DSP Design – Lecture 6

Unfolding

Dr. Fredrik Edman

fredrik.edman@eit.lth.se

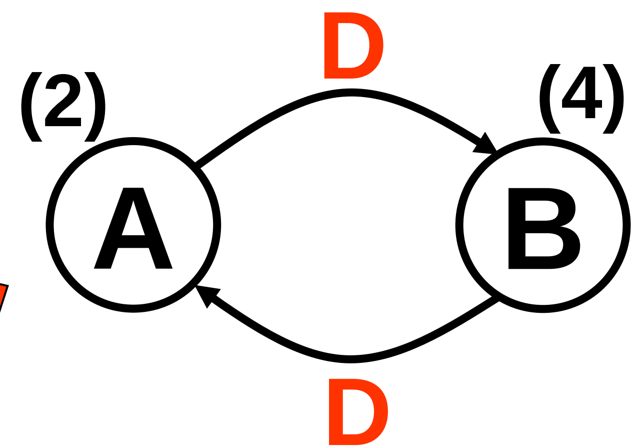
Repetition

Retiming

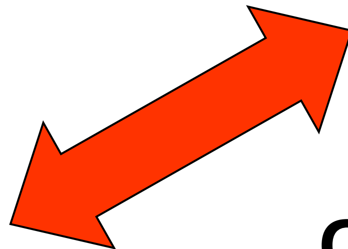
Loop bound = $\frac{T_j}{W_j}$ — T_j: loop computation time; W_j: number of delays in the loop

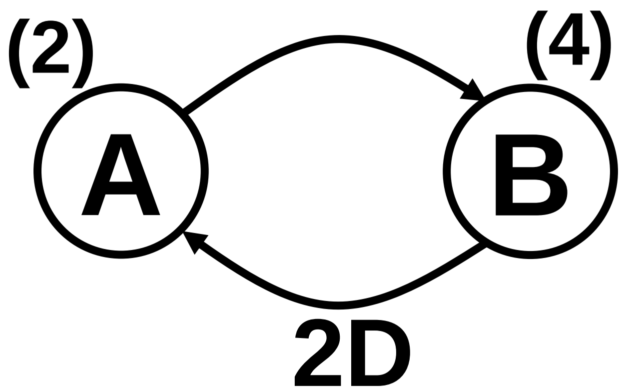
Retiming does not change

- **delay in loop**
- **the iteration bound** $T_\infty = \max_{l \in L} \left\{ \frac{t_l}{w_l} \right\}$



Critical path = 4
Loop bound = 6/2 = 3





Critical path = 6

Loop bound = 6/2 = 3

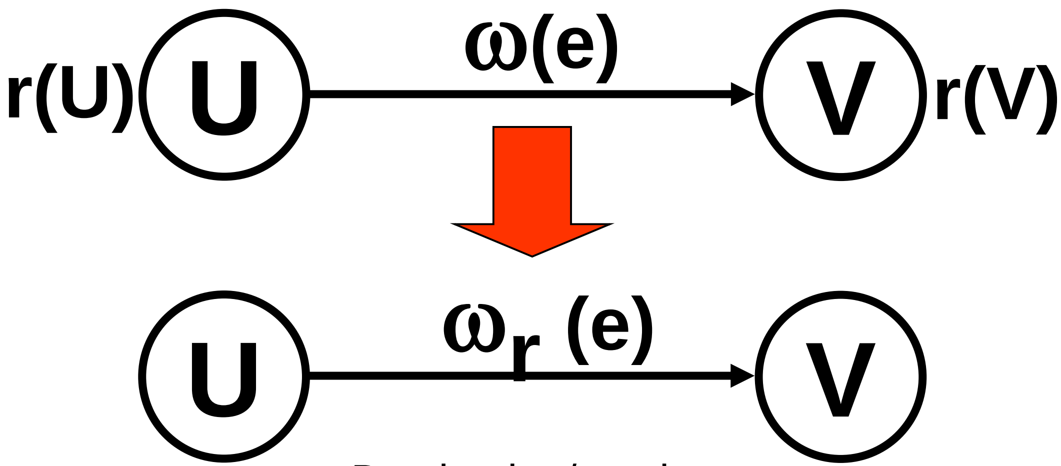
...but it changes the critical path!

Repetition

Retiming Formulation

$\omega(e)$ = weight of edge *e* = # of delays

$r(x)$ = retiming values



Destination/receive

Source/send

$$\omega_r(e) = \omega(e) + r(V) - r(U)$$

r(v)* = # of delays transferred from out-going edges to incoming edges of node *v with w(e)* = # of delays on edge *e

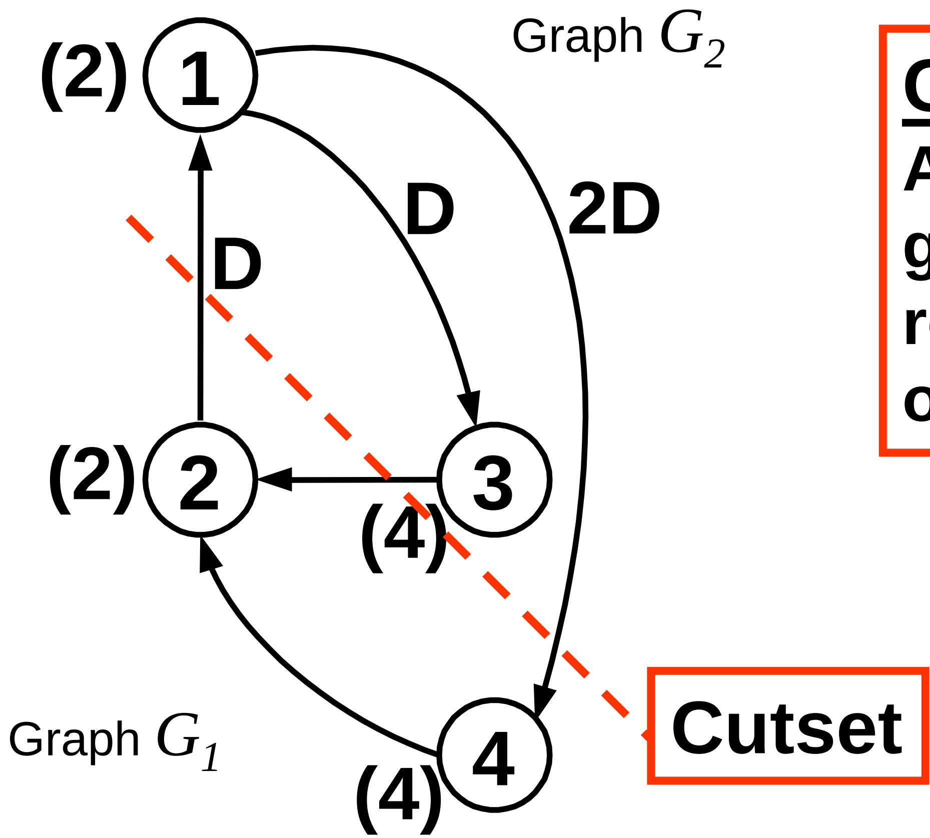
***$w_r(e)$* = # of delays on edge e <u>after</u> retiming**

Valid retiming if all $\omega_r(e) \geq 0$ for all edges!

Repetition

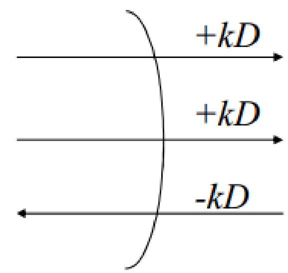
Cutset Retiming

Cutset: A set of edges that if removed, or cut, results in two disjoint graphs.



Cutset Retiming

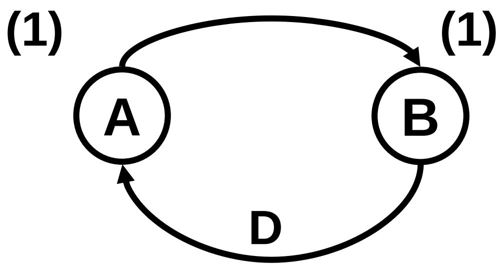
Add *k* delays to edges going one way and remove *k* delays from ones going the other.



Slow Down by k

Repetition

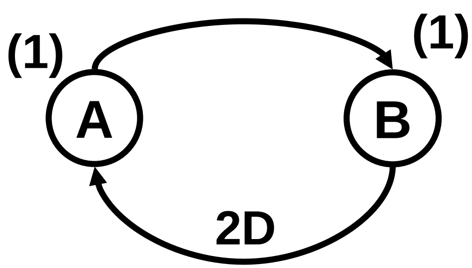
Replace each D by kD



Clock	
0	A0 → B0
1	A1 → B1
2	A2 → B2

T_{clk}= 2t.u.
T_{iter}= 2t.u.

After 2-slow transformation



Clock	
0	A0→B0
1	
2	A1→B1
3	
4	A2→B2

T_{clk}= 2t.u.
T_{iter}=2×2t.u.
=4t.u.

