i j) axis to (j' t), where j' is location(Processor) and t is time

Systolic Array Design Methodology (2)

1. Represent the Algorithm as a Dependence Graph (DG)
2. Applying Processor, Scheduling and Projection Vectors (Space-Time Representation)
3. Edge Mapping
4. Construct the Final Systolic Architecture

- **Processor Space Vector** : $\vec{p}^T = (p_1\ p_2)$

Any node with index $I^T = (i, j)$ would be executed by processor: $\vec{p}^T I$

- **Scheduling Vector**: $\vec{s}^T = (s_1\ s_2)$

Any node with index $I^T = (i, j)$ would be executed at time : $\vec{s}^T I$

- **Projection Vector** : $\vec{d}^T = (d_1\ d_2)$

Two nodes that are displaced by $\vec{d}$ or multiples of $\vec{d}$ are executed by the same processor

e.g, $\vec{d}^T = (1\ 0)$ → node (1 0) and (2 0) will map to the same PE

- **Hardware Utilization Efficiency:** HUE = $1/|\vec{s}^T d|$.

This is because two tasks executed by the same processor are spaced $|\vec{s}^T d|$ time units apart

Systolic Array Design Methodology (3)

1. Represent the Algorithm as a Dependence Graph (DG)
2. Applying Processor, Scheduling and Projection Vectors (Space-Time Representation)
3. Edge Mapping
4. Construct the Final Systolic Architecture

- Processor Space Vector and Projection Vector must be orthogonal to each other and $\vec{p}^T\vec{d} = 0$

Proof: I_A and I_B are two computational nodes in the DG

Therefor processor for $I_A(\vec{p}^T I_A)$ and $I_B(\vec{p}^T I_B)$ are same, if $I_A - I_B = \vec{d}$ -> $\vec{p}^T\vec{d} = 0$

- If A(I_A) and B(I_B)are mapped to the same processor, then they cannot be executed at the same time, i.e.,$\vec{s}^T I_A \neq \vec{s}^T I_B$, that is, $\vec{s}^T d \neq 0$.

- Edge mapping: If an edge $\vec{e}$ exists in the space representation or DG, then an edge $\vec{p}^T\vec{e}$ is introduced in the systolic array with $\vec{s}^T\vec{e}$ delays.

- Hardware utilization: $\vec{s}^T d$= time different between successive nodes (computation) on same processor

e.g,. If $\vec{s}^T d$=1 -> new computation on array nodes with 100% HUE

$\vec{s}^T d$=2 -> .. 50% HUE

FIR Filter Design B1(Broadcast Inputs, Move Results, Weights Stay)

$$FIR\ filter:\ y(n) = w_0 x(n) + w_1 x(n-1) + w_2 x(n-2)$$

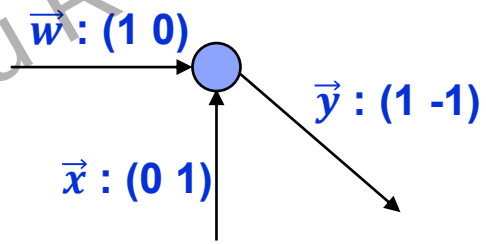
$$\vec{d}^T = (1\ 0),\ \vec{p}^T = (0\ \ 1),\ \vec{s}^T = (1\ \ 0)$$

Any node in the DG with index $I^T = (i, j)$:

- Is mapped to processor $\vec{p}^T I = j$
- Is executed at time $\vec{s}^T I = i$

Since $\vec{s}^T d = 1$ we have HUE = $1/|\vec{s}^T d|$ = 100%.

Edge mapping: The 3 fundamental edges corresponding to *weight*, *input* and *result* can be mapped to corresponding edges in the systolic array as per the following table:



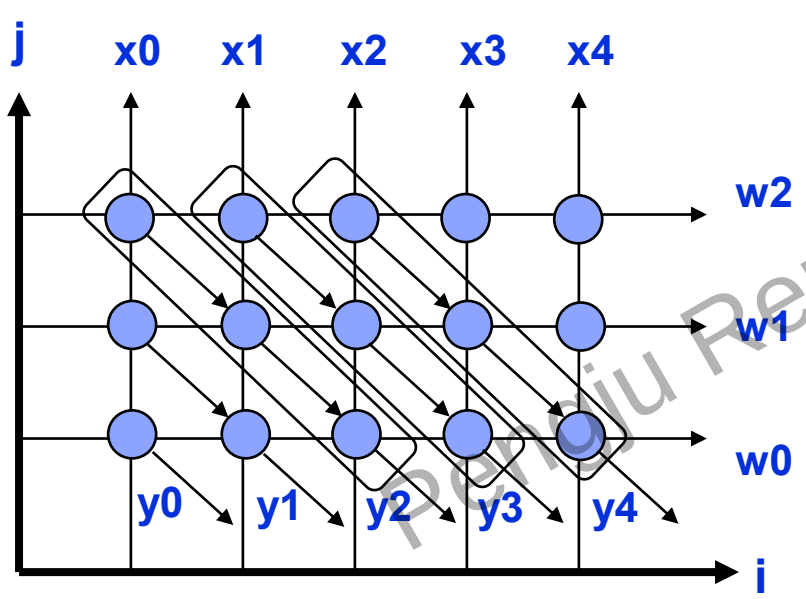
$\vec{e}$	$\vec{p}^T\vec{e}$	$\vec{s}^T\vec{e}$
weight $\vec{w}$=(1 0)	0	1
Inputs $\vec{x}$=(0 1)	1	0
Result $\vec{y}$=(1 -1)	-1	1

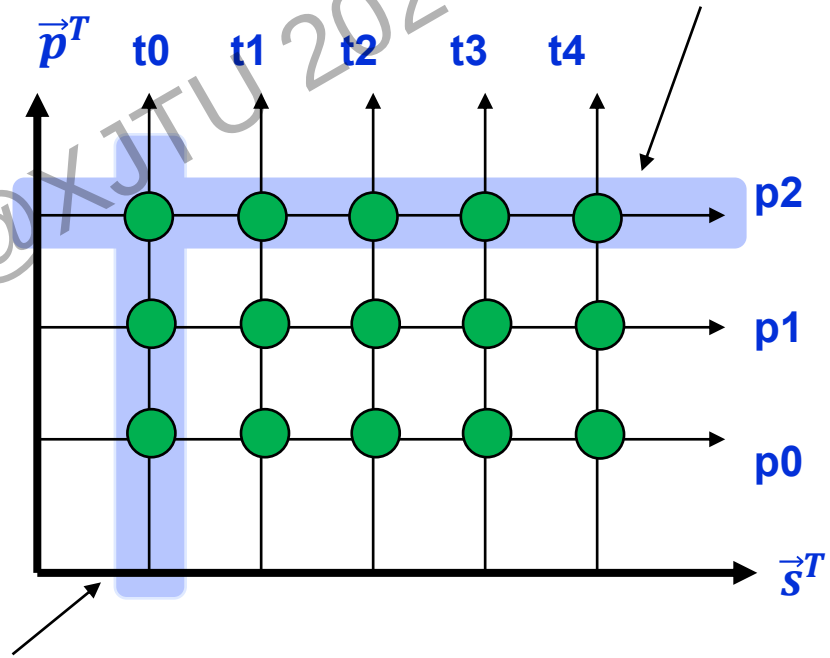
FIR Filter Design B1(Broadcast Inputs, Move Results, Weights Stay)

