DSP Design – Lecture 7

Unfolding cont. & Folding

Dr. Fredrik Edman

fredrik.edman@eit.lth.se

Repetition

Unfolding of Switches

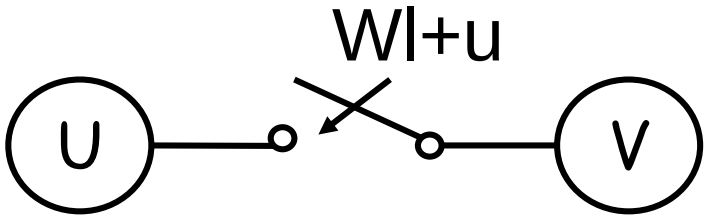
- The following assumptions are made when unfolding an edge U→V containing a switch :
  - The wordlength W is a multiple of the unfolding factor J, i.e. W = W'J.
  - All edges into and out of the switch have no delays.

- 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 $U_{u\%J}$ ⇨ $V_{u\%J}$,

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

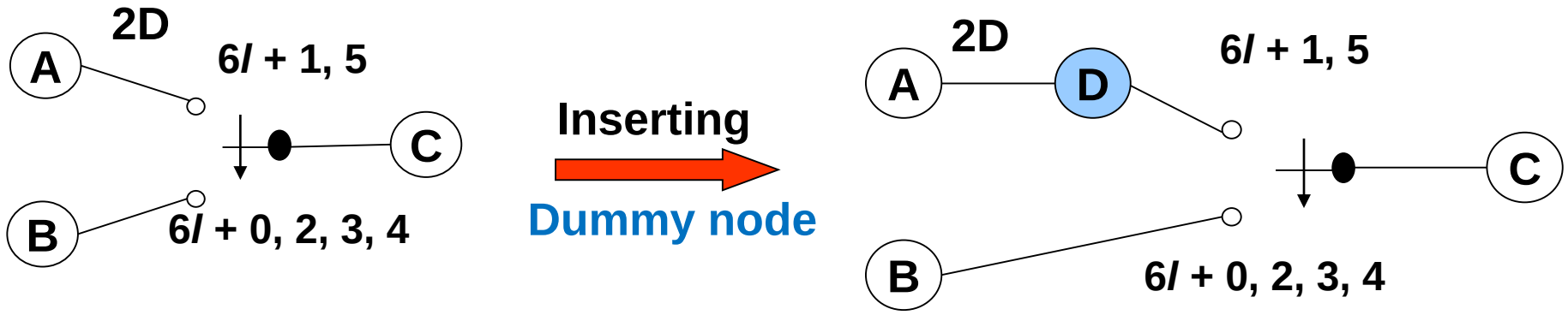


Unfolding (cont.)

