Introduction of Processor Design for AI Applications

L06 – Parallel Architectures (unfolding)

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

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

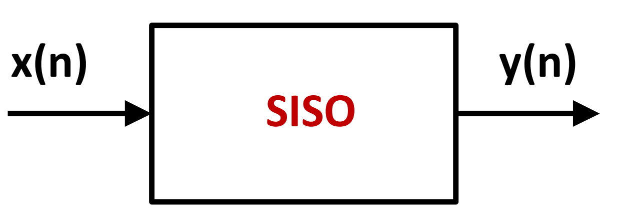
Parallel Processing

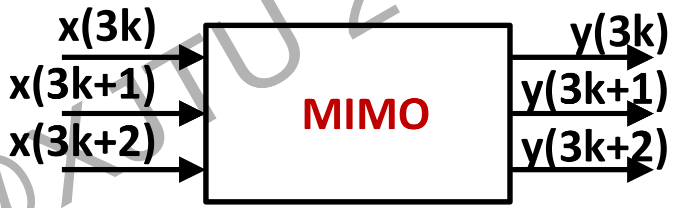
- Multiple outputs are computed in parallel in a clock period
- The effective sampling speed is increased by the level of parallelism
- Can also be used to reduce the power consumption

Parallel Architecture (Unfolding) (1)

Consider a *single-input single-output (SISO)* FIR filter:

$$FIR: \quad y(n) = a * x(n) + b * x(n-1) + c * x(n-2)$$



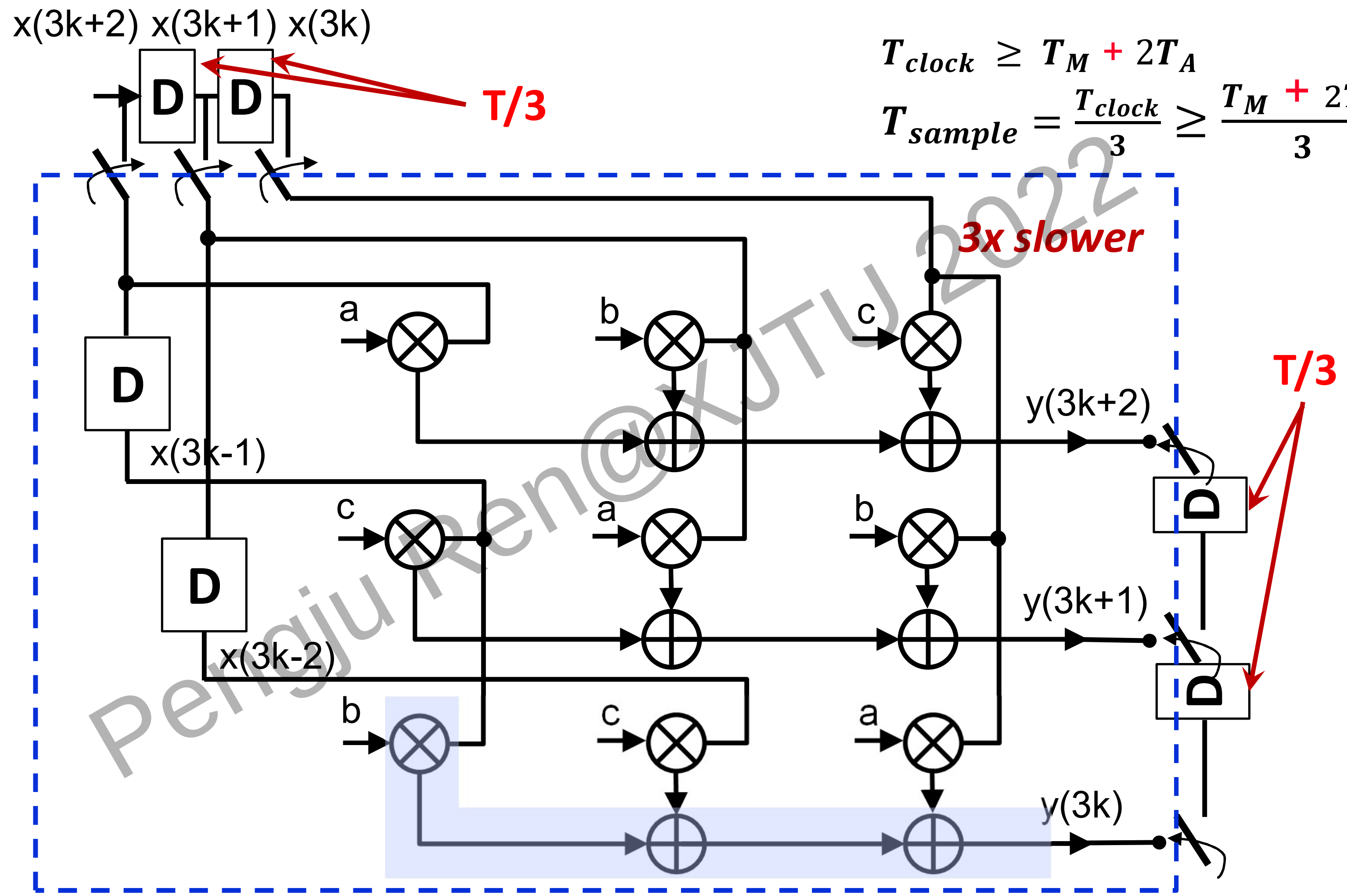


Convert the *SISO* system into an *MIMO* (*multiple-input multiple-output*) system in order to obtain a parallel processing structure.

For example: a 3-level（***unfolding factor=3***）parallel MIMO implementation

$$y(3k) = a * x(3k) + b * x(3k-1) + c * x(3k-2)$$
$$y(3k+1) = a * x(3k+1) + b * x(3k) + c * x(3k-1)$$
$$y(3k+2) = a * x(3k+2) + b * x(3k+1) + c * x(3k)$$