$$FIR\ filter:\ y(n) = w_0 x(n) + w_1 x(n-1) + w_2 x(n-2)$$

All the horizontal notes execute at the same processor

$(\vec{p}^T I = j)$





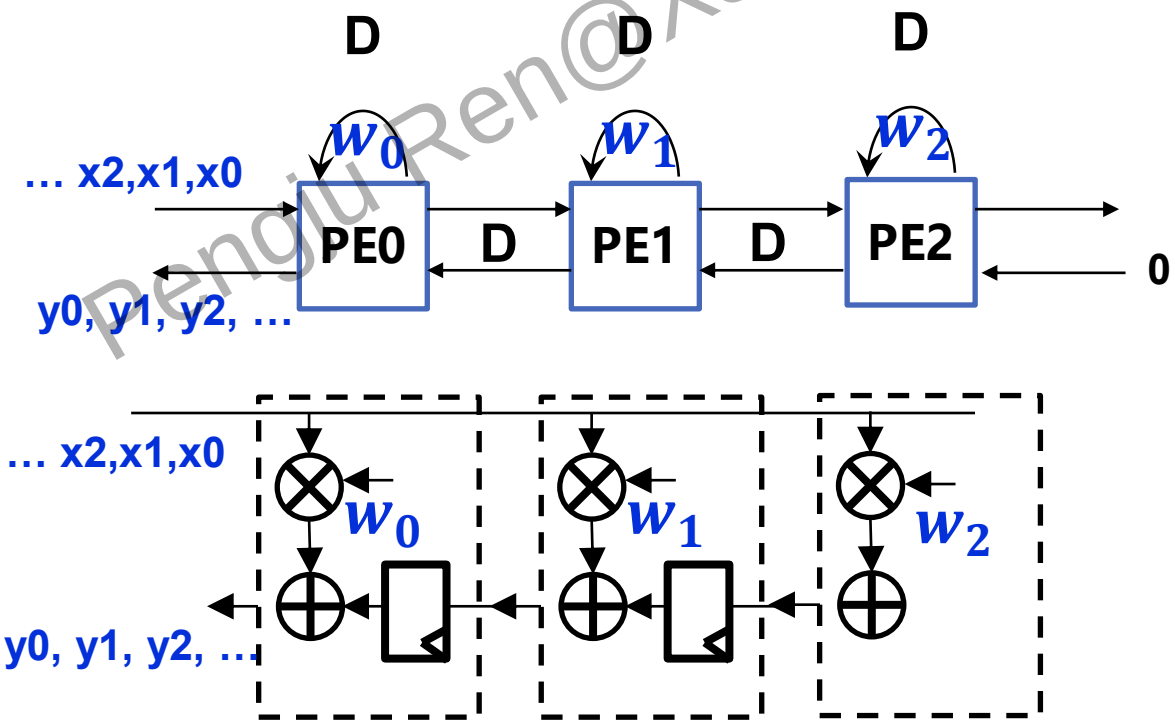
All the vertical notes execute at the same time

$(\vec{s}^T I = i)$

FIR Filter Design B1(Broadcast Inputs, Move Results, Weights Stay)

$FIR\ filter$: $y(n) = w_0 x(n) + w_1 x(n-1) + w_2 x(n-2)$

$\vec{e}$	$\vec{p}^T\vec{e}$	$\vec{s}^T\vec{e}$
weight $\vec{w}$=(1 0)	0	1
Inputs $\vec{x}$=(0 1)	1	0
Result $\vec{y}$=(1 -1)	-1	1



Alternative Designs

- B1 (Broadcast inputs, Move results, Weight Stay)
- B2 (Broadcast inputs, Move Weight, Results stay)
- F (Fan-in results, Move inputs, Weight stay)
- R1 (Results stay, Inputs and Weight move in opposite directions)
- R2 and Dual R2 (Results stay, Inputs and Weights move in the same direction but at different speeds)
- W1 (Weights stay, Inputs and Results move in opposite directions)
- W2 and Dual W2 (Weights stay, Inputs and Results move in same direction but at different speeds)

FIR Filter Design B2(Broadcast Inputs, Move Weights, Results Stay)

$$FIR\ filter:\ y(n) = w_0 x(n) + w_1 x(n-1) + w_2 x(n-2)$$

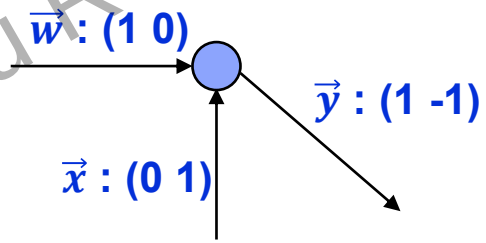
$$\vec{d}^T = (1\ -1), \vec{p}^T = (1\ \ 1), \vec{s}^T = (1\ \ 0)$$

Any node in the DG with index $I^T = (i, j)$:

- **Is mapped to processor $\vec{p}^T I_i = i + j$**
- **Is executed at time $\vec{s}^T I_i = i$**

Since $\vec{s}^T d$=1 we have HUE = $1/|\vec{s}^T d|$ = 100%.

Edge mapping: The 3 fundamental edges corresponding to *weight, input* and *result* can be mapped to corresponding edges in the systolic array as per the following table:

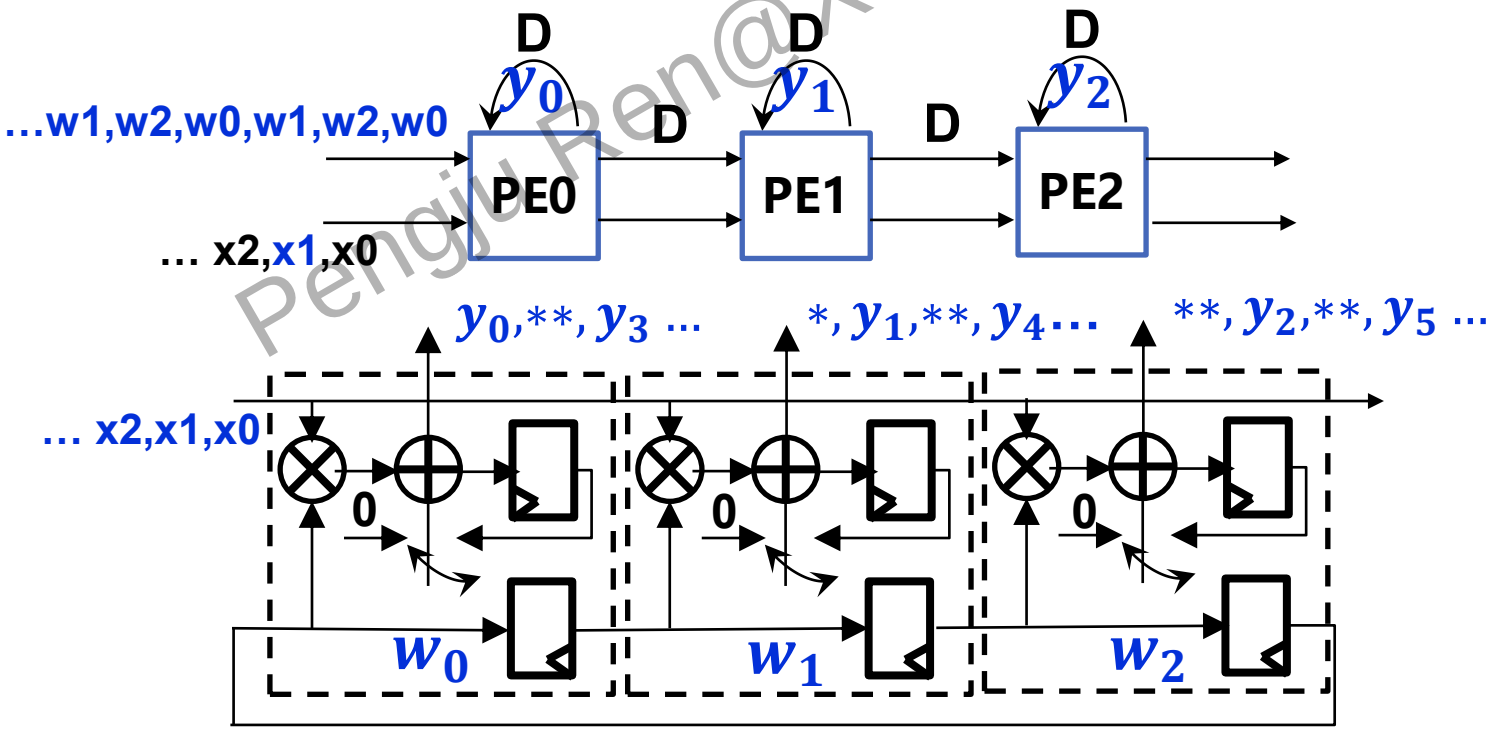


$\vec{e}$	$\vec{p}^T\vec{e}$	$\vec{s}^T\vec{e}$
weight $\vec{w}$=(1 0)	1	1
Inputs $\vec{x}$=(0 1)	1	0
Result $\vec{y}$=(1 -1)	0	1

FIR Filter Design B2(Broadcast Inputs, Move Weights, Results Stay)

$FIR\ filter$: $y(n) = w_0x(n) + w_1x(n-1) + w_2x(n-2)$

$\vec{e}$	$\vec{p}^T\vec{e}$	$\vec{s}^T\vec{e}$
weight $\vec{w}$=(1 0)	1	1
Inputs $\vec{x}$=(0 1)	1	0
Result $\vec{y}$=(1 -1)	0	1



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

$$FIR\ filter:\ y(n) = w_0 x(n) + w_1 x(n-1) + w_2 x(n-2)$$

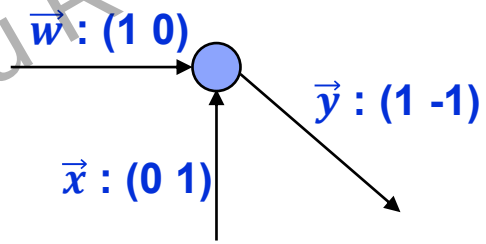
$$\vec{d}^T = (1\ 0),\ \vec{p}^T = (0\ \ 1), \vec{s}^T = (1\ \ 1)$$

Any node in the DG with index $I^T = (i, j)$:

- Is mapped to processor $\vec{p}^T I_i = j$
- Is executed at time $\vec{s}^T I_i = i$+j

Since $\vec{s}^T d$=1 we have HUE = $1/|\vec{s}^T d|$ = 100%.