- Input new samples every alternate cycles.
- null operations account for odd clock cycles.
- Hardware utilized only 50% time

Unfolding

Chapter 5

Unfolding

- Unfolding is a structured way to achieve parallel processing
- Unfolding creates a program with more than one iteration
- J is called the unfolding factor

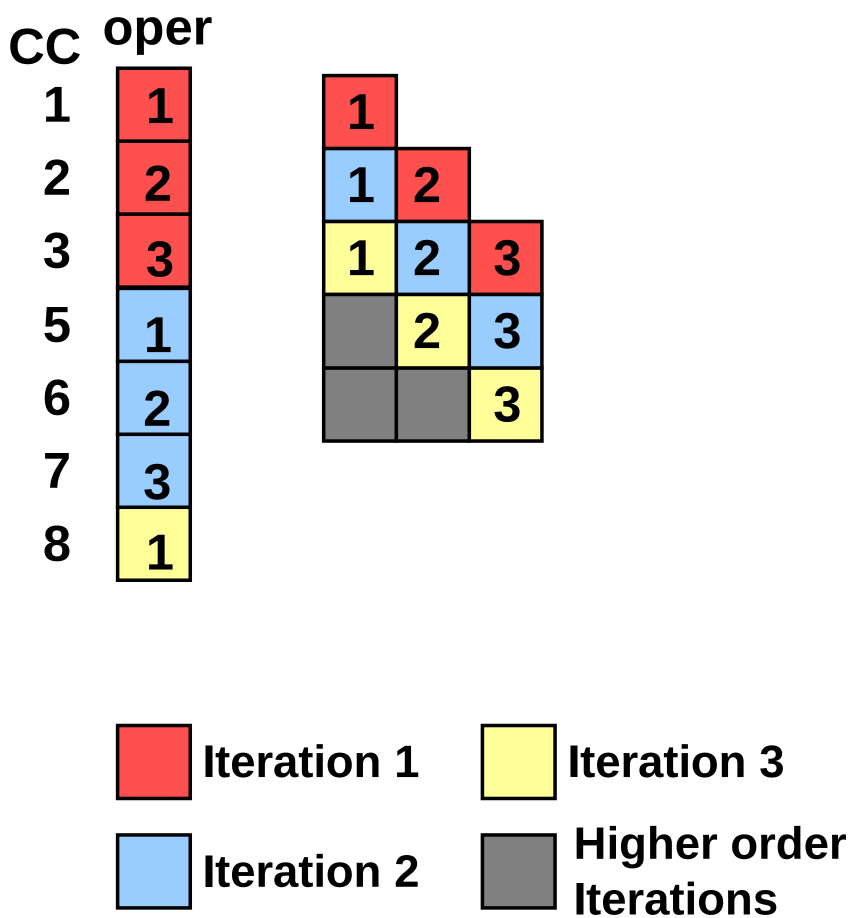
Applications

- Reveal hidden concurrencies so that the program can be scheduled to a smaller iteration period T_{∞}
- Parallel processing
- Bit-serial and Digit-serial

Unfolding in software is called ”loop unrolling” or ”loop unwinding”

- assembly programming
- compiler theory

Example: Loop unrolling + Software Pipelining



GSM Speechcoder

- Org. C-code = 250k cc
- Mod. C-code = 90k cc
- Hand Opt. = 50k cc

Example: Loop unrolling

Example: A procedure in a computer program is to delete 100 items from a collection.

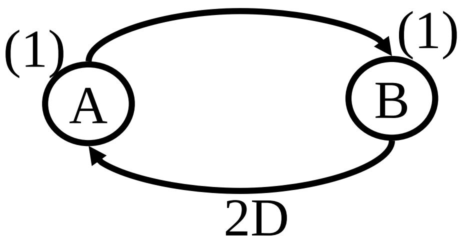
This can be accomplished by means of a for-loop which calls the function delete(item_number) 100 times.

If this part of the program is to be optimized, and the overhead of the loop requires significant resources compared to those for the delete(x) loop, unwinding can be used to speed it up as shown below.

Normal loop	After loop unrolling

`int x;`
`for (x = 0; x < 100; x++)`
`{`
`    delete(x);`
`}`
	`int x;`
	`for (x = 0; x < 100; x += 5)`
	`{`
	`    delete(x);`
	`    delete(x + 1);`
	`    delete(x + 2);`
	`    delete(x + 3);`
	`    delete(x + 4);`
	`}`

Unfolding ≡ Parallel Processing

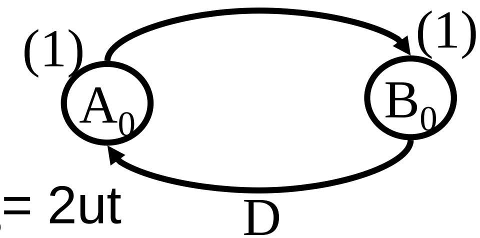


$$T_\infty = \max_{l \in L}\left\{\frac{t_l}{w_l}\right\}$$

$A_0 \rightarrow B_0 \Rightarrow A_2 \rightarrow B_2 \Rightarrow A_4 \rightarrow B_4 \Rightarrow \ldots\ldots$
$A_1 \rightarrow B_1 \Rightarrow A_3 \rightarrow B_3 \Rightarrow A_5 \rightarrow B_5 \Rightarrow \ldots\ldots$

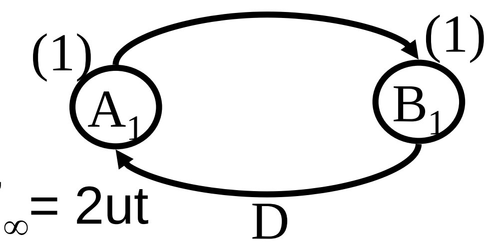
2 nodes & 2 edges & 2 delays
$T_\infty = (1+1)/2 = 1\text{ut}$

2-unfolded



0,2,4,….

$T'_\infty = 2\text{ut}$



1,3,5,….

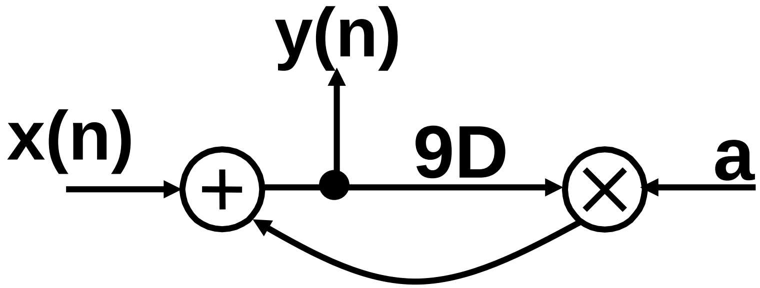
$T'_\infty = 2\text{ut}$

4 nodes & 4 edges & 2 delays
$T_\infty = 2/2 = 1\text{ut}$

- **In a '*J*' unfolded system each delay is *J*-slow ⇨ if input to a delay element is *x*(*kJ* + *m*) ⇨ the output is *x*(*J*(*k*-1) + *m*) = *x*(*kJ* + *m* – *J*).**

J samples

Example: ”unfolding by hand”



$$y(n) = ay(n-9) + x(n)$$

Unfold the system 2-times (J=2) ⇨

Begin by replacing n with $Jk+0,1,\ldots J\text{-}1$. In this case we get $\begin{cases} n = 2k \\ n = 2k+1 \end{cases}$

$$\begin{cases} y(2k) = ay(2k-9) + x(2k) \\ y(2k+1) = ay(2k-8) + x(2k+1) \end{cases}$$

Example: ”unfolding by hand”

We have that
$$\begin{cases} y(2k) = ay(2k-9) + x(2k) \\ y(2k+1) = ay(2k-8) + x(2k+1) \end{cases}$$

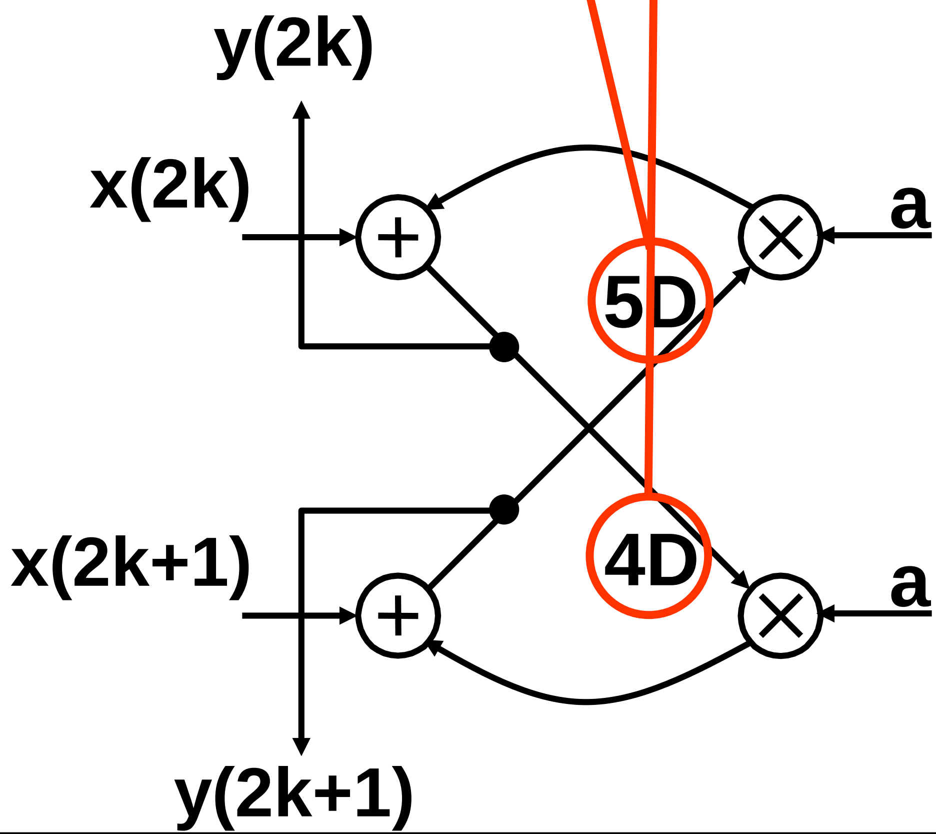
The input to a delay element can be described by $x(kJ+m)$.
After J unfolding, the output from the delay element can be described as $x(J(k\text{-}1)+m)$.
Thus, the above equations can be expressed as

$$\begin{cases} y(2k) = ay(2(k-5)+1) + x(2k) \\ y(2k+1) = ay(2(k-4)+0) + x(2k+1) \end{cases}$$

From above we can see that the inputs to the system are $x(2k)$, $x(2k+1)$ and the constant a, the outputs are $y(2k)$ and $y(2k+1)$. The terms $(k\text{-}5)$ and $(k\text{-}4)$ relates to the number of delays in the two branches ($y(2k)$ and $y(2k+1)$) in the unfolded system.

Example: "unfolding by hand"

$$\begin{cases} y(2k) = ay(2(k-5)+1) + x(2k) \\ y(2k+1) = ay(2(k-4)+0) + x(2k+1) \end{cases}$$



Not trivial even for a simple graph! Need a method!!