MIMO (multiple-input multiple-output) Architecture

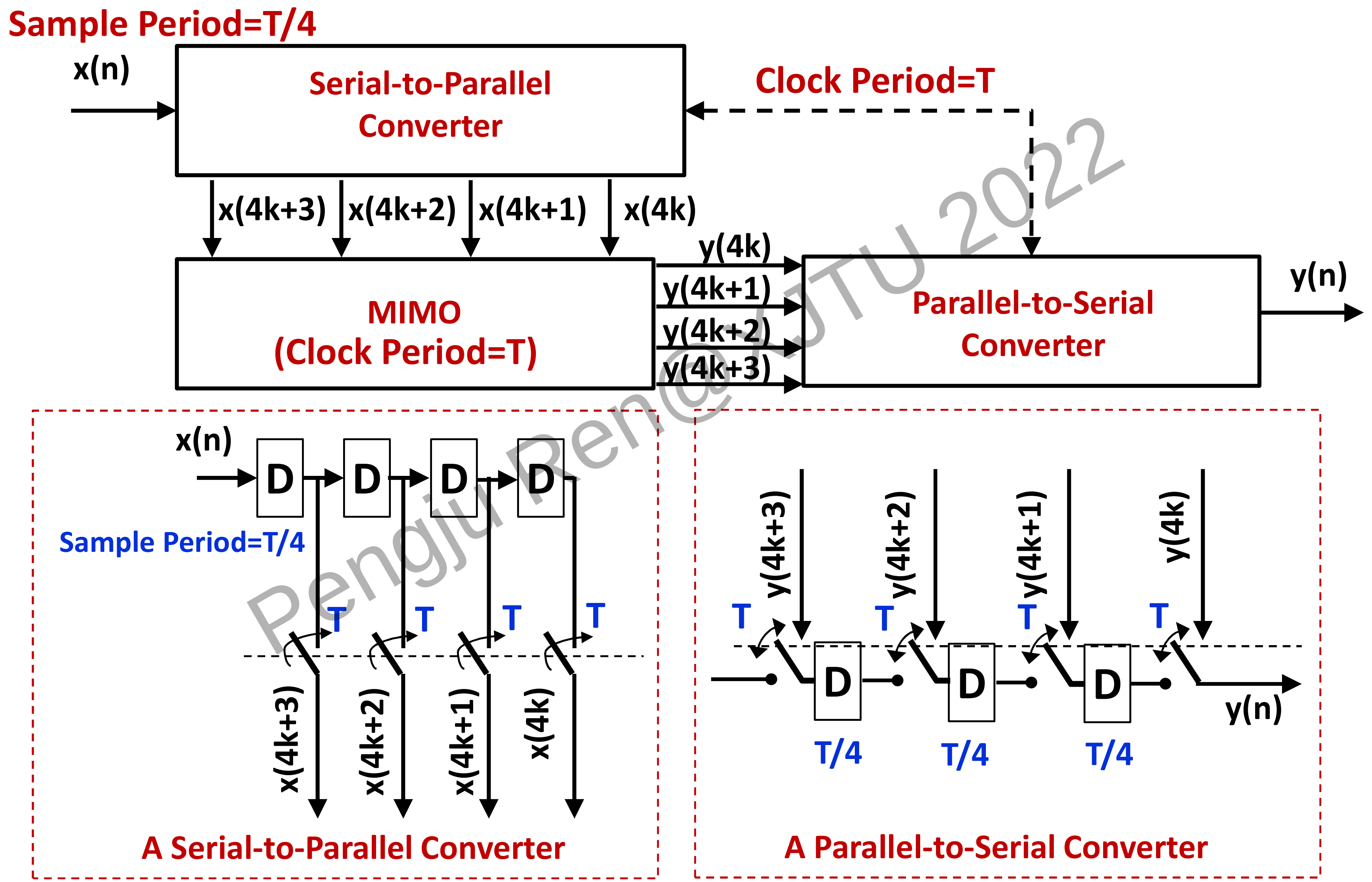


$$T_{clock} \geq T_M + 2T_A$$

$$T_{sample} = \frac{T_{clock}}{3} \geq \frac{T_M + 2T_A}{3}$$

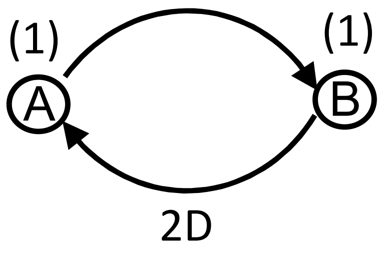
T （NOTE： Clock period of MIMO is 3x of the sample period)

Parallel Architecture (Full system, unfolding factor = 4)



Unfolding == Parallel Processing

Unfolding is a transformation technique that can be applied to a data-stream program to create a new program describing *more than one iterations (Unfolding factor **J**)* of the original program



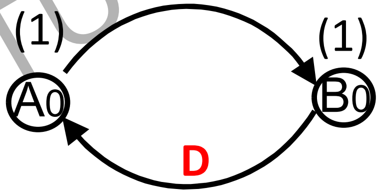
$A_0 \rightarrow B_0 => A_2 \rightarrow B_2 => A_4 \rightarrow B_4 => \cdots$

$A_1 \rightarrow B_1 => A_3 \rightarrow B_3 => A_5 \rightarrow B_5 => \cdots$

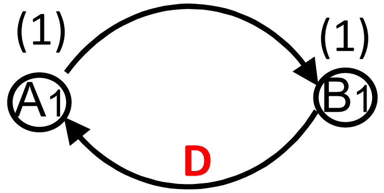
2 nodes & 2 edges

T_∞ = 1+1/2=1 u.t.

A0, A2, A4, A6, … T'_∞ = 1+1/1=2 u.t.



A1, A3, A5, A7, … T''_∞ = 1+1/1=2 u.t.



4 nodes & 4 edges

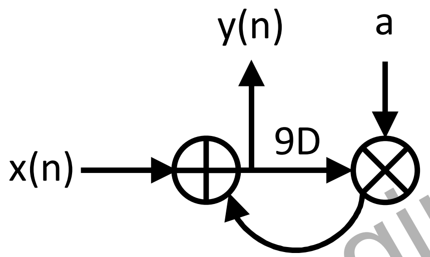
T_∞ = 2/2=1 u.t.

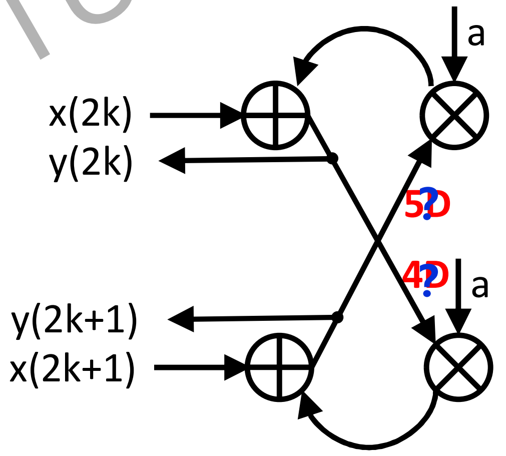
Note that: in unfolded systems, each delay is ***J*- slow**

For each node (edge) in the original DFG, there are ***J*** nodes(edges)

Unfolding

$$y(n) = ay(n-9) + x(n) \xrightarrow{\textit{2x Unfolding (J=2)}} \begin{array}{r} y(2k) = ay(2k-9) + x(2k) \\ y(2k+1) = ay(2k-8) + x(2k+1) \end{array}$$

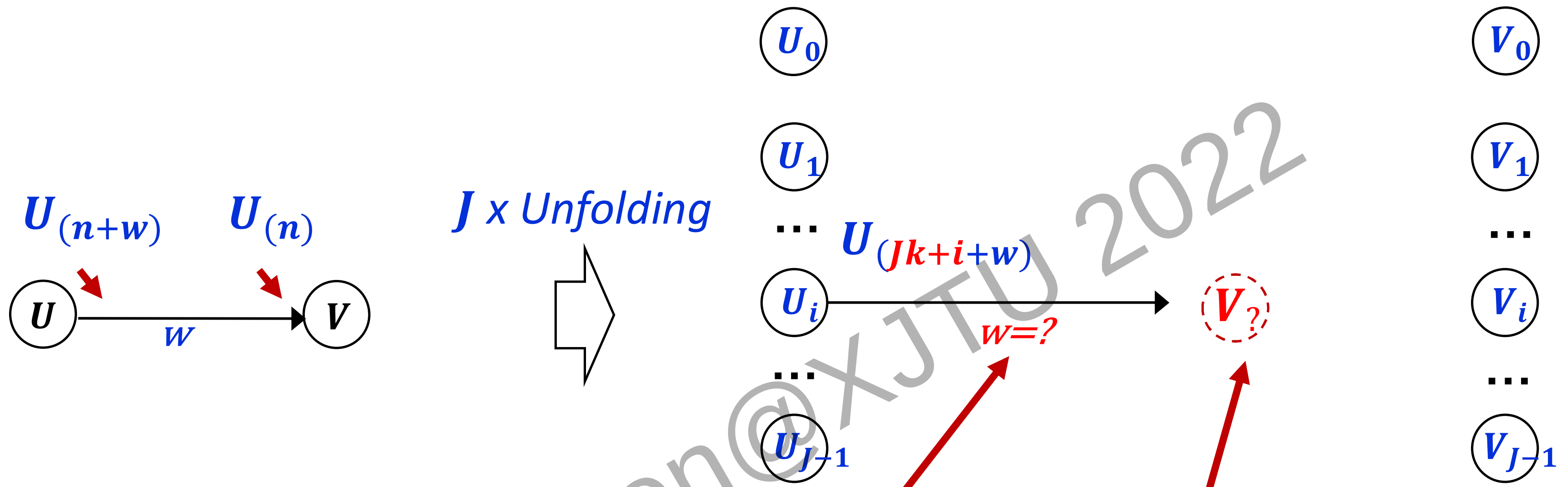




$$y(2k) = ay(2k-9) + x(2k) = ay(2(k-5)+1) + x(2k)$$
$$y(2k+1) = ay(2k-8) + x(2k+1) = ay(2(k-4) + x(2k+1)$$

In a **'J'** unfolded system each delay is ***J***-slow => if input to a delay element is the signal $x(Jk+i)$, the output is $x(J(k-1)+i) = x(kJ+i-J)$.

Algorithm for unfolding (1)

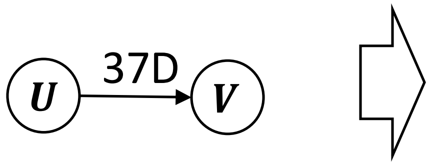


$$Jk + i + w = J\left(k + \boxed{\frac{i+w}{J}}\right) + \boxed{(i+w)\%J}$$

- For each node U in the original DFG, draw J nodes $U_0, U_1, U_2 \dots U_{J-1}$
- For each edge $U \rightarrow V$ with w delays in the original DFG, draw the J edges $U_i \rightarrow V_{(i+w)\%J}$ with $\lfloor (i+w)/J \rfloor$ delays for $i = 0, 1, \dots, J\text{-}1$.