Chapter 5

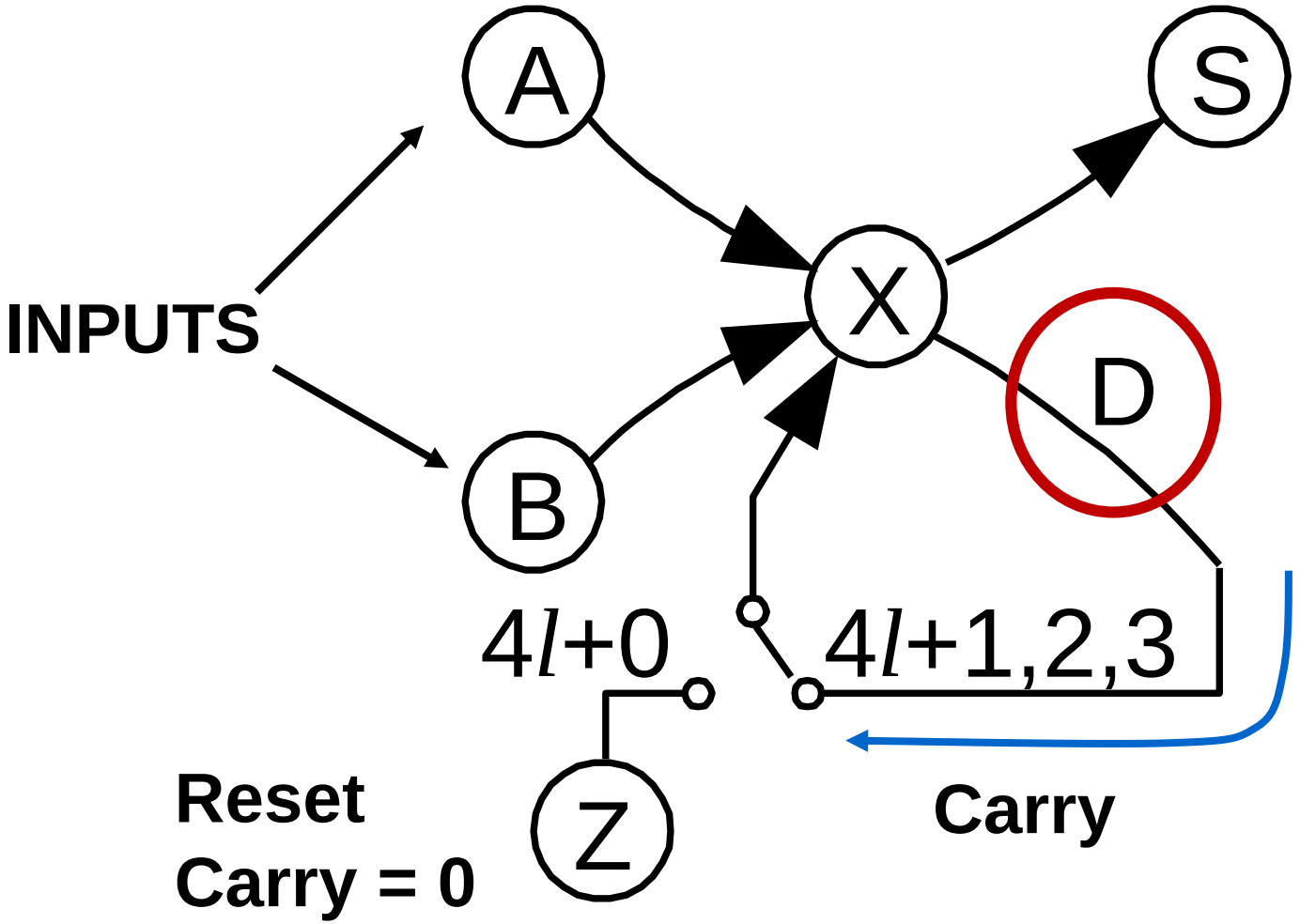
What about Switches with Delays?

Unfolding a DFG containing an edge having a switch and a positive number of delays is done by introducing a dummy node.

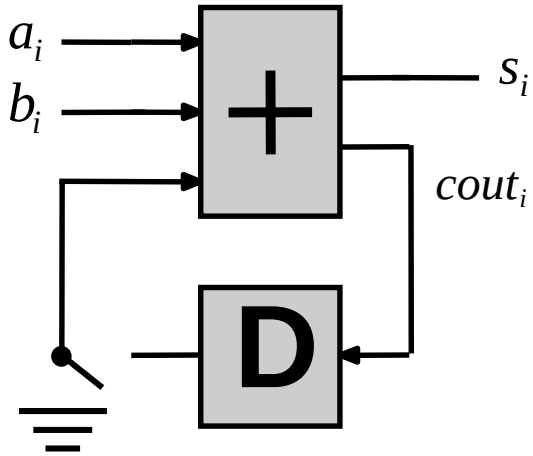


After unfolding remove the dummy node!