Definitions

$\lfloor x \rfloor$ is the floor of x, largest integer $\leq x$

$\lceil x \rceil$ is the ceiling of x, smallest integer $\geq x$

$a\%b$ remainder after a/b

Examples

x	Floor $\lfloor x \rfloor$	Ceiling $\lceil x \rceil$
−1.1	−2	−1
0	0	0
1.01	1	2
2.9	2	3
3	3	3

$\lfloor x \rfloor$ **is the floor of** x, **largest integer** $\leq x$

$\lceil x \rceil$ **is the ceiling of** x, **smallest integer** $\geq x$

Example

$$a\%b \text{ remainder after } a/b$$

In arithmetic, the remainder is the integer "left over“ after dividing one integer by another to produce an integer quotient (integer division).

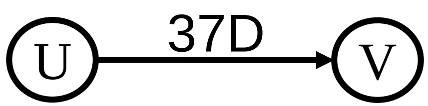
$a = 43$
$b = 5$

$$43 = 8 \times 5 + 3 \Rightarrow a\%b = 43\%5 = 3$$

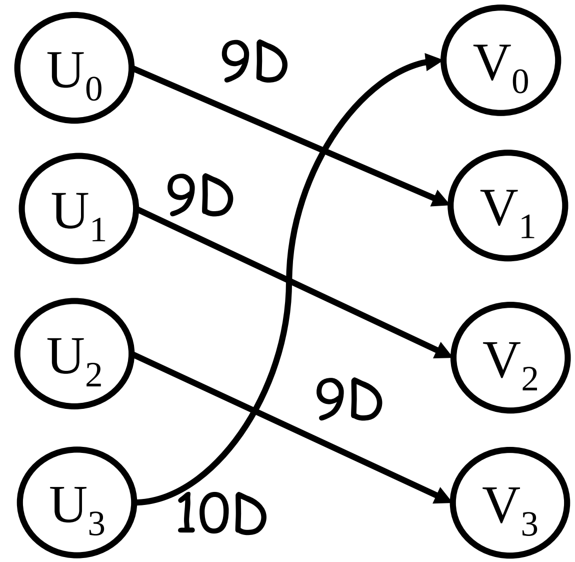
General Algorithm for unfolding

Step 1. **For each node *U* in the original DFG, draw *J* nodes U_0 , U_1 , U_2 ,…, U_{J-1}**

J=4

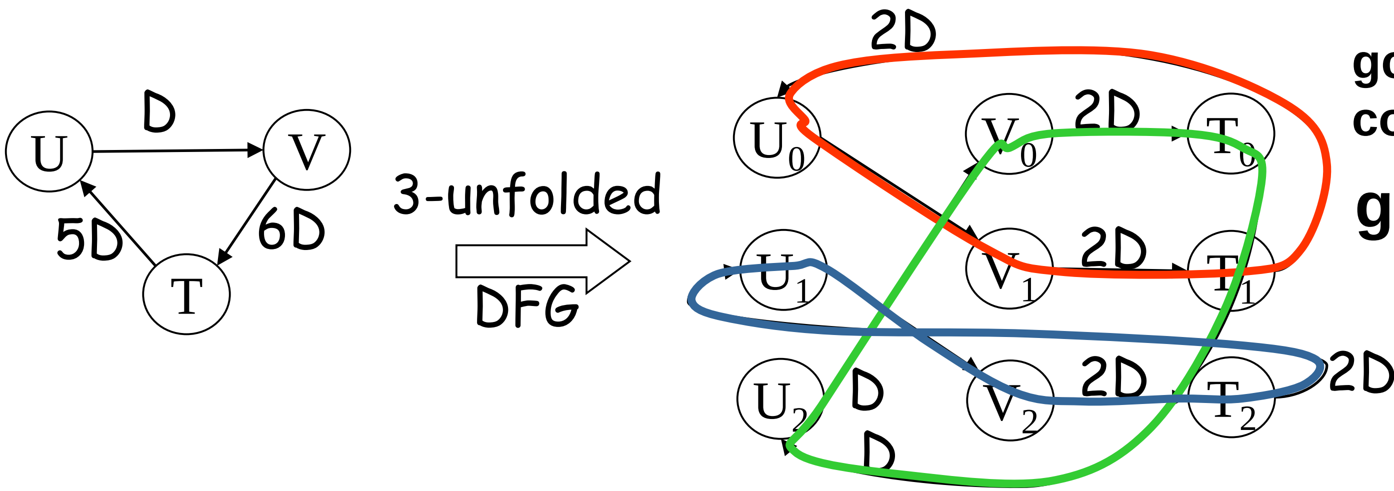


$$\left\lfloor \frac{(i+w)}{J} \right\rfloor = \left\lfloor \frac{(i+37)}{4} \right\rfloor = \begin{cases} 9, i = 0,1,2 \\ 10, i = 3 \end{cases}$$



Step 2. **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, …, J-1**

Properties of unfolding



gcd=greatest common divisor

gcd(12 , 3) =3

- **Unfolding preserves the number of delays in a DFG**

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

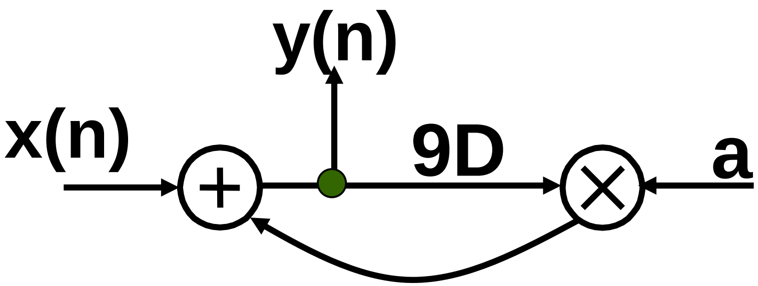
- **Unfolding preserves precedence constraints**

- **J-unfolding of a loop with w_l delays in the original DFG ⇨ gcd(w_l , J) loops in the unfolded DFG. Each loop contains w_l/gcd(w_l , J) delays and J/ gcd(w_l , J) copies of each node.**

- **Unfolding a DFG with iteration bound T_∞ results in a J-unfolded DFG with iteration bound JT_∞ .**