Edge mapping: The 3 fundamental edges corresponding to *weight, input* and *result* can be mapped to corresponding edges in the systolic array as per the following table:

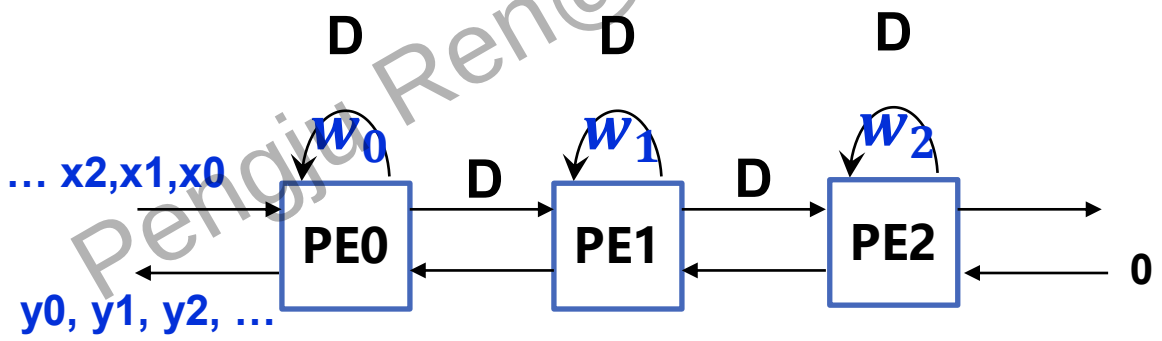


$\vec{e}$	$\vec{p}^T\vec{e}$	$\vec{s}^T\vec{e}$
weight $\vec{w}$=(1 0)	0	1
Inputs $\vec{x}$=(0 1)	1	1
Result $\vec{y}$=(1 -1)	-1	0

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

$FIR\ filter$: $y(n) = w_0x(n) + w_1x(n-1) + w_2x(n-2)$

$\vec{e}$	$\vec{p}^T\vec{e}$	$\vec{s}^T\vec{e}$
weight $\vec{w}$=(1 0)	0	1
Inputs $\vec{x}$=(0 1)	1	1
Result $\vec{y}$=(1 -1)	-1	0



FIR Filter Design R1(Results Stay, Inputs and Weights Move in Opposite Direction)

$$FIR\ filter:\ y(n) = w_0 x(n) + w_1 x(n-1) + w_2 x(n-2)$$

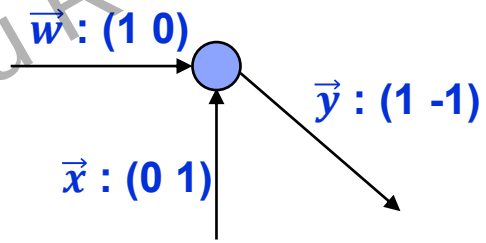
$$\vec{d}^T = (1\ -1),\ \vec{p}^T = (1\ \ 1), \vec{s}^T = (1\ \ -1)$$

Any node in the DG with index $I^T = (i, j)$:

- Is mapped to processor $\vec{p}^T I_i = i + j$
- Is executed at time $\vec{s}^T I_i = i$ –j

Since $\vec{s}^T d$=1 we have HUE = $1/|\vec{s}^T d|$ = ½=50%.

Edge mapping: The 3 fundamental edges corresponding to *weight, input* and *result* can be mapped to corresponding edges in the systolic array as per the following table:

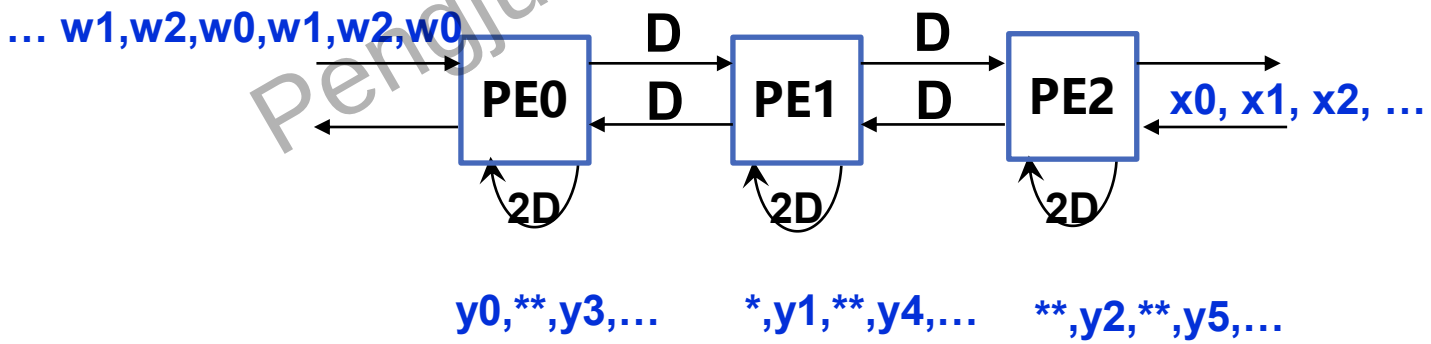


$\vec{e}$	$\vec{p}^T\vec{e}$	$\vec{s}^T\vec{e}$
weight $\vec{w}$=(1 0)	1	1
Inputs $\vec{x}$=(0 1)	-1	1
Result $\vec{y}$=(1 -1)	0	2

FIR Filter Design R1(Results Stay, Inputs and Weights Move in Opposite Direction)

$$FIR\ filter: y(n) = w_0 x(n) + w_1 x(n-1) + w_2 x(n-2)$$

$\vec{e}$	$\vec{p}^T\vec{e}$	$\vec{s}^T\vec{e}$
weight $\vec{w}$=(1 0)	1	1
Inputs $\vec{x}$=(0 1)	-1	1
Result $\vec{y}$=(1 -1)	0	2



FIR Filter Design R2 and Dual R2 (Results Stay, Inputs and Weights Move in Same Direction but at Different Speeds)

$$FIR\ filter:\ y(n) = w_0 x(n) + w_1 x(n-1) + w_2 x(n-2)$$

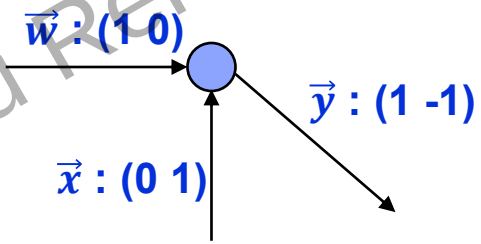
$$\vec{d}^T = (1\ \ -1),\ \vec{p}^T = (1\ \ 1), R2:\ \vec{s}^T = (2\ \ 1);\ Dual\ R2: \vec{s}^T = (1\ \ 2)$$

Any node in the DG with index $I^T = (i, j)$:

- Is mapped to processor $\vec{p}^T I_i = i + j$
- Is executed at time $R2:\ \vec{s}^T I_i = 2i$ +j; $Dual\ R2:\ \vec{s}^T I_i = i$ +2j

Since $\vec{s}^T d$=1 we have HUE = $1/|\vec{s}^T d|$ = 100%.

Edge mapping: The 3 fundamental edges corresponding to *weight, input* and *result* can be mapped to corresponding edges in the systolic array as per the following table:



	R2		Dual R2	
$\vec{e}$	$\vec{p}^T\vec{e}$	$\vec{s}^T\vec{e}$	$\vec{p}^T\vec{e}$	$\vec{s}^T\vec{e}$
weight $\vec{w}$=(1 0)	1	2	1	1
Inputs $\vec{x}$=(0 1)	1	1	1	2
Result $\vec{y}$=(1 -1)	0	1	0	1

FIR Filter Design W1(Weights Stay, Inputs and Results Move in Opposite Direction)

$$FIR\ filter:\ y(n) = w_0 x(n) + w_1 x(n-1) + w_2 x(n-2)$$

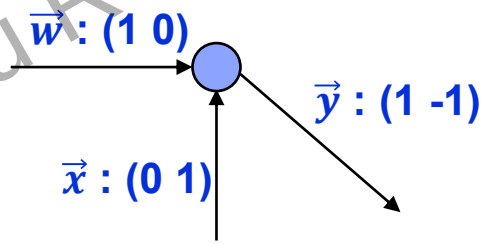
$$\vec{d}^T = (1\ 0),\ \vec{p}^T = (0\ \ 1), \vec{s}^T = (2\ \ 1)$$

Any node in the DG with index $I^T = (i, j)$:

- Is mapped to processor $\vec{p}^T I_i = j$
- Is executed at time $\vec{s}^T I_i = 2i$ +j

Since $\vec{s}^T d$=1 we have HUE = $1/|\vec{s}^T d|$ = ½=50%.