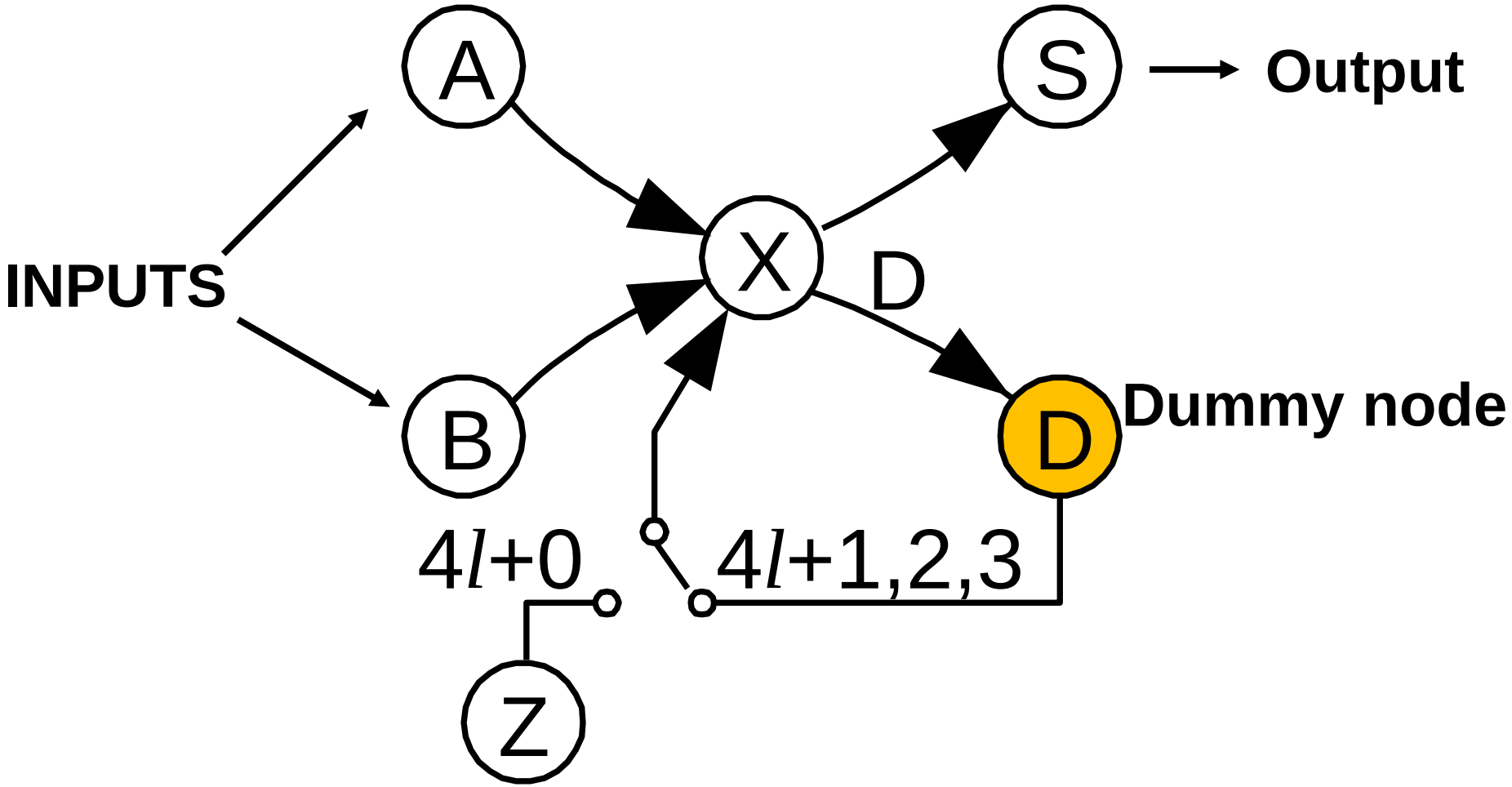
Example: How to Unfold a Bit-serial Adder