Relation Unfolding and Iteration Bound

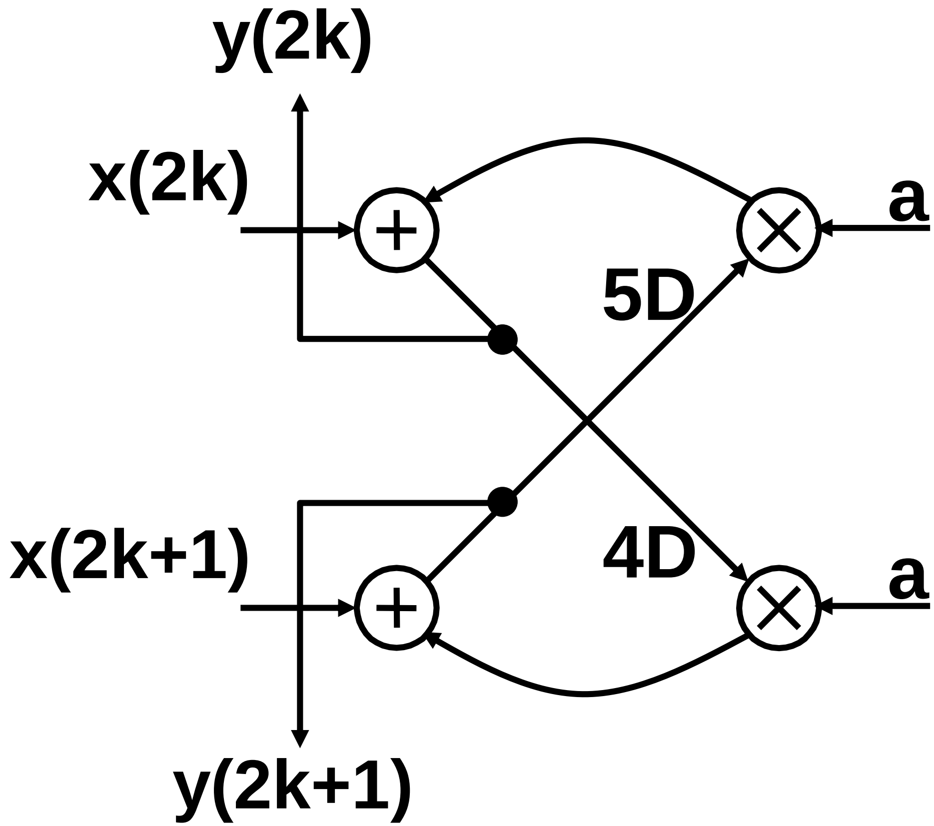
T_A=3, TM=6



$$T_\infty = 9/9 = 1$$

gcd(9 , 2) = 1 ⇨ 1 loop

$$T_\infty = 18/9 = 2$$

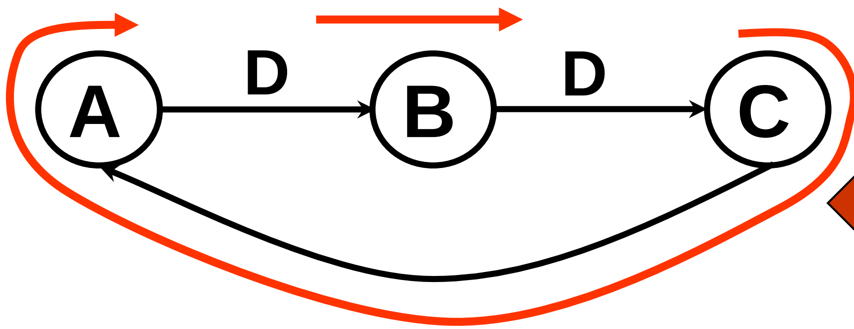


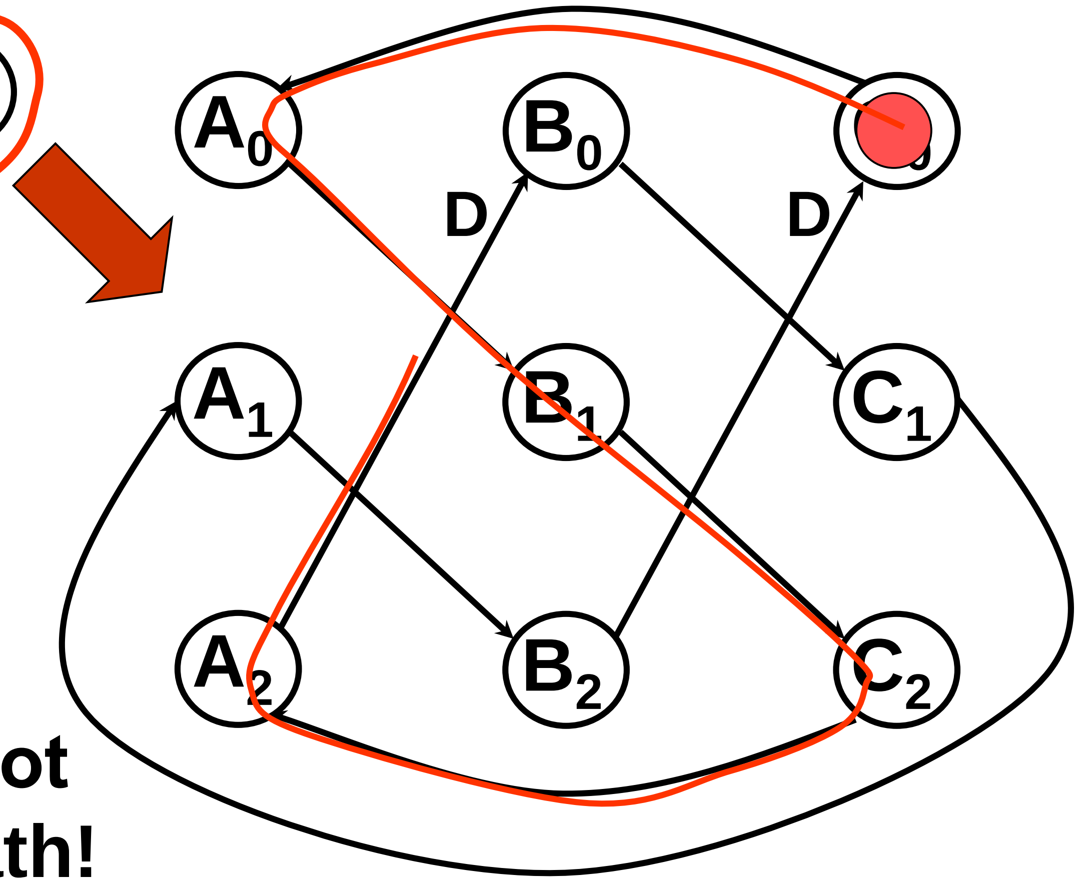
JT_∞

But we process 2 samples

Relation Unfolding and the Critical Path

If edge with w<J ⇨ (J-w) paths with zero delay and w paths with 1 delay





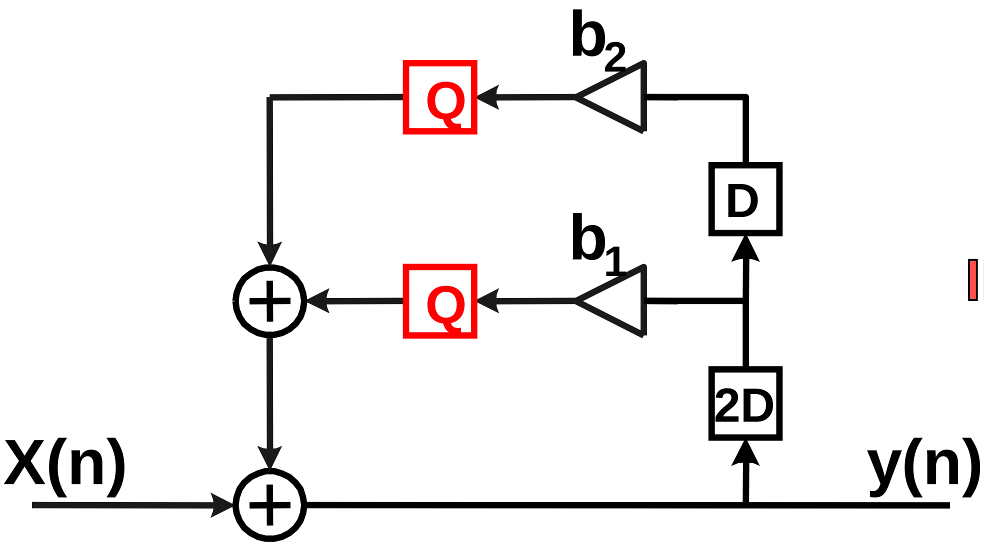
Can lead to increased critical path!

Edge with w≥J will not create new critical path!

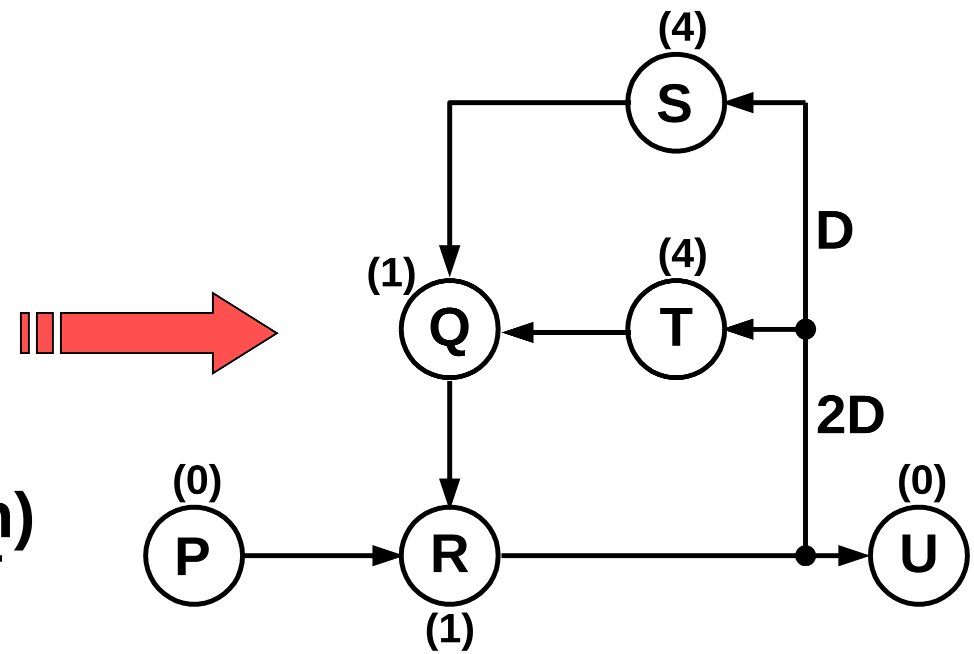
Applications of Unfolding: Sample Period Reduction