Edge mapping: The 3 fundamental edges corresponding to *weight, input* and *result* can be mapped to corresponding edges in the systolic array as per the following table:



$\vec{e}$	$\vec{p}^T\vec{e}$	$\vec{s}^T\vec{e}$
weight $\vec{w}$=(1 0)	0	2
Inputs $\vec{x}$=(0 1)	1	1
Result $\vec{y}$=(1 -1)	-1	1

FIR Filter Design W2 and Dual W2 (Weights Stay, Inputs and Results Move in Same Direction but at Different Speeds)

$$FIR\ filter:\ y(n) = w_0 x(n) + w_1 x(n-1) + w_2 x(n-2)$$

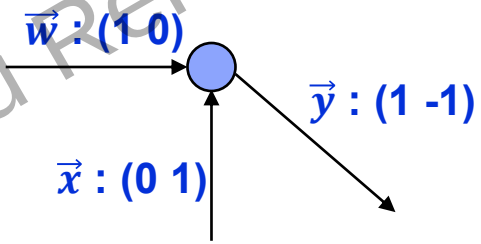
$$\vec{d}^T = (1\ 0),\ \vec{p}^T = (0\ \ 1), W2:\ \vec{s}^T = (1\ \ 2);\ Dual\ W2: \vec{s}^T = (1\ \ -1)$$

Any node in the DG with index $I^T = (i, j)$:

- Is mapped to processor $\vec{p}^T I_i = j$
- Is executed at time $R2:\ \vec{s}^T I_i = i + 2j$; $Dual\ R2:\ \vec{s}^T I_i = i - j$

Since $\vec{s}^T d$=1 we have HUE = $1/|\vec{s}^T d|$ = 100%.

Edge mapping: The 3 fundamental edges corresponding to *weight, input* and *result* can be mapped to corresponding edges in the systolic array as per the following table:



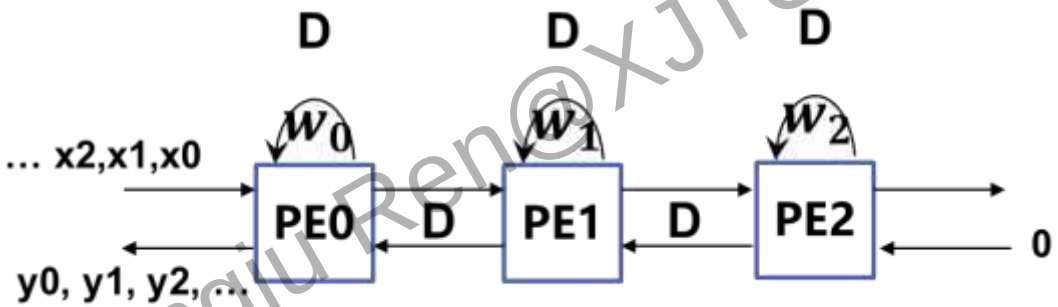
	$W2$		$Dual\ W2$	
$\vec{e}$	$\vec{p}^T\vec{e}$	$\vec{s}^T\vec{e}$	$\vec{p}^T\vec{e}$	$\vec{s}^T\vec{e}$
weight $\vec{w}$=(1 0)	0	1	0	1
Inputs $\vec{x}$=(0 1)	1	2	-1	1
Result $\vec{y}$=(1 -1)	1	1	-1	2

Relationship to Different Transformations

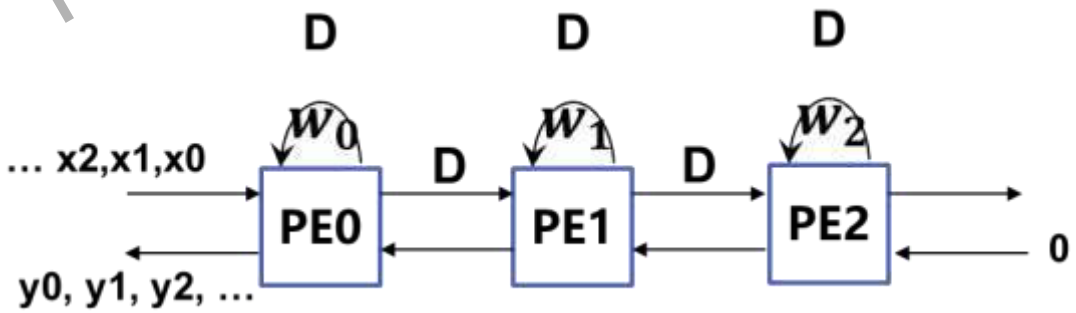
Systolic array architectures with same project vector and processor space vector, but different scheduling vectors can be derived from other transformations

- **Transform, retiming, and pipelining. For example:**

B1: $\vec{d}^T = (1\ 0), \vec{p}^T = (0\ \ 1), \vec{s}^T = (1\ \ 0)$



F: $\vec{d}^T = (1\ 0),\ \vec{p}^T = (0\ \ 1), \vec{s}^T = (1\ \ 1)$



Selection of Schedule Vector

Choose the schedule vector $\vec{s}$ first, then $\vec{d}$ and $\vec{p}$ can be selected according to:

$$\vec{s}^T\vec{d} \neq 0 \qquad \vec{p}^T\vec{d} = 0$$

Procedure to get $\vec{s}$:

- **Capture all the fundamental edges in *reduced dependence graph (RDG)* constructed by *regular iteration algorithm (RIA)* description**
- **Construct the scheduling inequalities**
- **Solve feasible $\vec{s}^T$ ($\vec{s}^T\vec{d}$=1 or $\vec{s}^T\vec{d}$ = *iteration bound* is optimal)**

Scheduling Inequalities

For an edge U to V:

U: $I_u = \begin{pmatrix} i_u \\ j_u \end{pmatrix}$ 

V: $I_v = \begin{pmatrix} i_v \\ j_v \end{pmatrix}$

The scheduling inequality is:

$$S_v \geq S_u + T_u$$

S_u : scheduling time for note U
S_v : scheduling time for node V
T_u : computation time of node U

Linear scheduling:

$$S_u = S^T I_u = (s_1 \quad s_2)\begin{pmatrix} i_u \\ j_u \end{pmatrix}$$
$$S_v = S^T I_v = (s_1 \quad s_2)\begin{pmatrix} i_v \\ j_v \end{pmatrix}$$

Affine scheduling

$$S_u = S^T I_u + \gamma_u = (s_1 \quad s_2)\begin{pmatrix} i_u \\ j_u \end{pmatrix} + \gamma_u$$
$$S_v = S^T I_v + \gamma_v = (s_1 \quad s_2)\begin{pmatrix} i_v \\ j_v \end{pmatrix} + \gamma_v$$

So the scheduling inequalities:

$$S^T I_v + \gamma_v \geq S^T I_u + \gamma_u + T_u \Rightarrow S^T e_{u\to v} + \gamma_v - \gamma_u \geq T_u$$

RDG constructed by RIA

For the FIR filter

$$w(i+1,j) = w(i,j)$$

$$x(i,j+1) = x(i,j)$$

$$y(i+1,j-1) = y(i,j) + w(i+1,j-1)x(i+1,j-1)$$

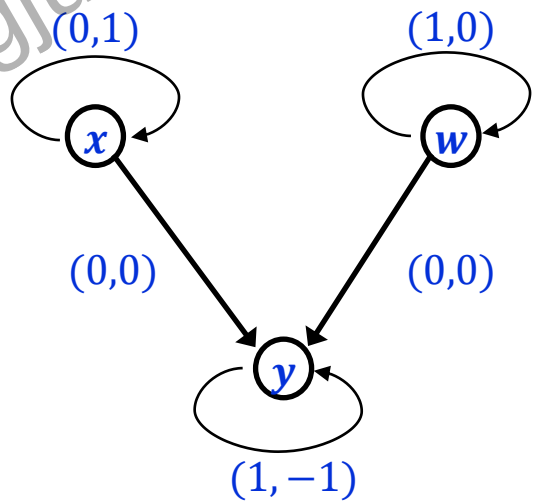
Standard output *Regular Iteration Algorithm* (**RIA**) form

$$w(i,j) = w(i-1,j)$$

$$x(i,j) = x(i,j-1)$$

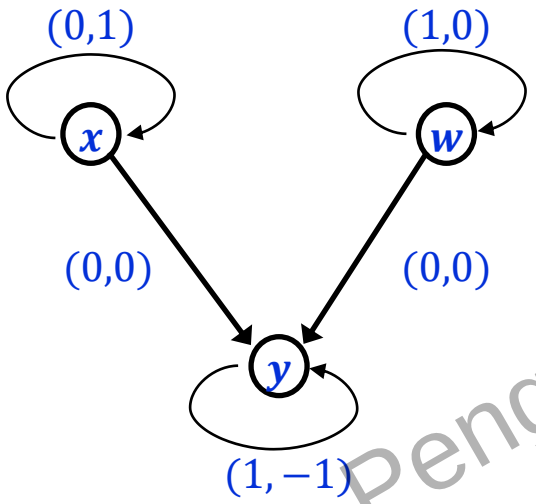
$$y(i,j) = y(i-1,j+1) + w(i,j)x(i,j)$$

The *Reduced Dependence Graph* (**RDG**):