Algorithm for unfolding (2)

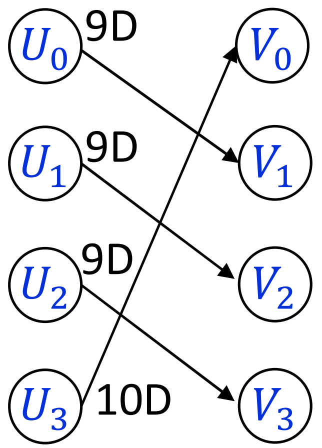
- **For each node U in the original DFG, draw J nodes $U_0, U_1, U_2 \ldots U_{J-1}$**
- **For each edge $U \rightarrow V$ with w delays in the original DFG, draw the J edges $U_i \rightarrow V_{(i+w)\%J}$ with $\lfloor (i+w)/J \rfloor$ delays for $i = 0, 1, \ldots, J-1$.**



$$\lfloor (i+w)\%J \rfloor = \begin{cases} 1, i = 0 \\ 2, i = 1 \\ 3, i = 2 \\ 0, i = 3 \end{cases}$$

$$\lfloor (i+w)/J \rfloor = \begin{cases} 9, & i = 0,1,2 \\ 10, & i = 3 \end{cases}$$

$$U_0 \xrightarrow{9} V_1$$
$$U_1 \xrightarrow{9} V_2$$
$$U_2 \xrightarrow{9} V_3$$
$$U_3 \xrightarrow{10} V_0$$

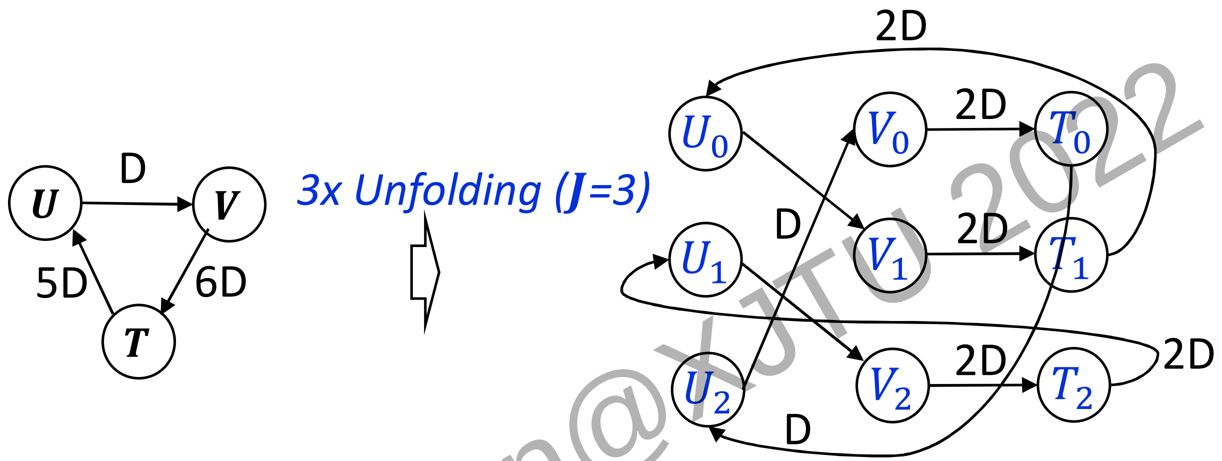


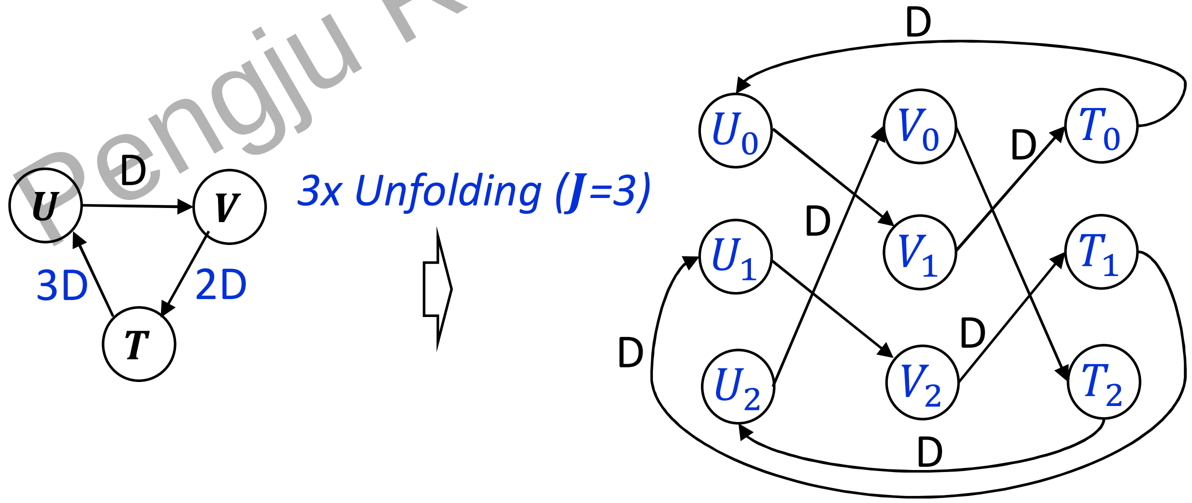
- Unfolding of an edge with w delays in the original DFG produces J-w edges with no delays and w edges with 1 delay in J unfolded DFG for $w < J$.
- Unfolding preserves the number of delays in a DFG.

This can be stated as follows:

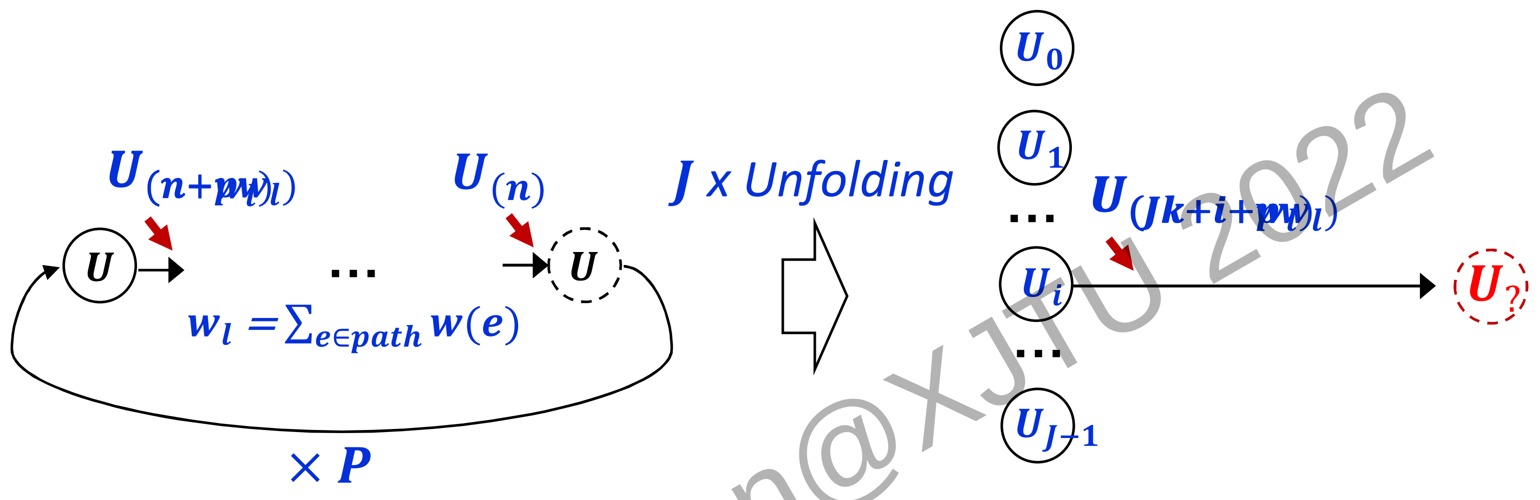
$$\lfloor w/J \rfloor + \lfloor (w+1)/J \rfloor + \cdots + \lfloor (w+J-1)/J \rfloor = w$$

Properties for unfolding (1)





Properties for unfolding (2)

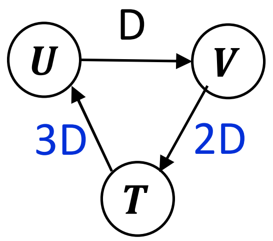


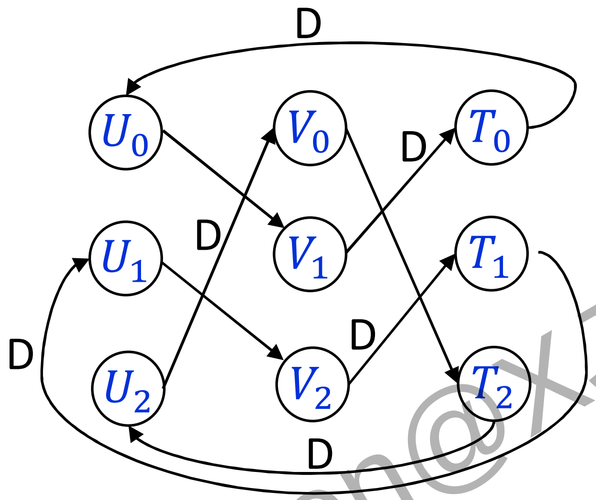
$$Jk + i + pw_l = J\left(k + \frac{i + pw_l}{J}\right) + (i + pw_l)\%J$$