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

Sample Period Reduction: case 1

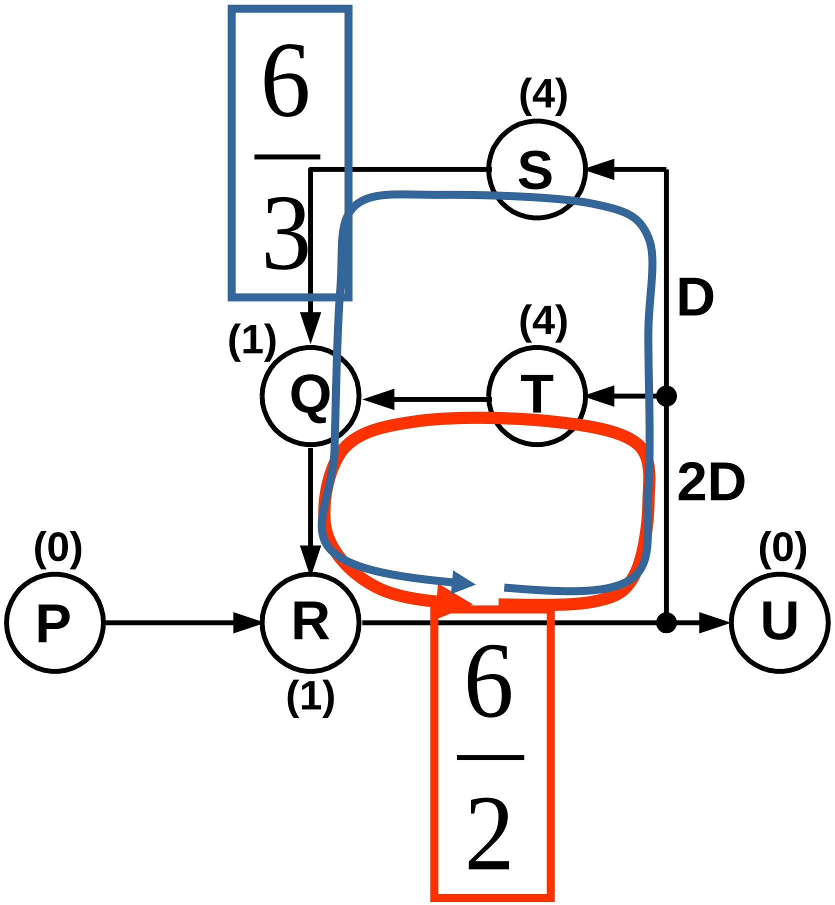


"Close to" IIR-filter



Sample Period Reduction: case 1

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



$$T_\infty = \max_{l \in L} \left\{ \frac{t_l}{w_l} \right\}$$

$$= \max_{l \in L} \left\{ \frac{6}{3}, \frac{6}{2} \right\} = 3$$

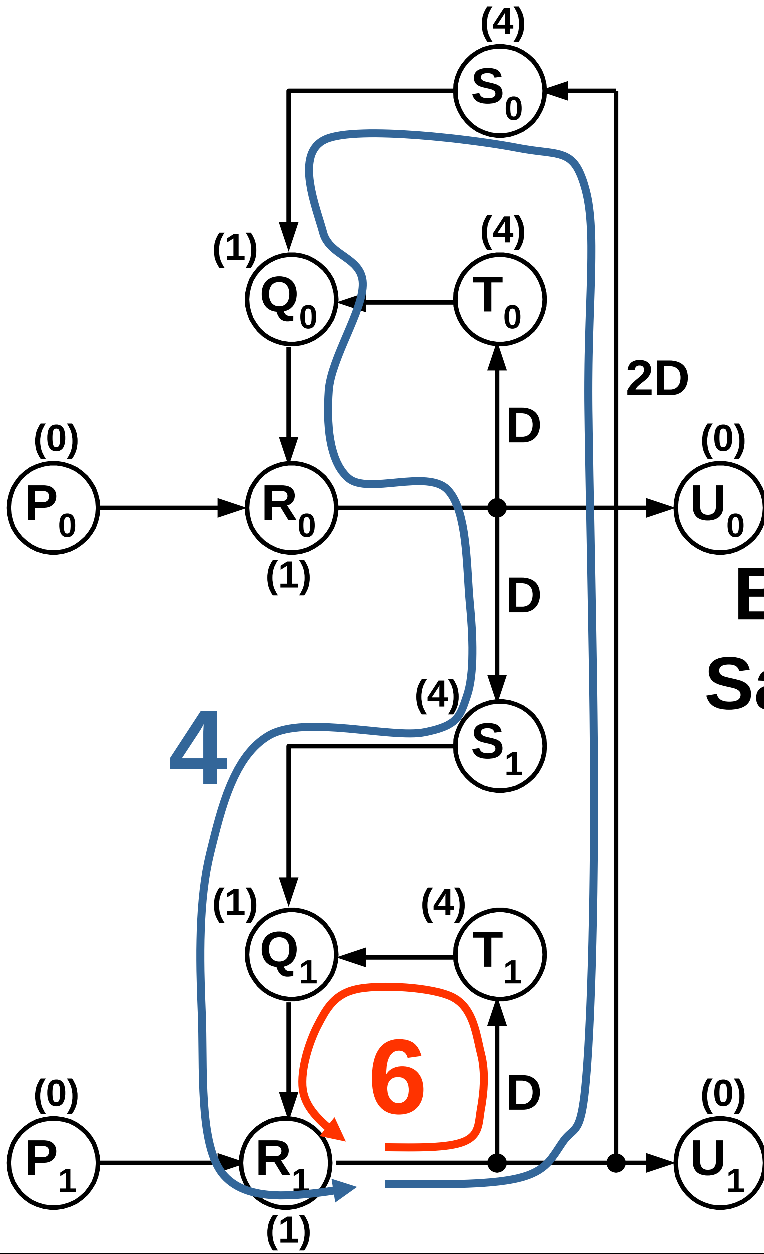
<4, **max node time**

Sample Period Reduction: case 1

If the computation time of a node ‘U’, t_u, is greater than the iteration bound T_∞, then $\lceil t_u/T_\infty \rceil$ - unfolding should be used.

$t_u = 4$ and $T_\infty = 3$

$\lceil 4/3 \rceil = 2$ - unfolding

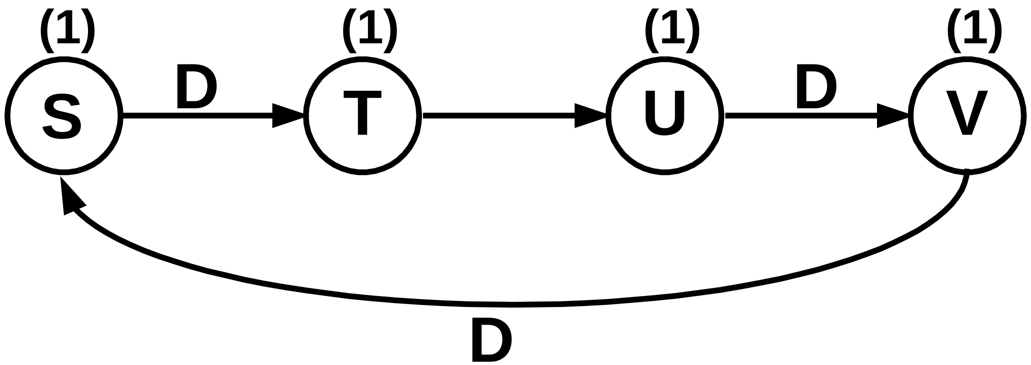


But two Samples!

3

Sample Period Reduction: case 2

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



$$T_\infty = \max_{l \in L} \left\{ \frac{t_l}{w_l} \right\} = \frac{4}{3}$$

If a critical loop bound is of the form t_l/w_l where t_l and w_l are mutually co-prime, then w_l-unfolding should be used.

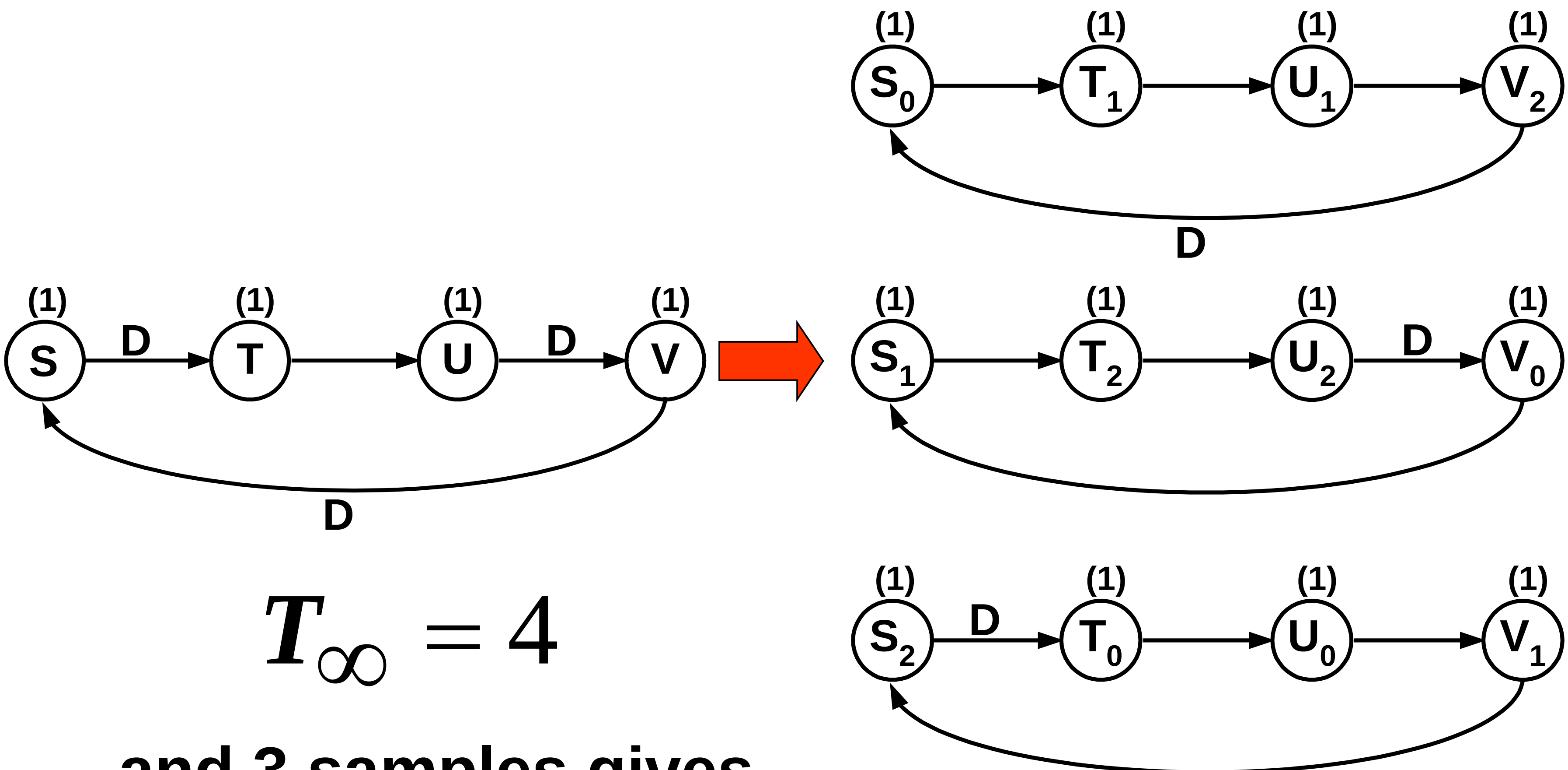
Unfolding of 3

Mutally Co-Prime

- Two integers *a* and *b* are co-prime if the only positive integer that divides both of them is 1.

- For example, the integers 6, 10, 15
are coprime because 1 is the only positive integer that divides all of them.

Sample Period Reduction: case 2 (2)



$$T_\infty = 4$$

and 3 samples gives minimum sample period 4/3

Sample Period Reduction: case 3

The original DFG cannot have sample period equal to the iteration bound because the longest node computation is larger than the iteration bound T_∞, and T_∞ is not an integer

The minimum J that achieves the iteration bound is the minimun value of J such that JT_∞ is an integer and is greater or equal to the longest node computation time