Scheduling Vector Selection

Assume: $T_{mult} = 5, T_{add} = 2, T_{trans} = 1,\ s = \begin{pmatrix} s_1 \\ s_2 \end{pmatrix}$



$x \to x : e = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, s_2 + \gamma_x - \gamma_x \geq 1$

$x \to y : e = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \gamma_y - \gamma_x \geq 0$

$w \to w : e = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, s_1 + \gamma_w - \gamma_w \geq 1$

$w \to y : e = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \ \gamma_y - \gamma_w \geq 0$

$y \to y : e = \begin{pmatrix} 1 \\ -1 \end{pmatrix}, s_1 - s_2 + \gamma_y - \gamma_x \geq 5 + 2 + 1$

Set $\gamma_x = \gamma_w = \gamma_y = 0$: $s_1 \geq 1,\ s_2 \geq 1,\ s_1 - s_2 \geq 8$

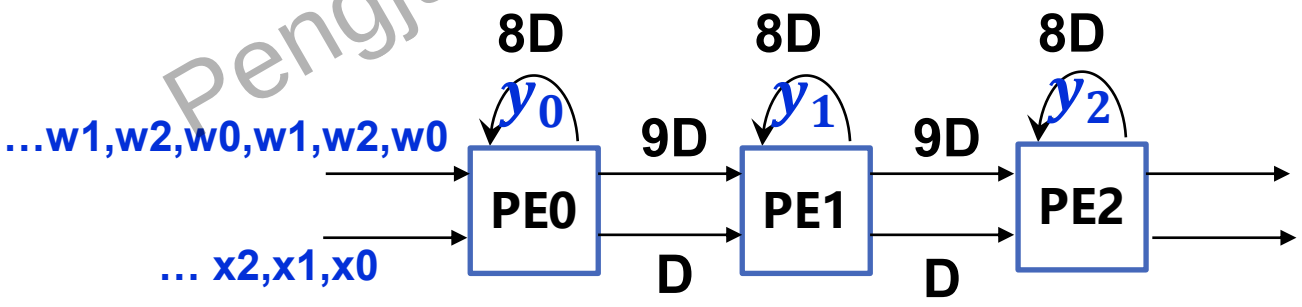
One solution: $s = \begin{pmatrix} 9 \\ 1 \end{pmatrix}$, then choose $\mathrm{d} = (1, -1),\ p = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$

Systolic Architecture Mapping

$$s = \begin{pmatrix} 9 \\ 1 \end{pmatrix},\ \mathrm{d} = (1, -1),\ p = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$\vec{e}$	$\vec{p}^T\vec{e}$	$\vec{s}^T\vec{e}$
weight $\vec{w}$=(1 0)	1	1
Inputs $\vec{x}$=(0 1)	1	9
Result $\vec{y}$=(1 -1)	0	8

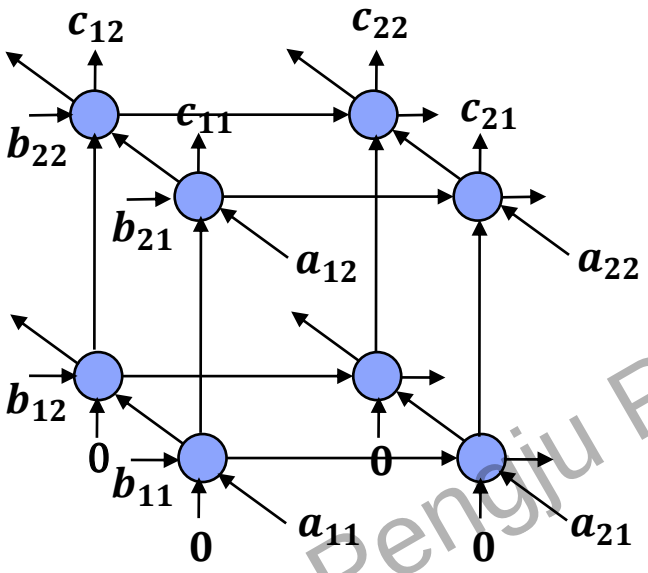
Since $\vec{s}^T d$=8 we have HUE = $1/|\vec{s}^T d|$ = 12.5%.

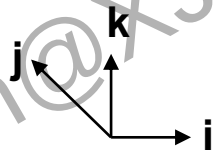


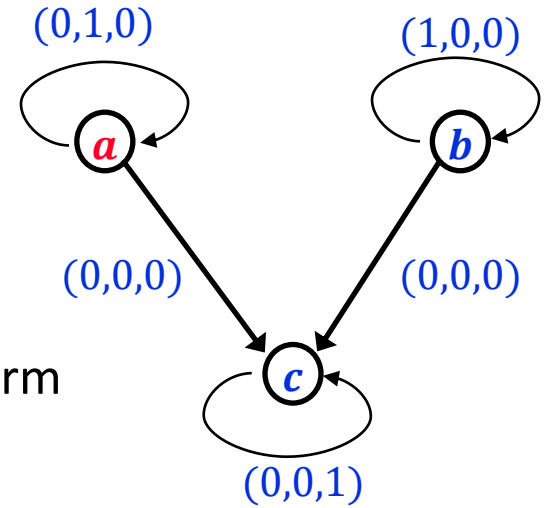
Matrix-Matrix multiplication and 2D Systolic Array Design (1)

C = A x B

$$\begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{bmatrix}$$







Standard output *Regular Iteration Algorithm* (**RIA)** form

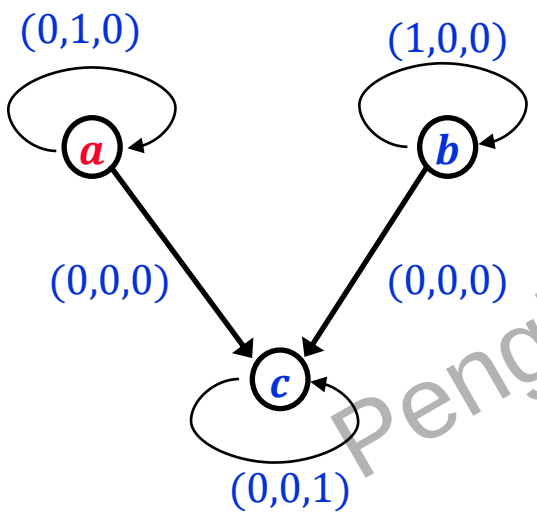
$$a(i,j,k) = a(i,j-1,k)$$

$$b(i,j,k) = b(i-1,j,k)$$

$$c(i,j,k) = c(i,j,k-1) + a(i,j,k)b(i,j,k)$$

Matrix-Matrix multiplication and 2D Systolic Array Design (2)

Assume: $T_{mac} = 1, s = \begin{pmatrix} s_1 \\ s_2 \\ s_3 \end{pmatrix}, \quad \gamma_x = \gamma_w = \gamma_y = 0$



$a \to a: e = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, s_2 \geq 0$

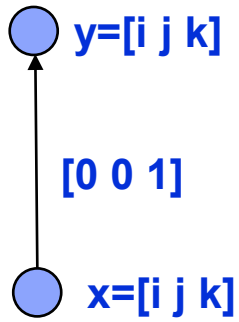
$b \to b: e = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, s_1 \geq 0$

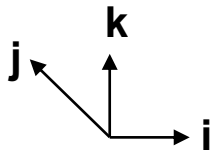
$c \to c: e = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, s_3 \geq 1$

One solution: $s = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$, then choose $\mathrm{d} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$, $p = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}$