- If $i = (i + pw_l)\%J$ then form a loop in the unfolding DFG

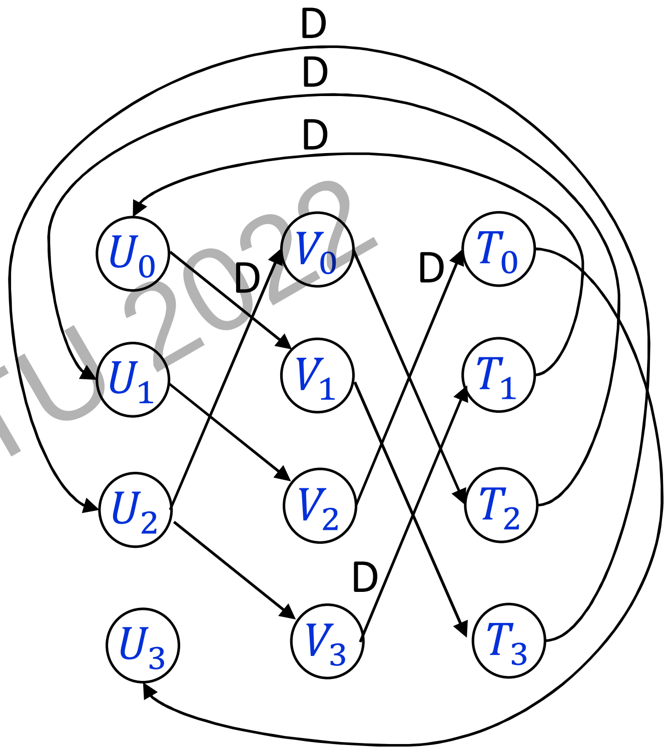
  We would like to find the minimum value of P

Properties for unfolding (3)





$$i = (i + pw_l)\%J = (i + 6p)\%3$$



$$i = (i + pw_l)\%J = (i + 6p)\%4$$

- The smallest positive integer p, q that satisfies $pw_l = qJ$ is $J/gcd(w_l, J)$, $w_l/gcd(w_l, J)$
- J-unfolding of a loop L with w_l delays in the original DFG leads to $gcd(w_l, J)$ loops in the unfolded DFG, and each of these loops contains $w_l/gcd(w_l, J)$ delays and $J/gcd(w_l, J)$ copies of each node that appears in L.
- Unfolding a DFG (iteration bound T_∞) results in a J-unfolded DFG (iteration bound JT_∞)

Applications of Unfolding

Applications of Unfolding

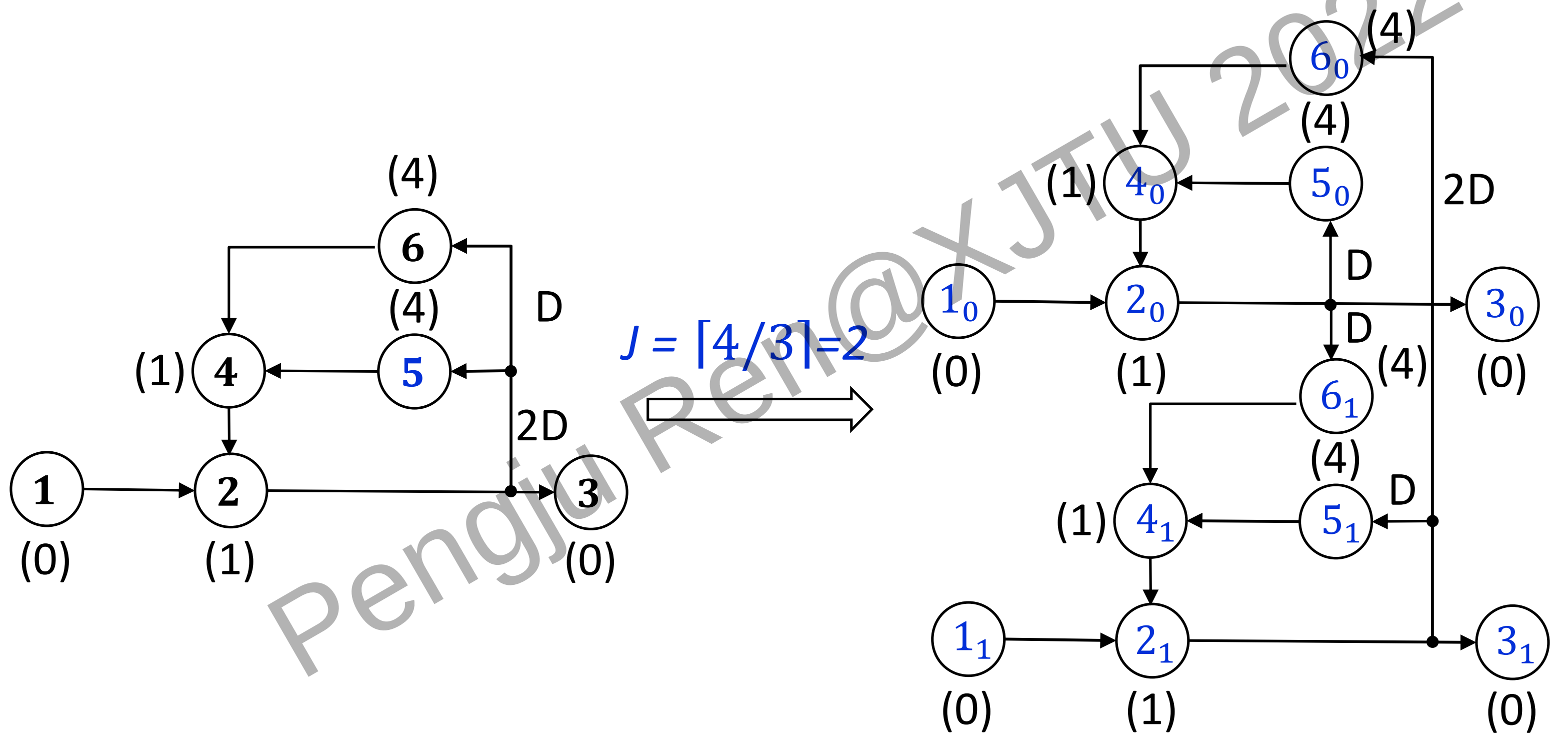
- ☐ Sample Period Reduction
- ☐ Parallel Processing

Sample Period Reduction

- ☐ Case 1 : A node in the DFG having computation time greater than T_∞ .
- ☐ Case 2 : Iteration bound is not an integer.
- ☐ Case 3（Case1+2）: Longest node computation is larger than the iteration bound T_∞ , and T_∞ is not an integer.

Case1

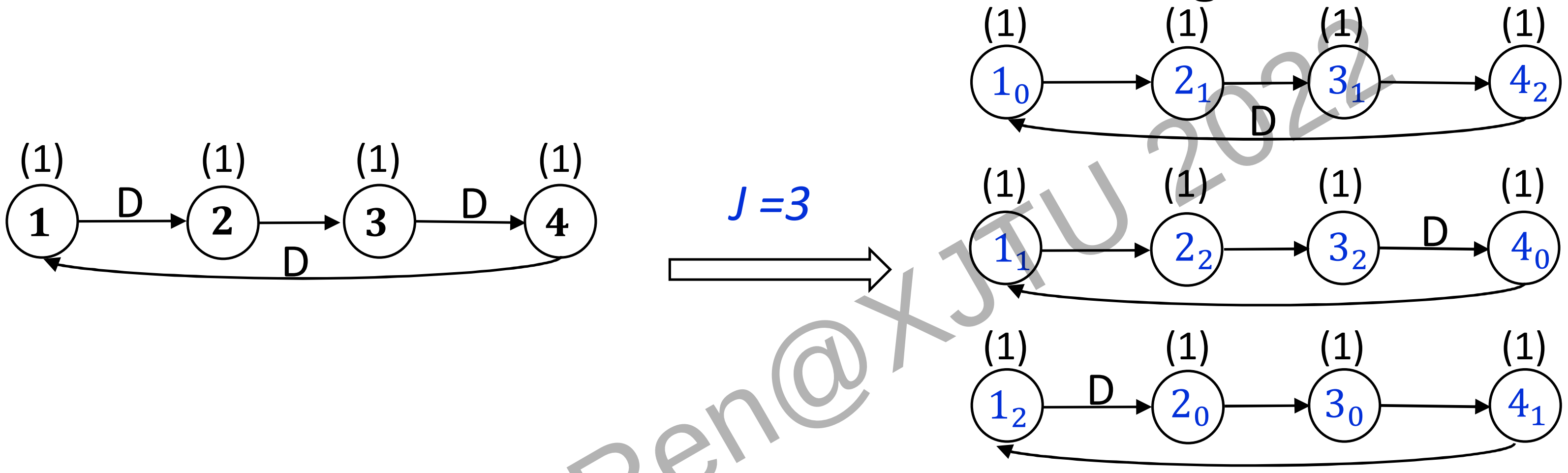
The original DFG cannot have sample period equal to the iteration bound because a node computation time is more than iteration bound



- If the computation time of a node is greater than the iteration bound T_∞, then $\lceil t_U/T_\infty \rceil$ - unfolding should be used.
- In the example, t_5 = 4, and T_∞ = 3, so $\lceil 4/3 \rceil = 2$- unfolding is used.