Sample Period Reduction: case 3

Basically case 3 = case I + case II

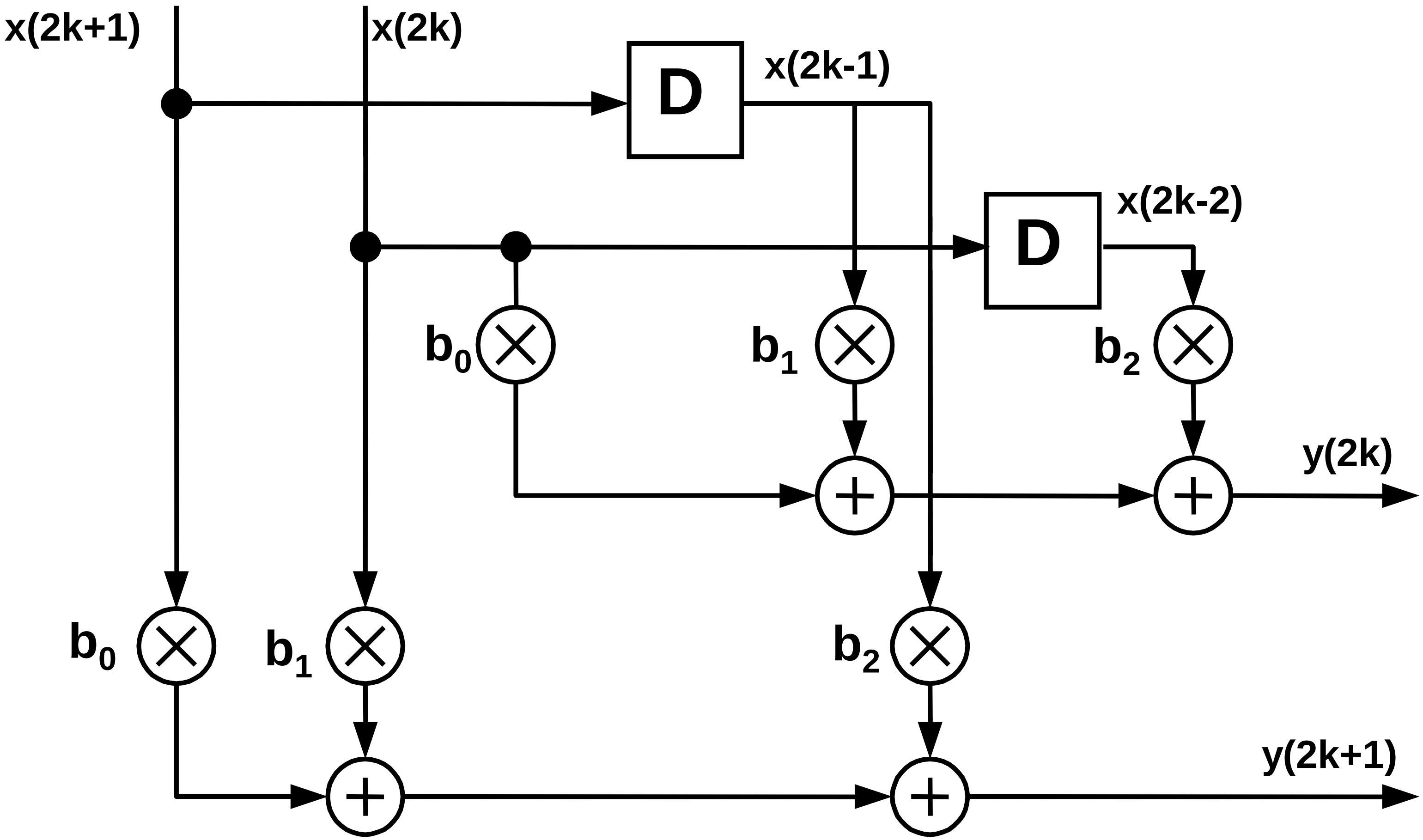
The minimum J that achieves the iteration bound is the minimun value of J such that JT_{∞} is an integer and is greater or equal to the longest node computation time.

Ex: Assume $\mathbf{T_{\infty}=4/3}$ and $\mathbf{t_{U,max} = 6}$

If $J \cdot T_{\infty} \geq t_{U,\max}$ then $J \cdot \frac{4}{3} \geq 6 \Rightarrow J = 6$

Parallel Processing and Unfolding

Parallel processing can be performed by unfolding (chapter 3)



Parallel Processing Techniques

Word-level Parallel Processing

- Unfolding a word-serial architecture by J creates a word-parallel architecture that processes J words per clock cycle

Bit-level Parallel Processing

Bit-serial processing

- One bit is processed per clock cycle and a complete word is processed in W clock cycles, where W is the word-length.

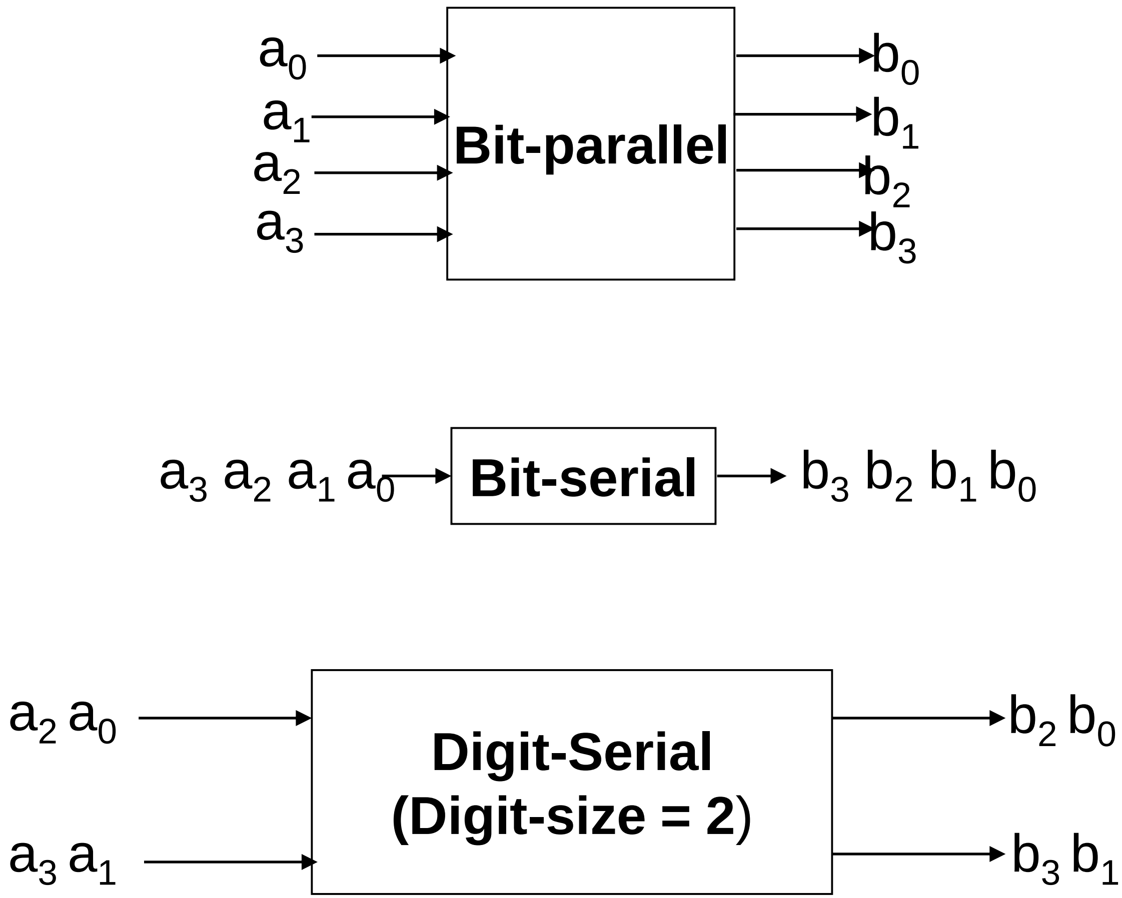
Bit-parallel processing

- One word of W bits is processed every clock cycle

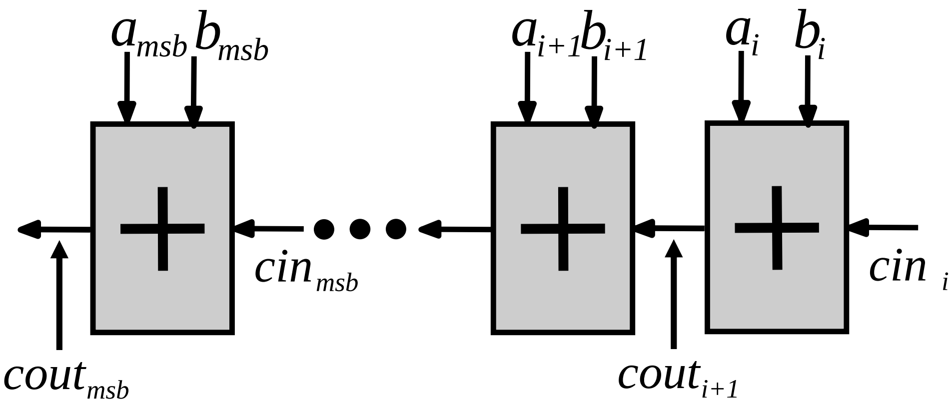
Digit-serial processing

- N bits are processed per clock cycle and a word is processed in W/N clock cycles, where N is referred to as the digit size