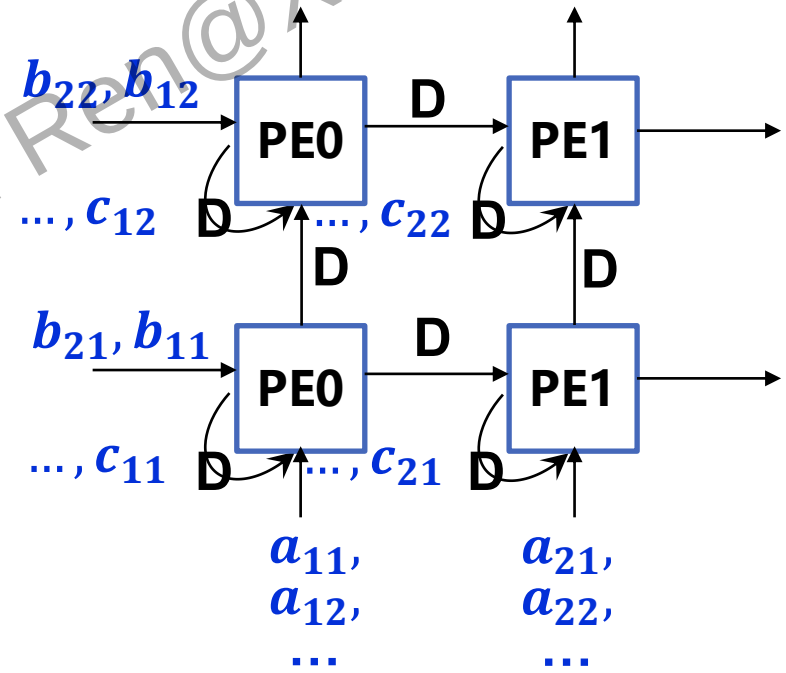
Matrix-Matrix multiplication and 2D Systolic Array Design（S.Y.Kung)

$$\vec{d}^T = (0\ 0\ 1), \vec{p}^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \vec{s}^T = (1\ 1\ 1)$$



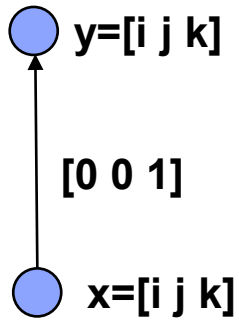


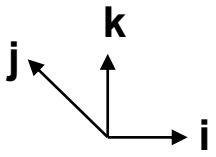
$\vec{e}$	$\vec{p}^T\vec{e}$	$\vec{s}^T\vec{e}$
$\vec{a}$: $(0\ 1\ 0)^T$	$(0\ 1)^T$	1
$\vec{b}$: $(1\ 0\ 0)^T$	$(1\ 0)^T$	1
$\vec{c}$: $(0\ 0\ 1)^T$	$(0\ 0)^T$	1



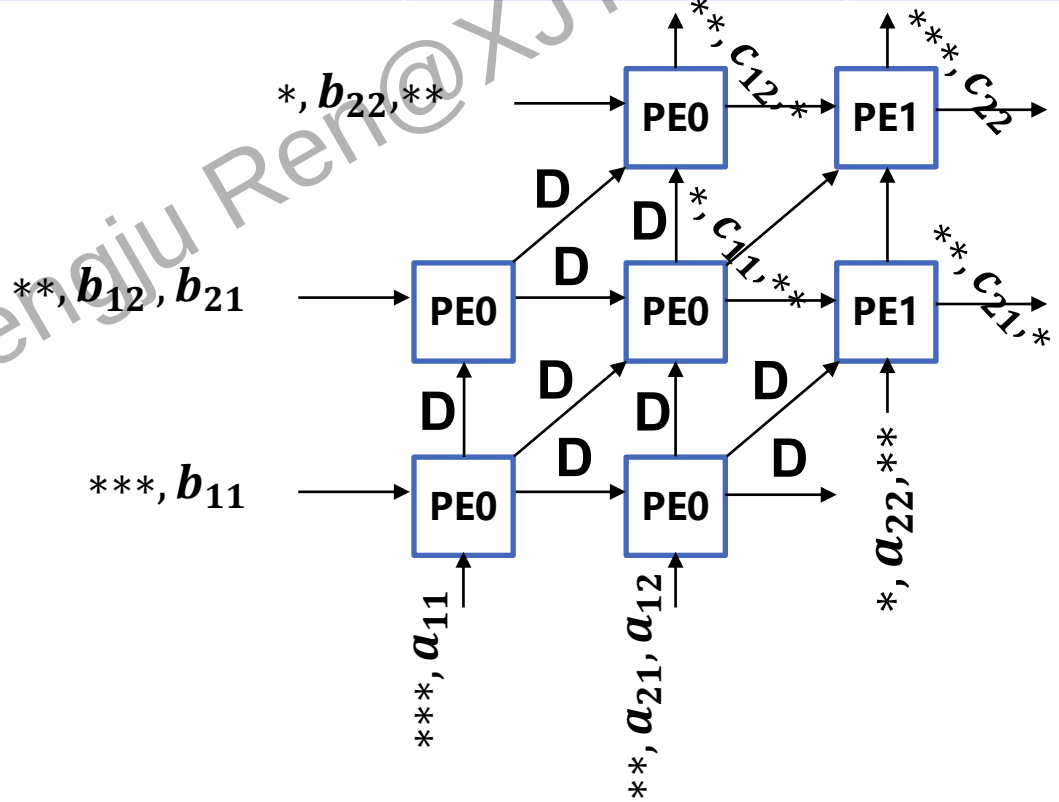
Matrix-Matrix multiplication and 2D Systolic Array Design（S.Y.Kung)

$$\vec{d}^T = (1\ 1\ -1), \vec{p}^T = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \vec{s}^T = (1\ 1\ 1)$$



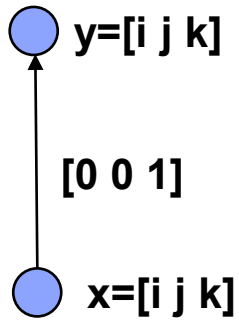


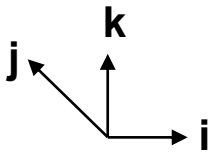
$\vec{e}$	$\vec{p}^T\vec{e}$	$\vec{s}^T\vec{e}$
$\vec{a}$: $(0\ 1\ 0)^T$	$(0\ 1)^T$	1
$\vec{b}$: $(1\ 0\ 0)^T$	$(1\ 0)^T$	1
$\vec{c}$: $(0\ 0\ 1)^T$	$(1\ 1)^T$	1



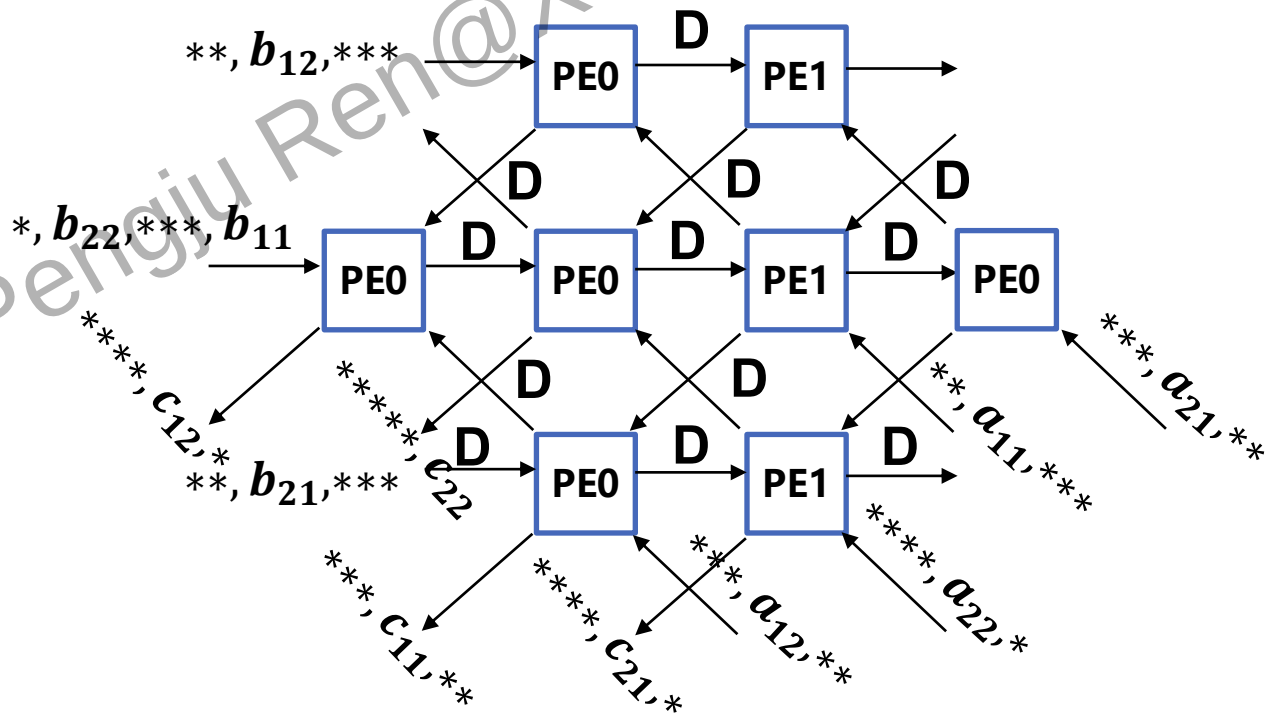
Matrix-Matrix multiplication and 2D Systolic Array Design（S.Y.Kung)

$$\vec{d}^T = (1\ 1\ -1), \vec{p}^T = \begin{bmatrix} 1 & -1 & -1 \\ 0 & 1 & -1 \end{bmatrix}, \vec{s}^T = (1\ 1\ 1)$$