Case2

The original DFG cannot have sample period equal to the iteration bound because the iteration bound is not an integer.



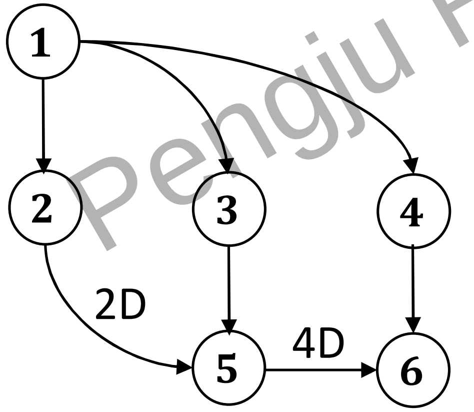
$T_\infty = 4/3$ $T_\infty = 4$

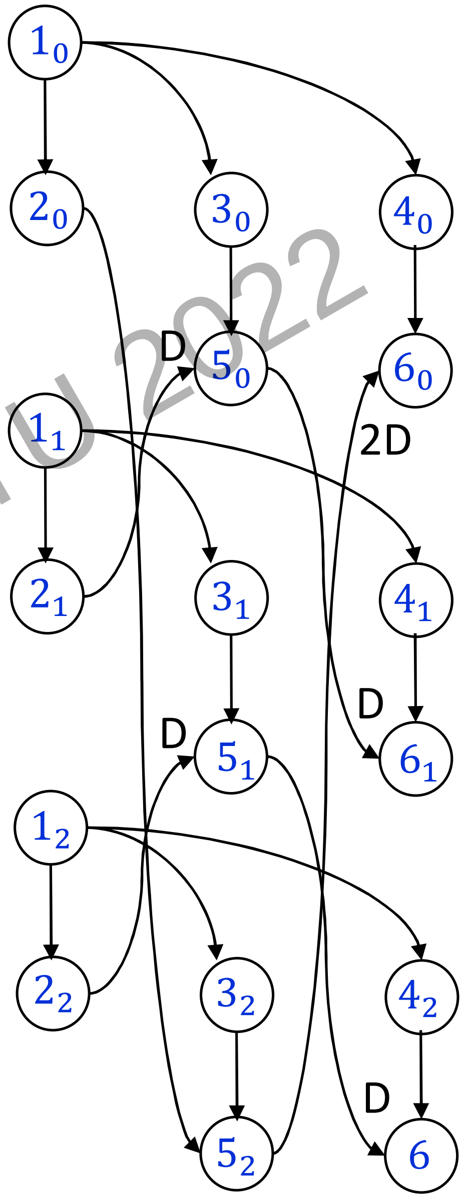
- If a critical loop bound is of the form t_I/w_I where t_I and w_I are mutually co-prime, then w_I -unfolding should be used.
- In the example t_I = 60 and w_I = 45, then t_I / w_I should be written as 4/3 and 3-unfolding should be used.

Case 3（Case1+Case2）: In this case the minimum unfolding factor that allows the iteration period to equal the iteration bound is the min value of J such that JT_∞ is an integer and is greater than the longest node computation time

Parallel Processing

- Word- Level Parallel Processing
- Bit Level Parallel processing
  - Bit-serial processing
  - Bit-parallel processing
  - Digit-serial processing





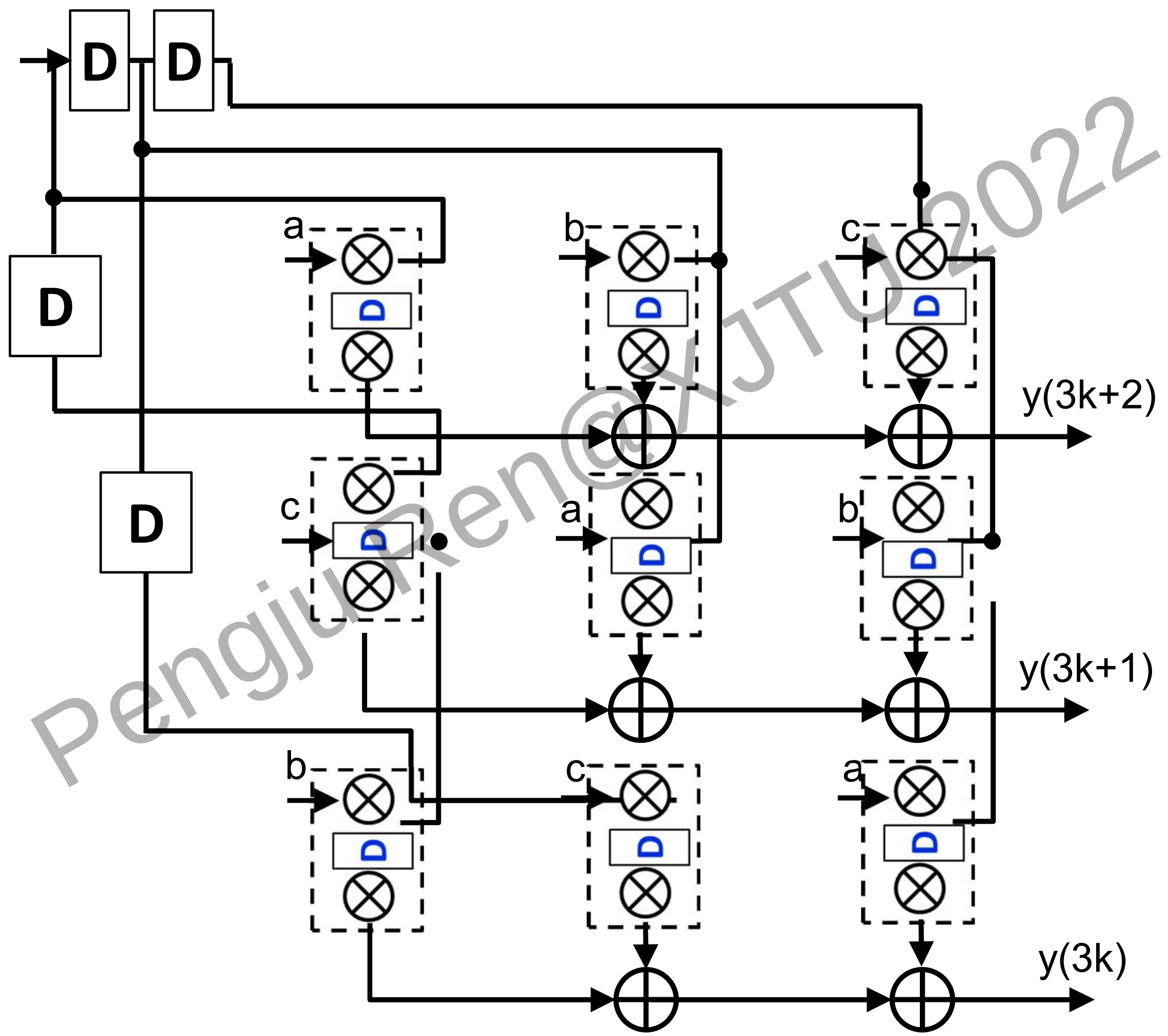
Combining Parallel Processing and Pipelining(1)

In some cases, pipelining can be combined with parallel processing to further increase the speed of the data-stream system

By combining parallel processing (block size: ***L***) and pipelining (pipelining stage: ***M***), the sample period can be reduce to:

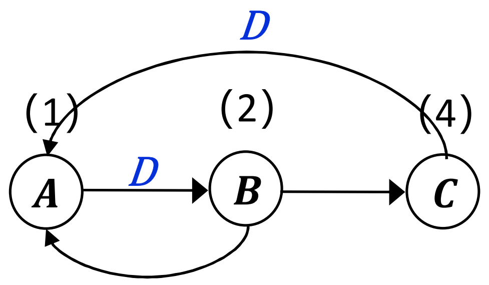
$$T_{iteration} = T_{sample} = \frac{T_{clock}}{LM}$$