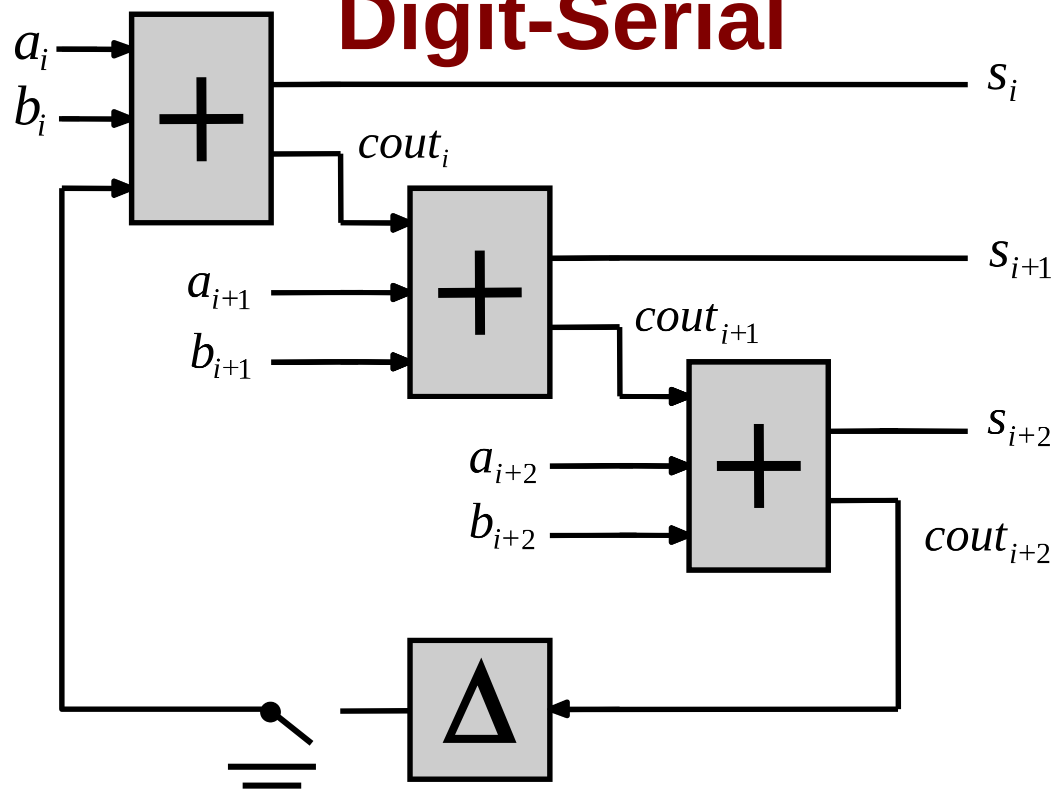
Bit-Level Parallel Processing



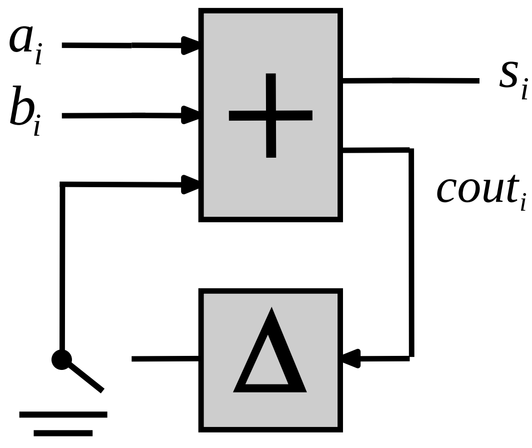
Bit-Parallel



Digit-Serial

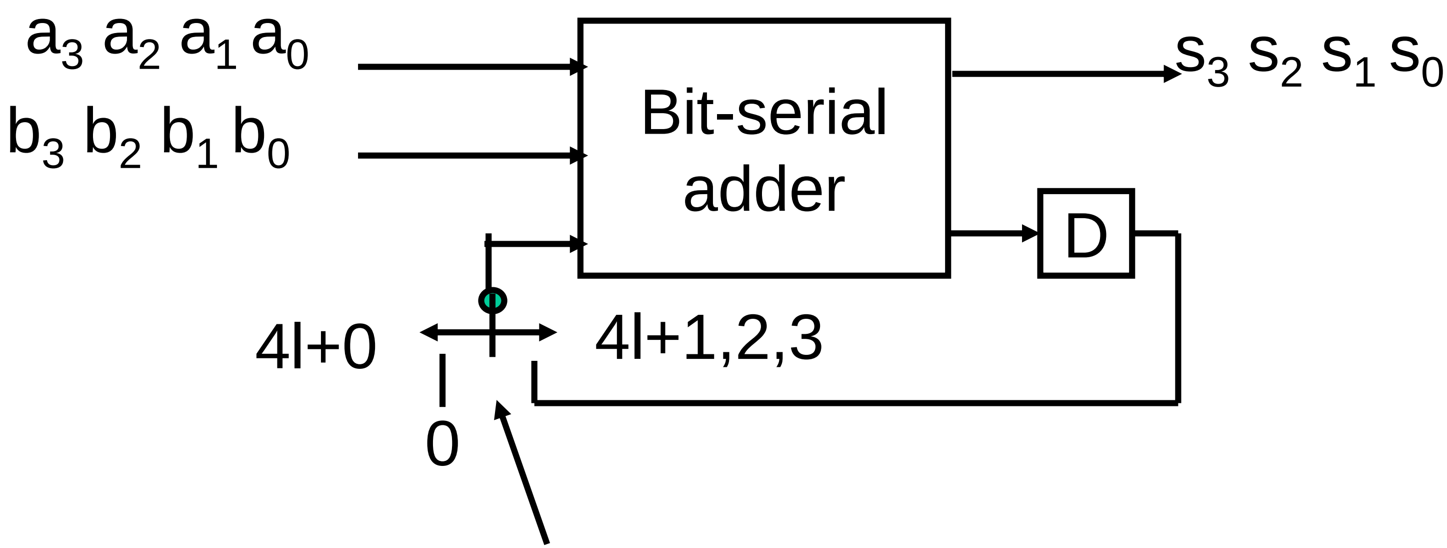


Bit-Serial



Bit-serial adder

Bit-serial can be seen as a time-multiplexed architecture, in this example on addition (i.e. 1 iteration) takes 4cc.

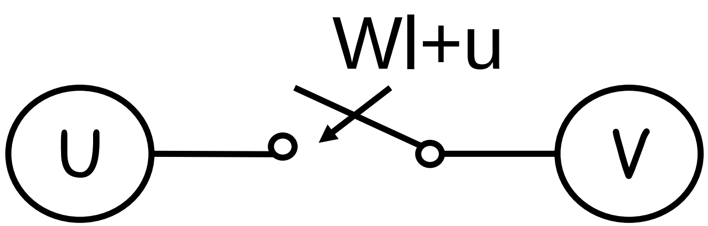


Switch for carry signal (Wl+u)

How to unfold switches?

Unfolding of Switches

- **The following assumptions are made when unfolding an edge U→V containing a switch :**
  - **The wordlength W is a multiple of the unfolding factor J, i.e. W = W'J.**
  - **All edges into and out of the switch have no delays.**



Unfolding of Switches

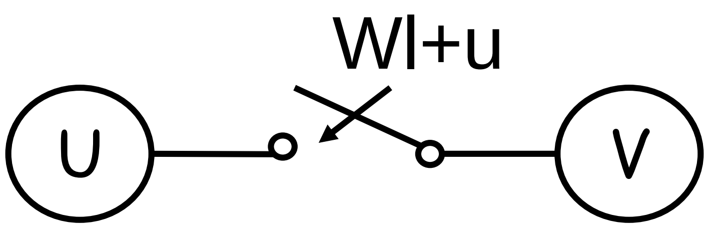
- **The following assumptions are made when unfolding an edge U→V containing a switch :**
  - **The wordlength W is a multiple of the unfolding factor J, i.e. W = W'J.**
  - **All edges into and out of the switch have no delays.**

- **If so, an edge U→V can be unfolded as:**
  - **Write the switching instance as**

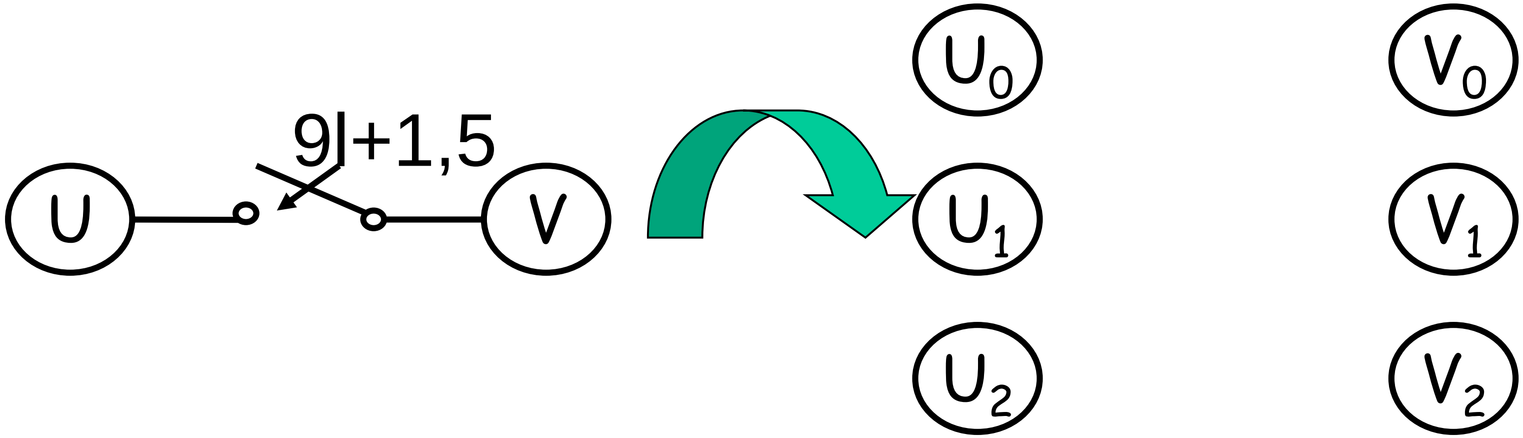
$$Wl + u = J(W'l + \lfloor u/J \rfloor) + (u\%J)$$

  - **Draw an edge from the node $U_{u\%J} \Rightarrow V_{u\%J}$,**

    which is switched at time instance $(W'l + \lfloor u/J \rfloor)$.