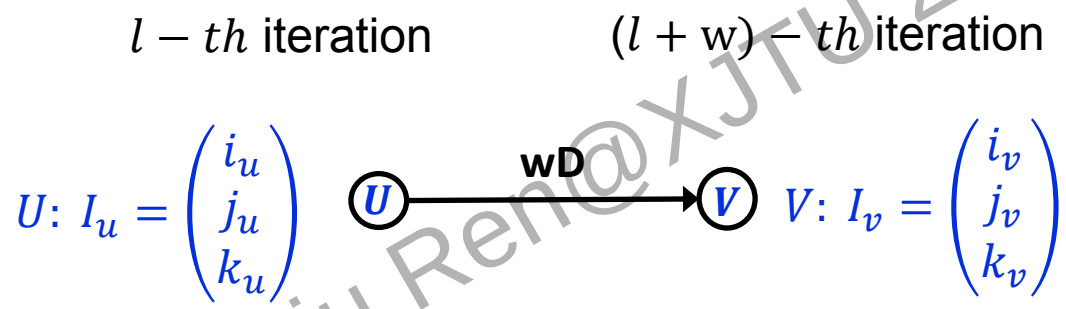
$\vec{e}$	$\vec{p}^T\vec{e}$	$\vec{s}^T\vec{e}$
$\vec{a}$: $(0\ 1\ 0)^T$	$(-1\ 1)^T$	1
$\vec{b}$: $(1\ 0\ 0)^T$	$(1\ 0)^T$	1
$\vec{c}$: $(0\ 0\ 1)^T$	$(-1\ -1)^T$	1



Multi-projection

Systolic Design for Space Representations with Delays

- Can be used for multilevel systolic mapping
- Define N': number of nodes mapped to a processor
- *Iteration period*: $N'|S^T d|$

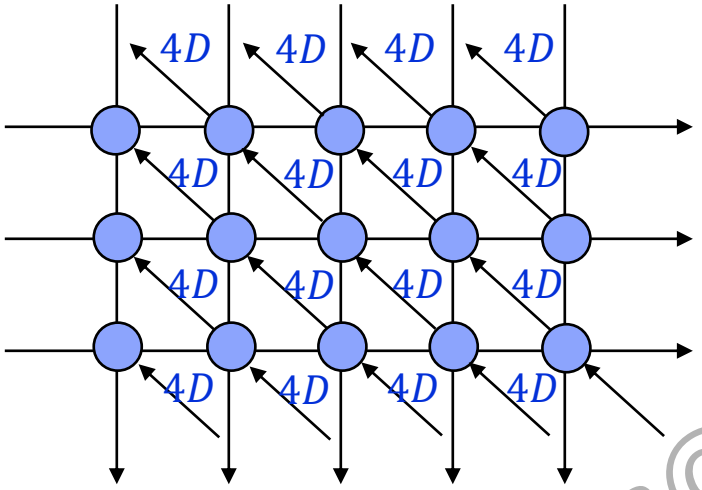


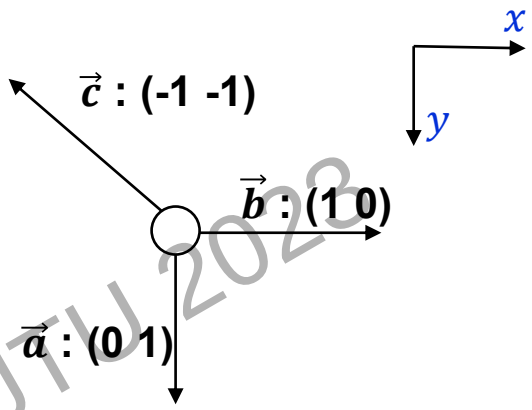
Scheduling inequality

$$S^T I_v + (l+w)N'|S^T d| \geq S^T I_u + lN'|S^T d| + T_u \quad \Rightarrow \quad S^T e_{u\to v} + w\, N'|S^T d| \geq T_x$$

All the mapping equations are the same except for the edge delay mapping $\vec{s}^T\vec{e} \Rightarrow \vec{s}^T\vec{e} + w\, N'|S^T d|$

Example of DG with Delays (1)





Assume: $d = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, N' = 4$

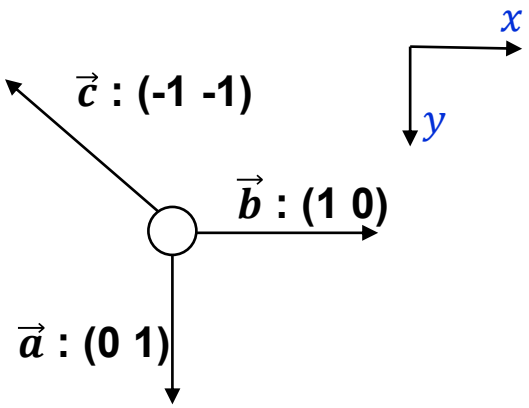
$e_a = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, $(s_1 \quad s_2)\begin{pmatrix} 0 \\ 1 \end{pmatrix} + 0 \; N'|S^T d| \geq 1$ => $s_2 \geq 1$

$e_b = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, $s_1 \geq 1$

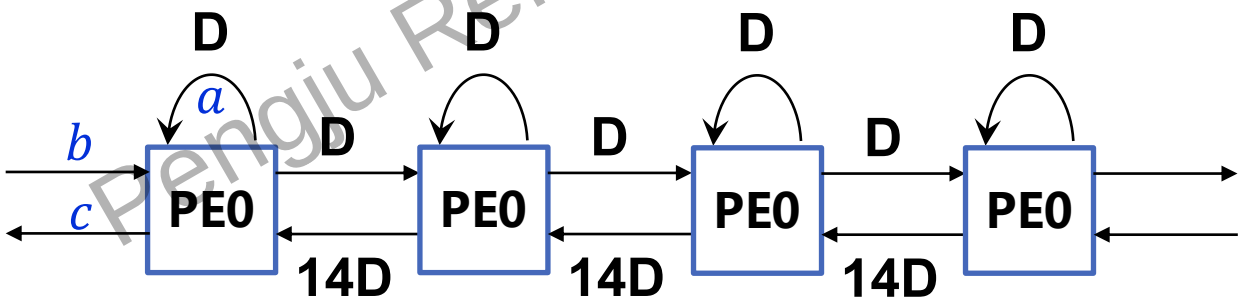
$e_c = \begin{pmatrix} -1 \\ -1 \end{pmatrix}$, $-s_1 - s_2 + 4 \; N'|\, s_1 \, d_1 + s_2 \, d_2| \geq 1 \quad \Rightarrow \quad -s_1 - s_2 + 16 \, s_2 \geq 1 \Rightarrow \; 15 s_2 - \; s_1 \geq 1$

One solution: $s = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, p = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