Combining Parallel Processing and Pipelining(2)

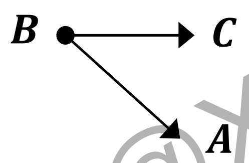


Combining Parallel Processing and Retiming (1)

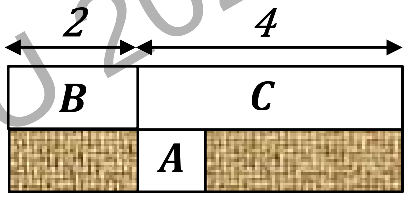
Unfolding can be exploited to reduce the iteration period of DFG
Retiming can be exploited to reduce the critical path (clock period)



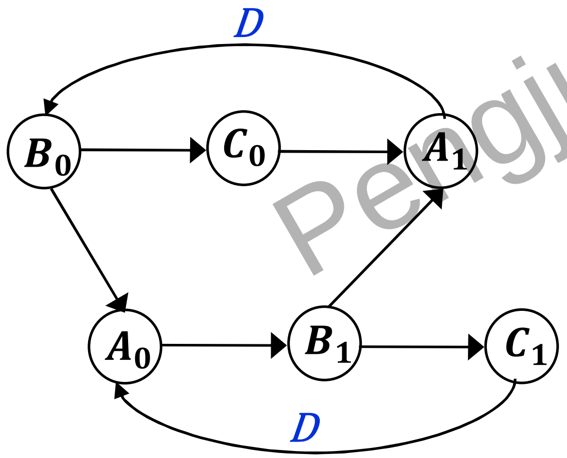
DFG



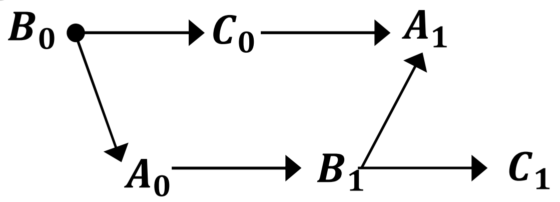
Acyclic precedence graph



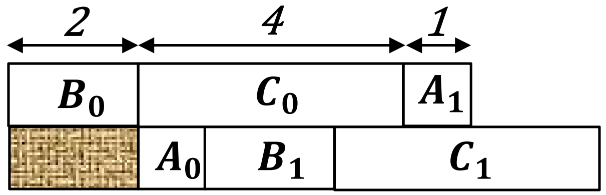
Periodic schedule



2x unfolding DFG



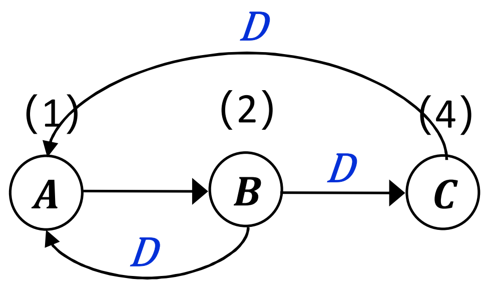
Acyclic precedence graph



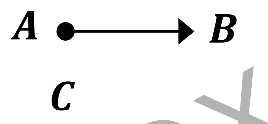
Periodic schedule

Combining Parallel Processing and Retiming (2)

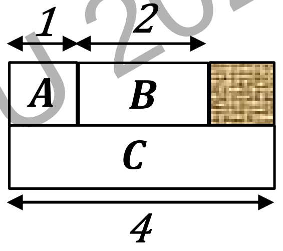
Unfolding can be exploited to reduce the iteration period of DFG
Retiming can be exploited to reduce the critical path (clock period)



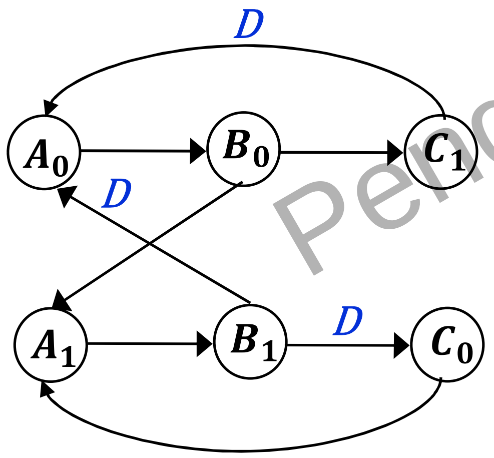
DFG



Acyclic precedence graph



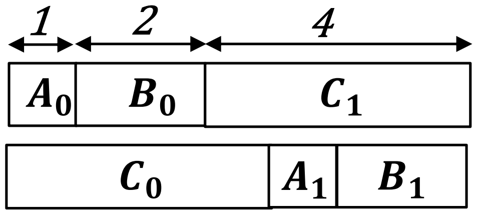
Periodic schedule



2x unfolding DFG

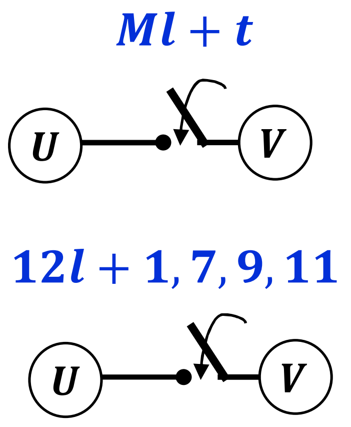
$$\begin{tikzcd} A_0 \arrow[r] & B_0 \arrow[r] \arrow[d] & C_1 \\ C_0 \arrow[r] & A_1 \arrow[r] & B_1 \end{tikzcd}$$