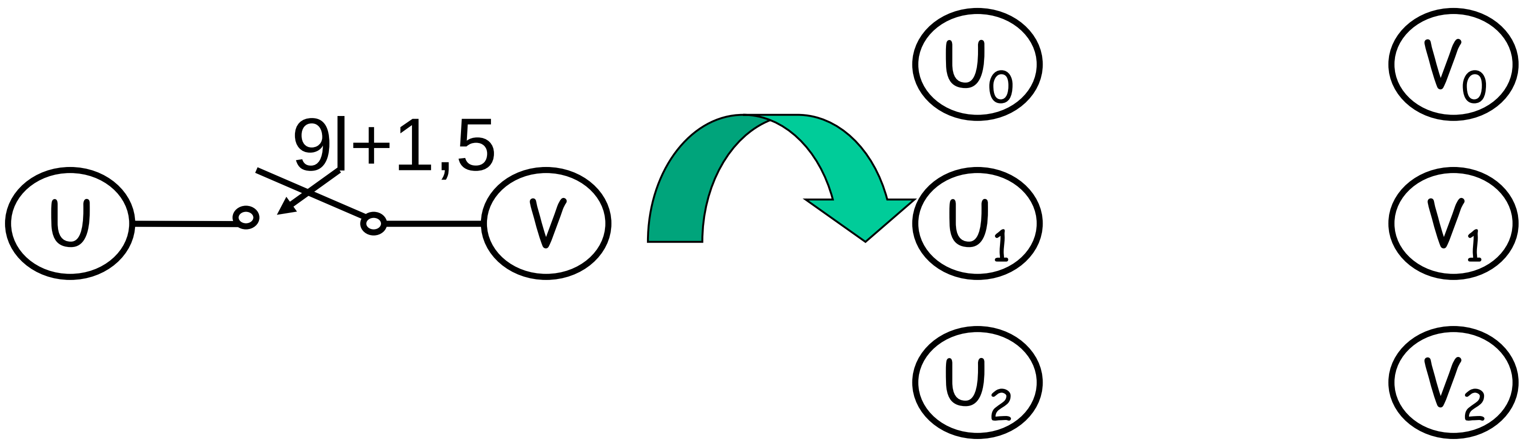
Example: Unfolding of Switches, J=3



- Write the switching instance as

$$Wl + u = J(\, W'l + \lfloor u/J \rfloor \,) + (u\%J)$$

Example: Unfolding of Switches, J=3



- **Write the switching instance as**

$$Wl + u = J(\,W'l + \lfloor u/J \rfloor\,) + (u\%J)$$

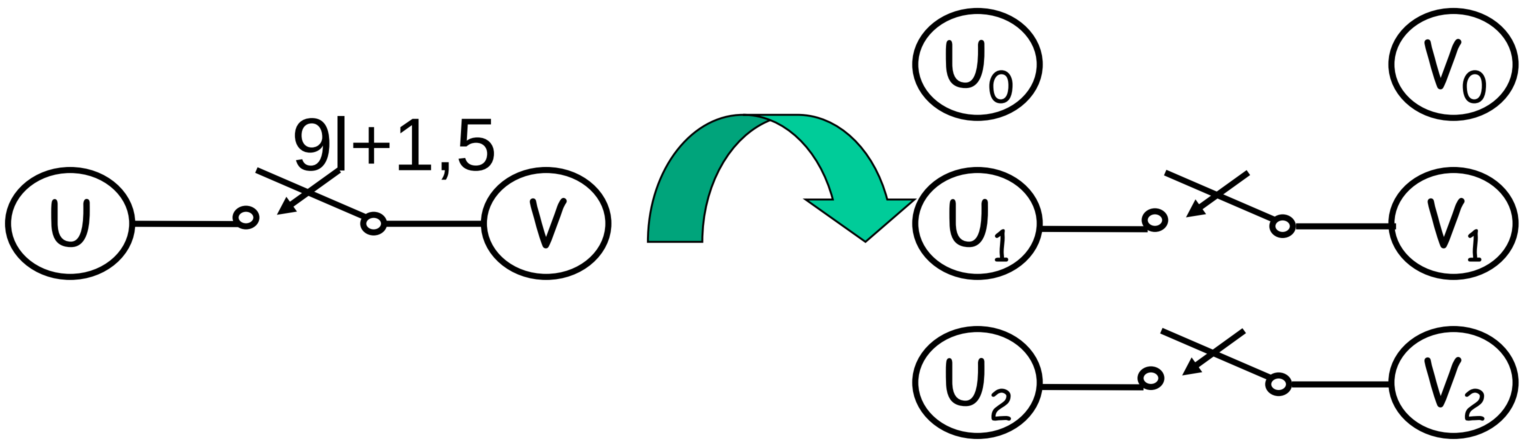
$$9l+1=3(3l + \lfloor 1/3 \rfloor) + (1\%3) = 3(3l + 0) + 1$$

$$9l+5=3(3l + \lfloor 5/3 \rfloor) + (5\%3) = 3(3l + 1) + 2$$

Edges between Nodes

Switched at time instances

Example: Unfolding of Switches, J=3



- **Write the switching instance as**

$$Wl + u = J(\, W'l + \lfloor u/J \rfloor\,) + (u\%J)$$

$$9l+1=3(3l + \lfloor 1/3 \rfloor) + (1\%3) = 3(3l + 0) + 1$$

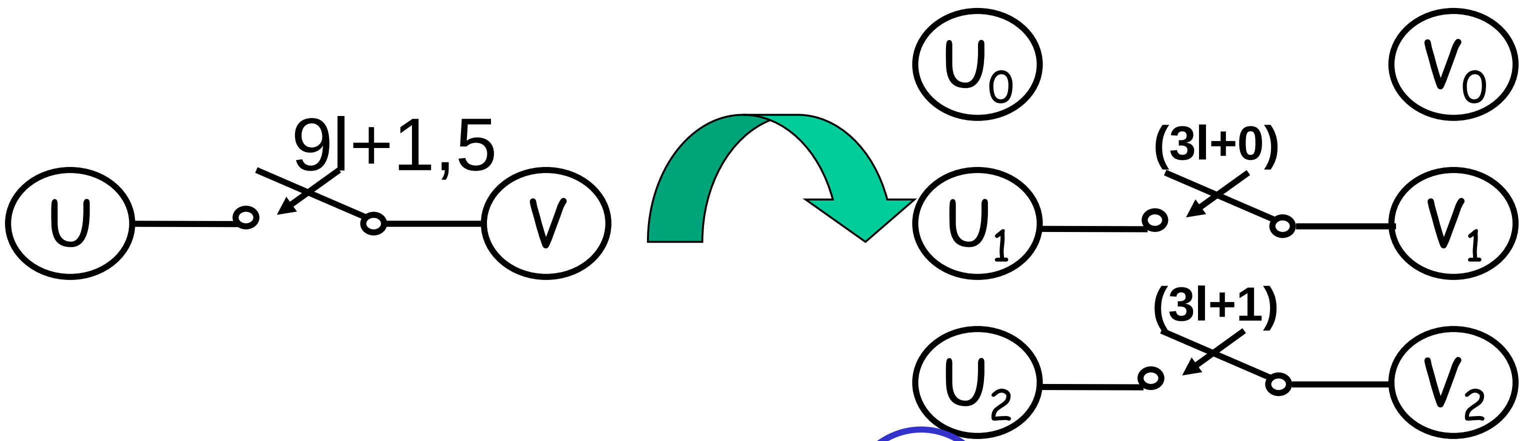
$$9l+5=3(3l + \lfloor 5/3 \rfloor) + (5\%3) = 3(3l + 1) + 2$$

Edges between Nodes

- **Draw an edge from the node** $U_{u\%J} \Rightarrow V_{u\%J'}$ **I.e.**

$U_1 \Rightarrow V_1$ and $U_2 \Rightarrow V_2$

Example: Unfolding of Switches, J=3



$9l+1=3(3l + \lfloor 1/3 \rfloor) + (1\%3) = 3(3l + 0) + 1$

$9l+1=3(3l + \lfloor 5/3 \rfloor) + (5\%3) = 3(3l + 1) + 2$

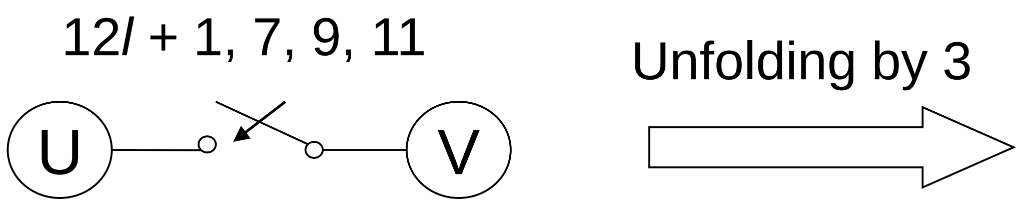
Switched at time instances

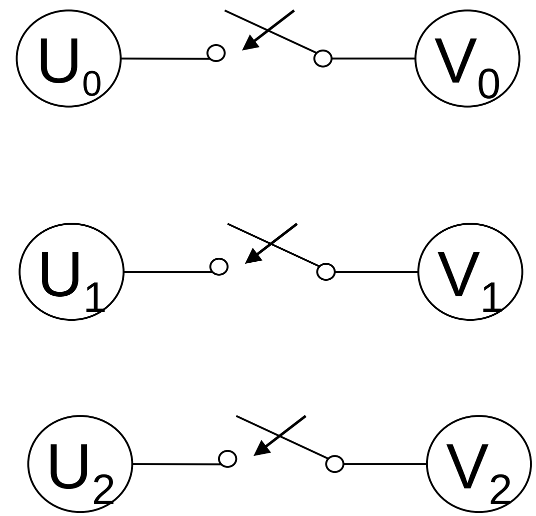
switched at time instance $(W'l + \lfloor u/J \rfloor)$, I.e.

$U_1 \Rightarrow V_1$ at (3l+0) and $U_2 \Rightarrow V_2$ at (3l+1)

Switch with multiple instances

Example :





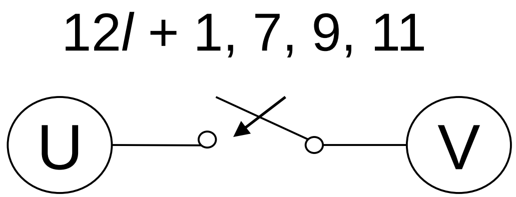
$$\mathbf{Wl + u = J(\,W'l + \lfloor u/J \rfloor\,) + (u\%J)}$$

To unfold the DFG by J=3, the switching instances are as follows

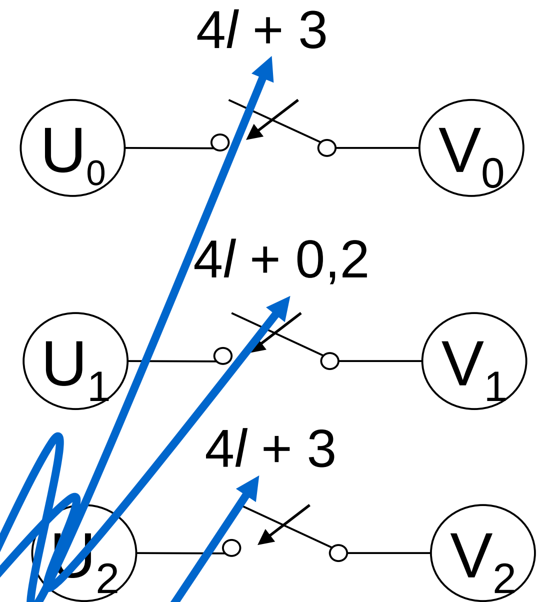
$$12l + 1 = 3(4l + 0) + 1$$
$$12l + 7 = 3(4l + 2) + 1$$
$$12l + 9 = 3(4l + 3) + 0$$
$$12l + 11 = 3(4l + 3) + 2$$

Switch with multiple instances

Example :



Unfolding by 3



$$Wl + u = J(W'l + \lfloor u/J \rfloor) + (u\%J)$$

Switched at time instances

$12l + 1 = 3(4l + 0) + 1$

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

$12l + 9 = 3(4l + 3) + 0$

$12l + 11 = 3(4l + 3) + 2$

End of Lecture