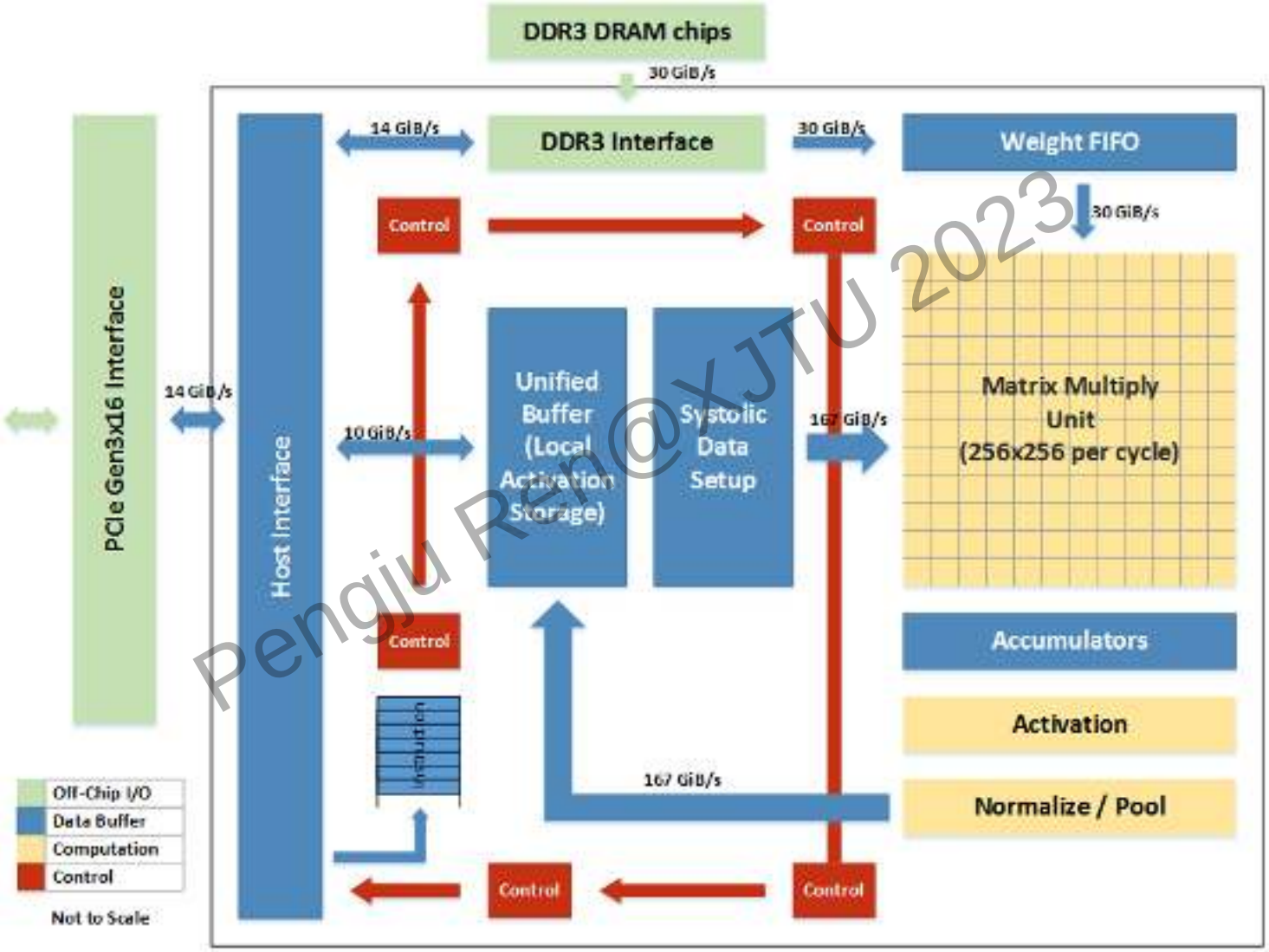
Example of DG with Delays (2)



$\vec{e}$	$\vec{p}^T\vec{e}$	$\vec{s}^T\vec{e} + w\,N'\lvert S^T d\rvert$
$\vec{a}$: (0 1)	0	1
$\vec{b}$: (1 0)	1	1
$\vec{c}$: (−1 −1)	−1	14



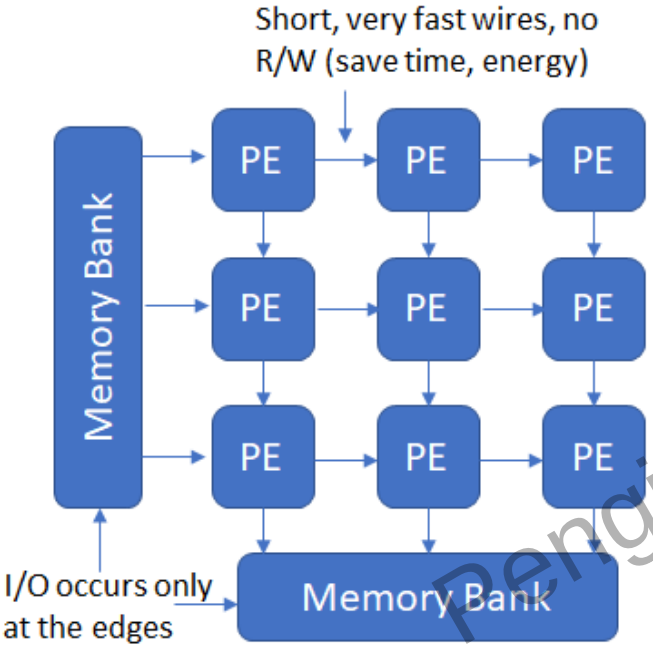
Google TPU-V1 (Systolic Array)

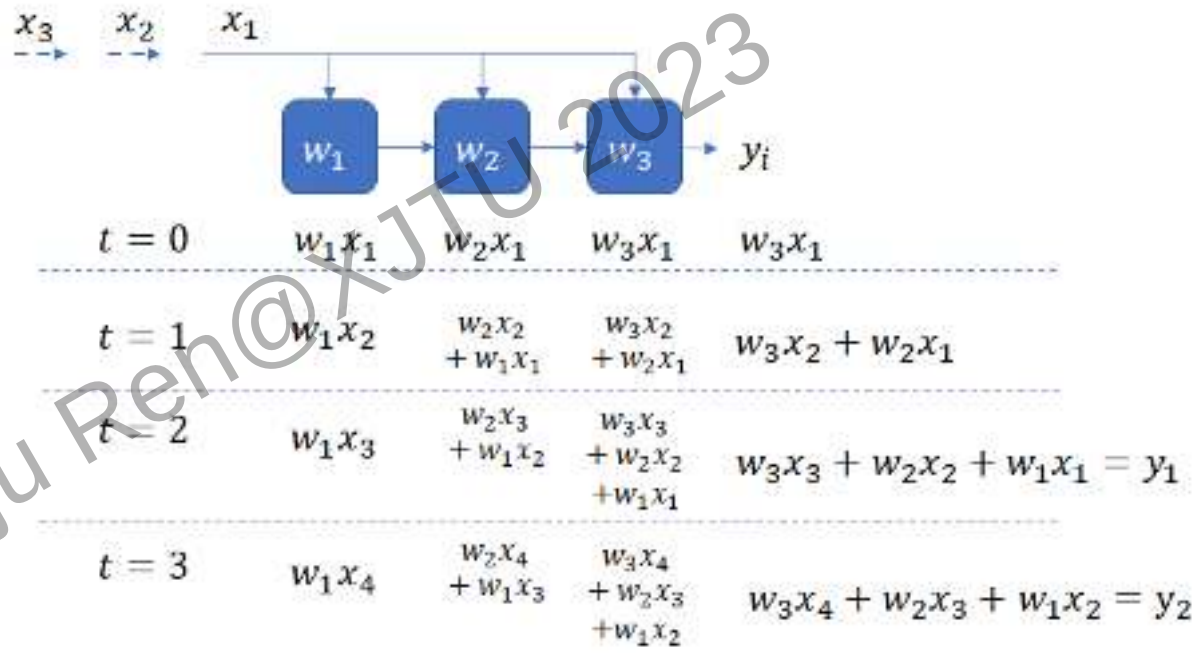


1. H. T. Kung, " Why systolic architectures? " , IEEE Computer, 1982.
2. Norman P. Jouppi, et al."In-Datacenter Performance Analysis of a Tensor Processing Unit“ ISCA’17

Google TPU-V1 (Systolic Array)

"weight stationary" design, as the weights stay stationary and the inputs are streamed in





- **Simplicity reduce the manufacturing Cost & Energy**
- **Hardware Utilization**
- **When Matrix is mismatching with the size of the underlying MXU, will waste memory bandwidth.**
- **No Direct Support for Sparsity**

Summary: Pros and Cons of Systolic Array

Pros:

- Regularity and modular design(Perfect for VLSI implementation)
- Local interconnections
- High degree of pipelining
- Simple I/O subsystem
- High speed and low cost

Cons:

- Global synchronization limits due to signal delays (Improved: Wavefront)
- High bandwidth requirements both for RAM and between PEs
- Poor run-time fault tolerance due to lack of interconnection protocol

Is there a hardware suit for Systolic Array?

YES! Reconfigurable Architecture:

- **Spatial parallel computing like 2D-PE array**
- **Implement hardware structures dynamically**
- **Connection is programmable**

Next Lecture： FPGA & CGRA

*(**F**ield **P**rogrammable **G**ate **A**rray*

&

***C**oarse **G**rained **R**econfigurable **A**rch)*