Acyclic precedence graph



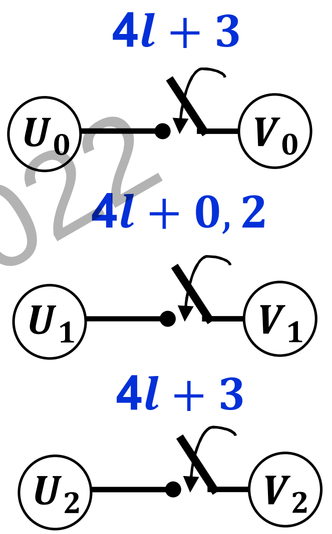
Periodic schedule

Unfolding the switch



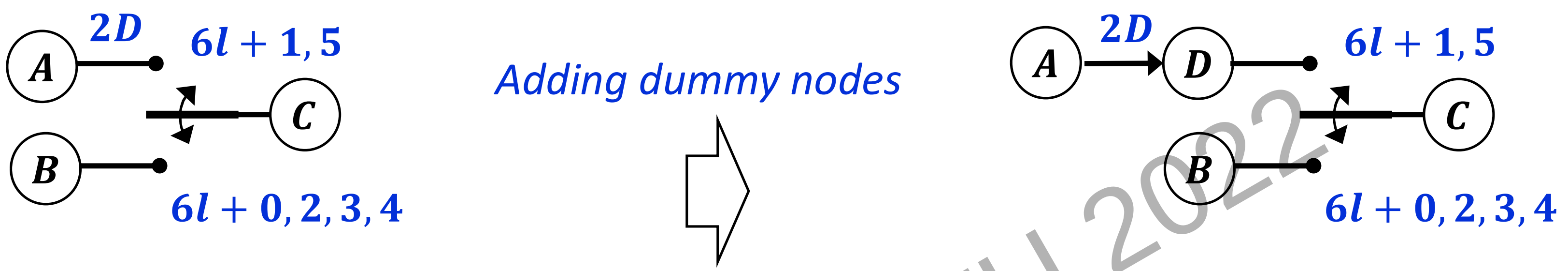
3 x *Unfolding*

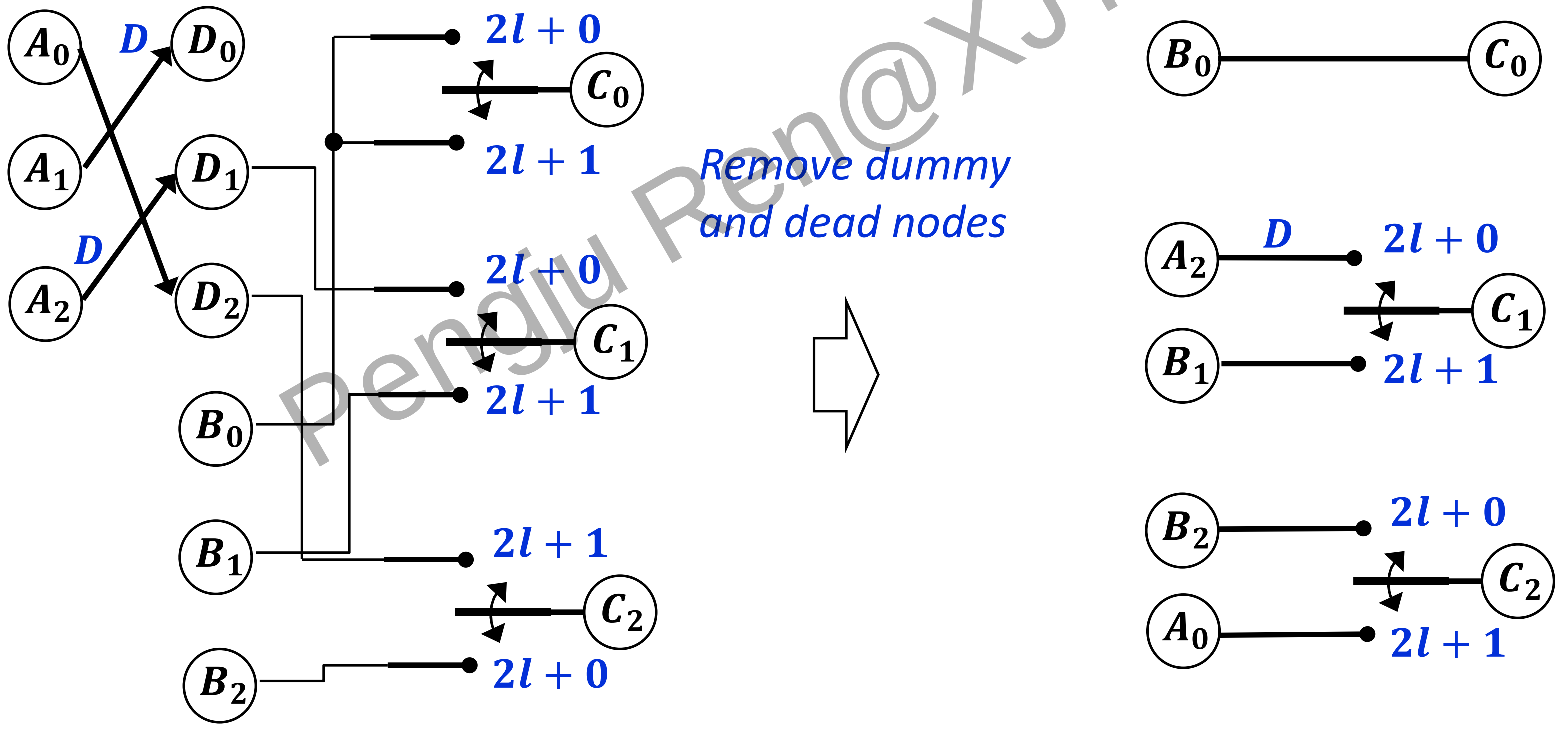
$$12l + 1 = 3(4l + 0) + 1$$
$$12l + 7 = 3(4l + 2) + 1$$
$$12l + 9 = 3(4l + 3) + 0$$
$$12l + 11 = 3(4l + 3) + 2$$



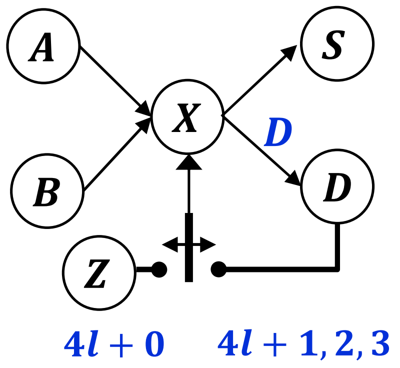
- Assume $M = M'J$
- Assume all edges have *no delays*
- Write the switch instance as $Ml + t = J(M'l + \left\lfloor\frac{t}{J}\right\rfloor) + (t\%J)$
- Draw an edge with no delays in the unfolded graph from the node $U_{t\%J}$ to the node $V_{t\%J}$, which is switched at time instance $(M'l + \left\lfloor\frac{t}{J}\right\rfloor)$

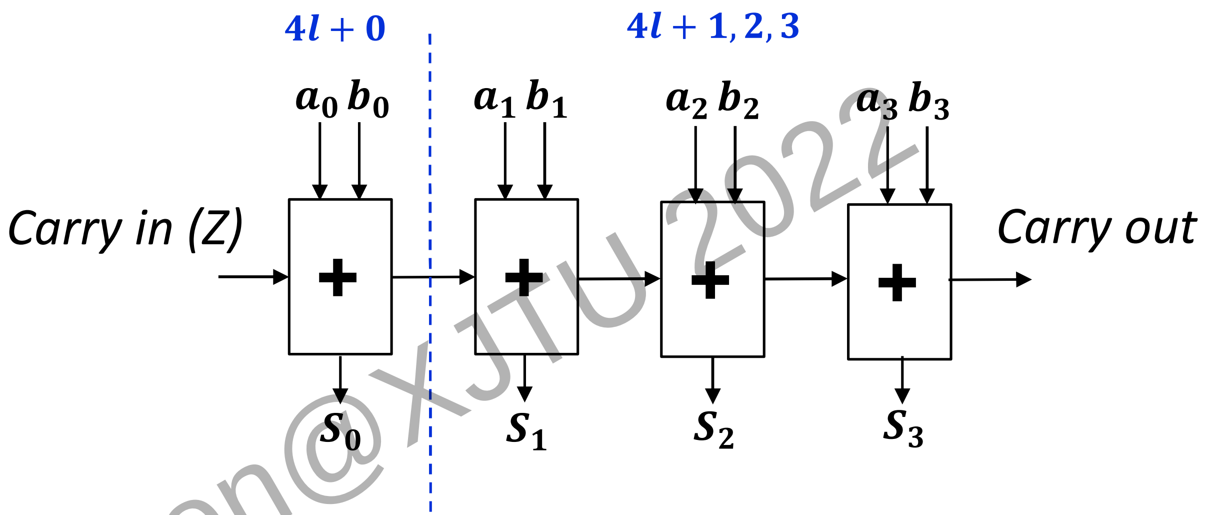
Unfolding the switch (with delays)

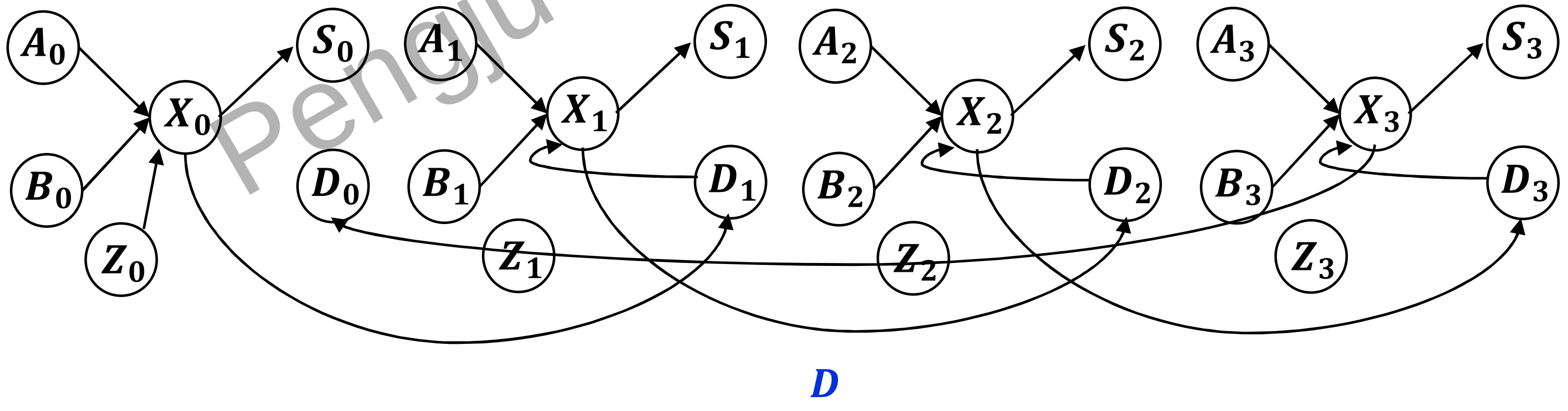




Bit-Parallel Adder(1)

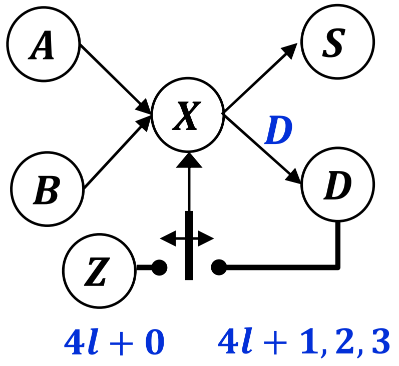


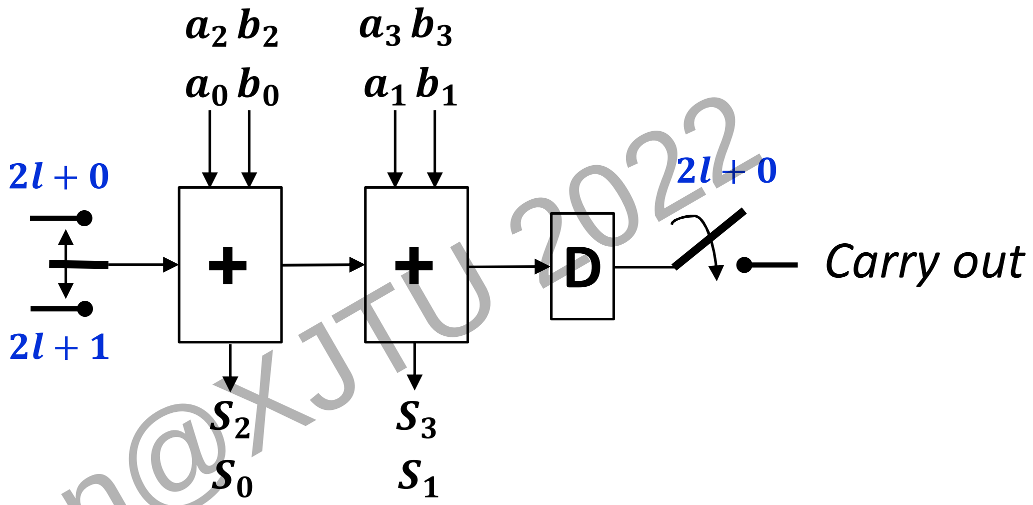


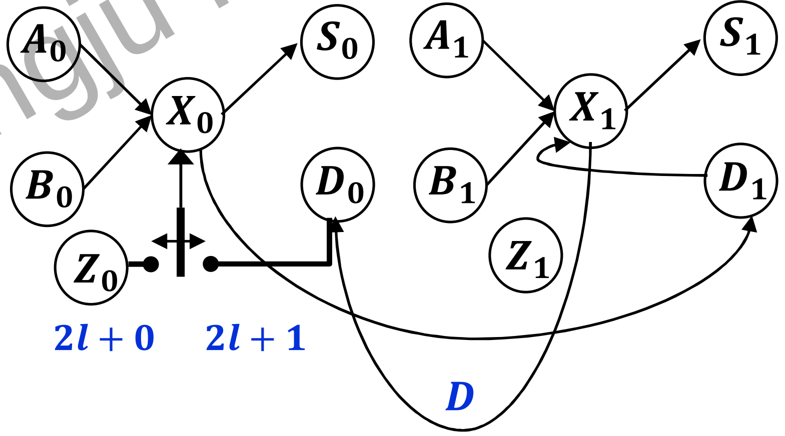


A_0 B_0 Z_0 X_0 S_0 D_0
A_1 B_1 Z_1 X_1 S_1 D_1
A_2 B_2 Z_2 X_2 S_2 D_2
A_3 B_3 Z_3 X_3 S_3 D_3
D

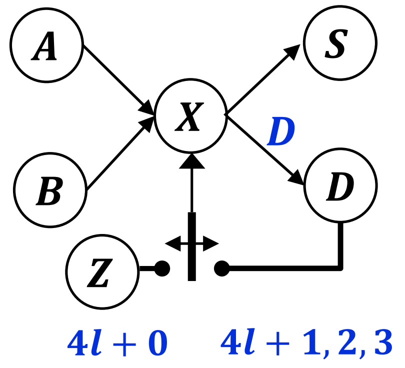
Bit-Parallel Adder(2)

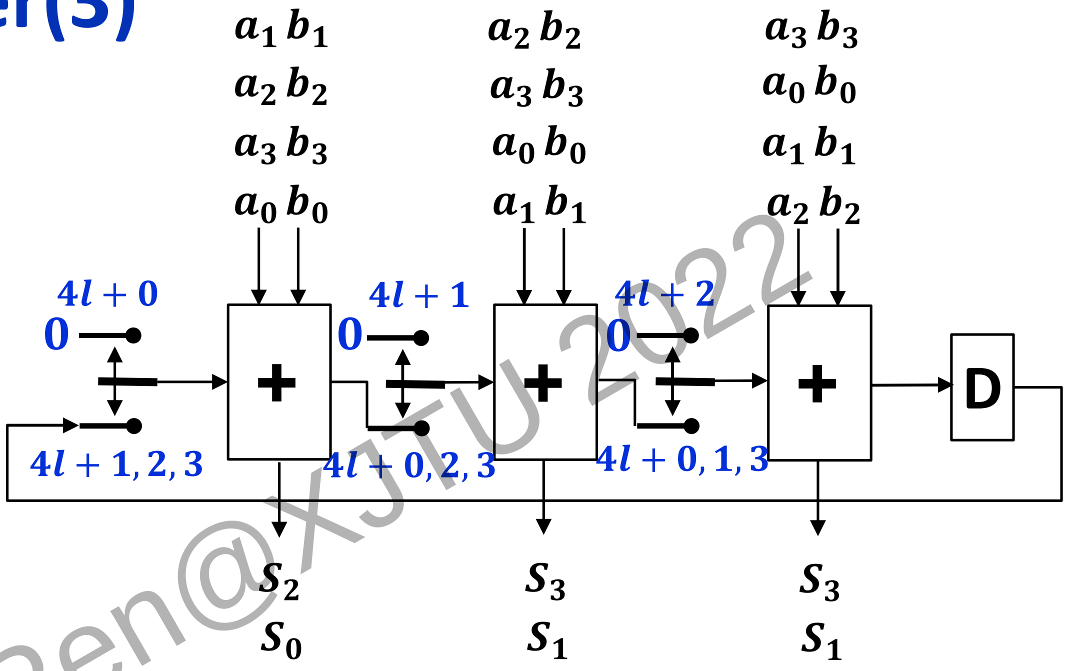




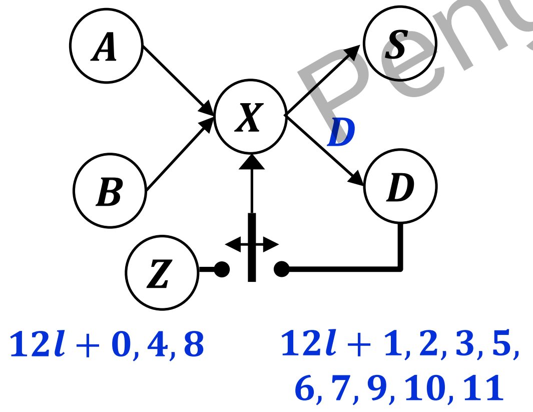


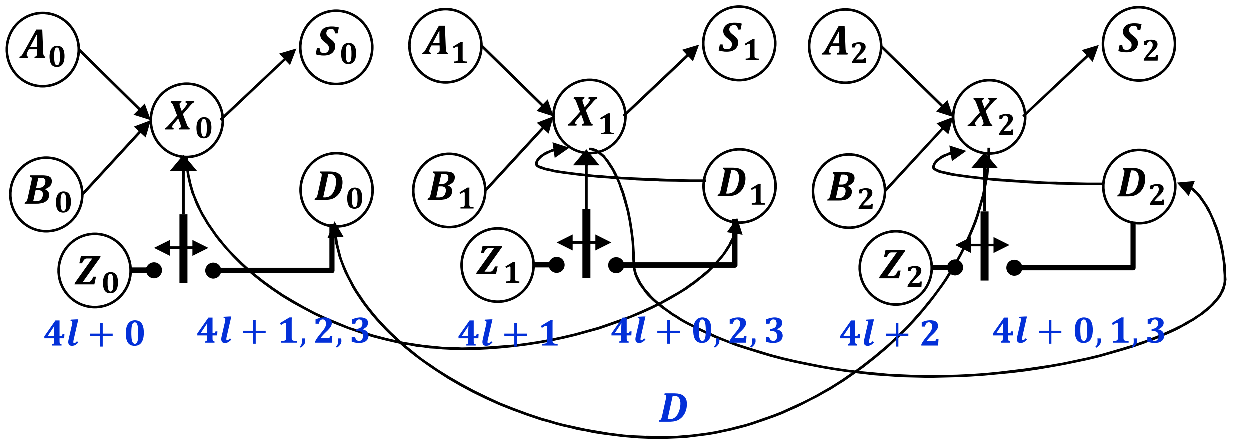
Bit-Parallel Adder(3)





$$L = lcm(M,J), Ml+u = Ll+u+mM \qquad m = 0,\ldots,\frac{L}{M}-1$$





Next Lecture：Folding