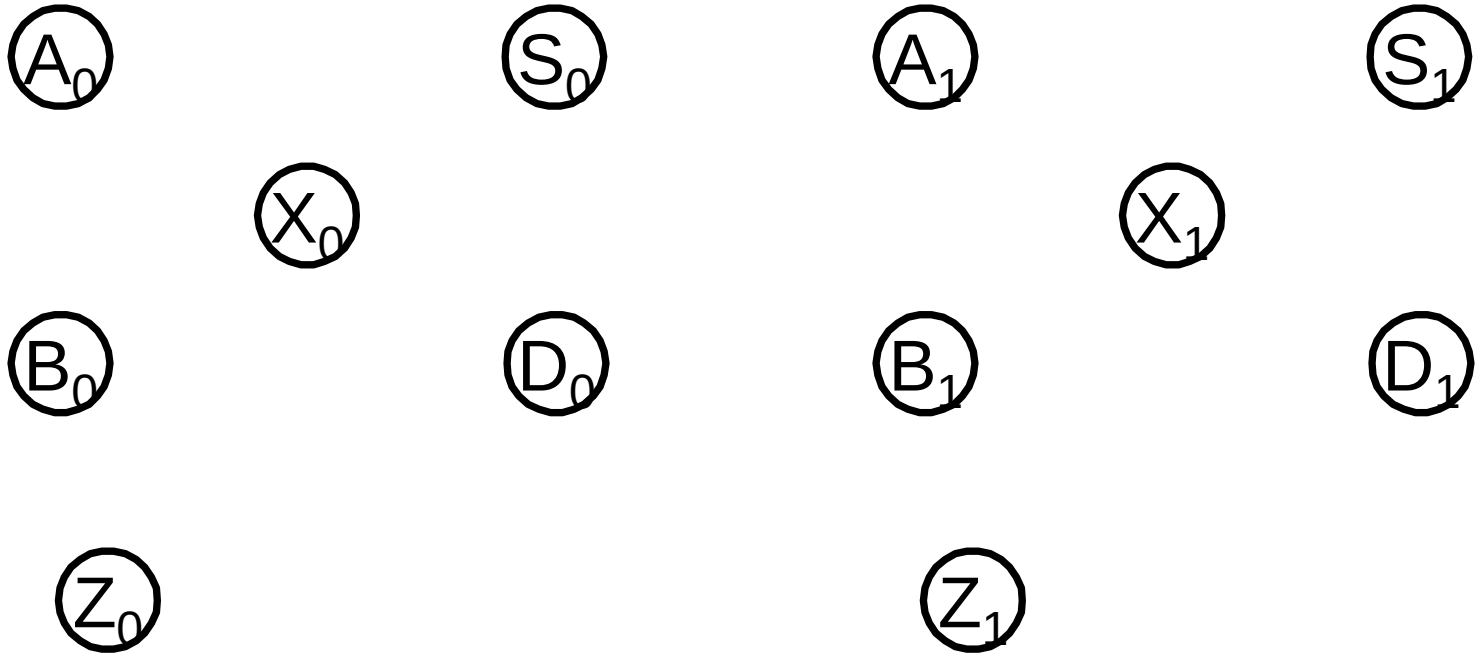
Bit-Serial



Example: How to Unfold a Bit-serial Adder

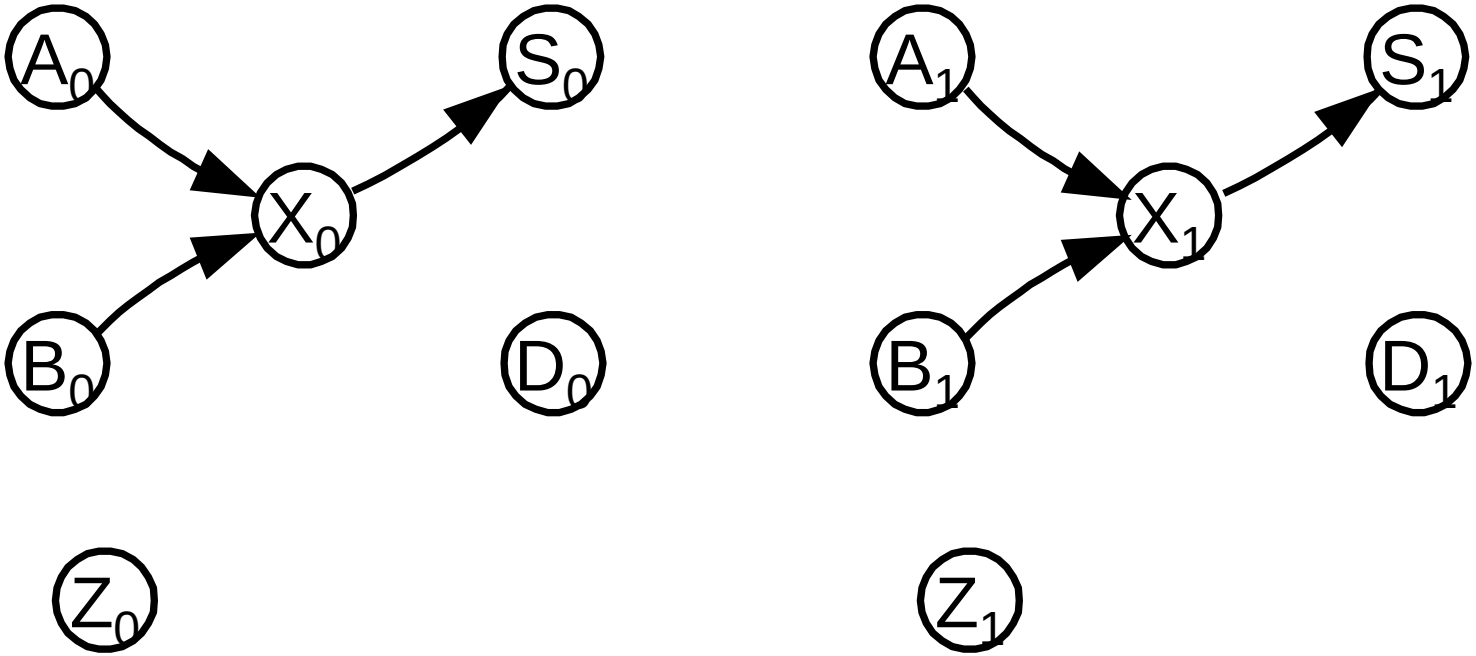


Unfold Bit-serial Adder, J=2



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

Unfold Bit-serial Adder, J=2

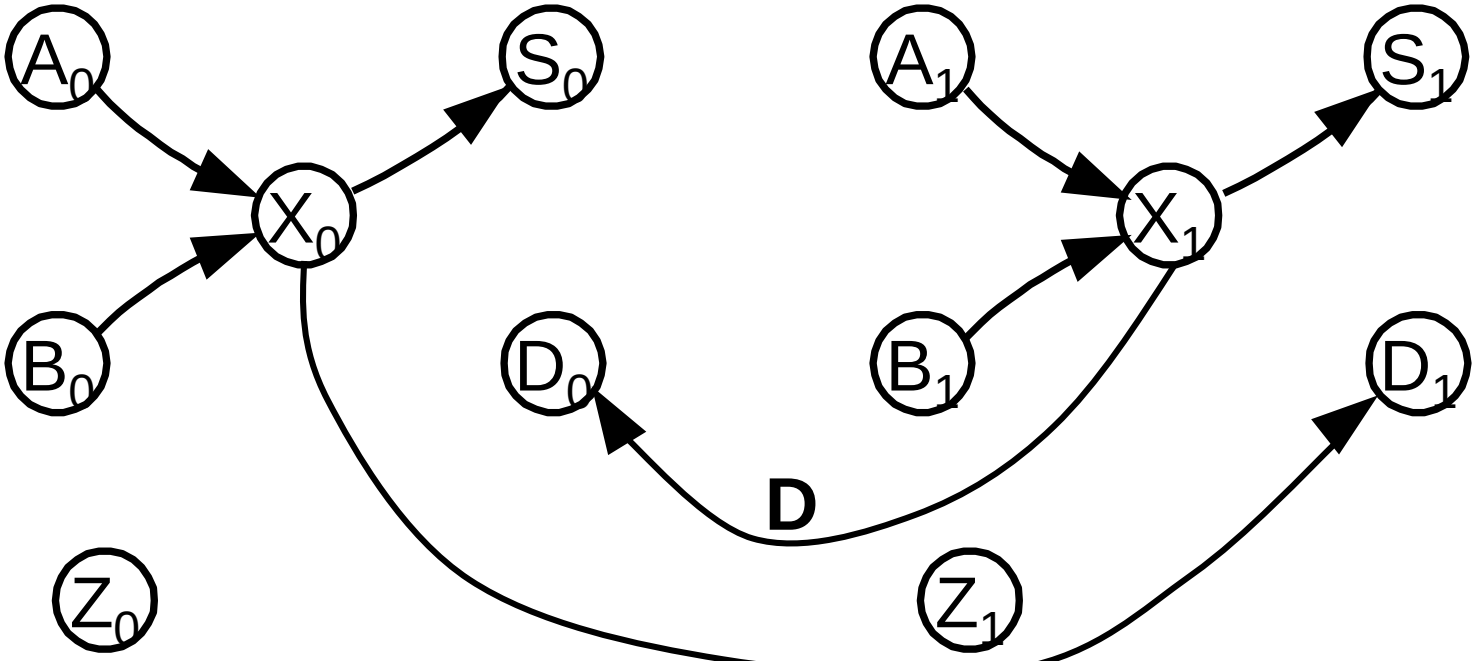


For each edge $U \rightarrow V$ with w delays in the original DFG,

draw the J edges $U_i \rightarrow V_{(i+w)\%J}$ with $\lfloor (i+w)/J \rfloor$ delays for $i = 0, 1, \ldots, J-1$

If edge has w=0 ⇨ $U_i \rightarrow V_i$ **with 0 delays**

Unfold Bit-serial Adder, J=2

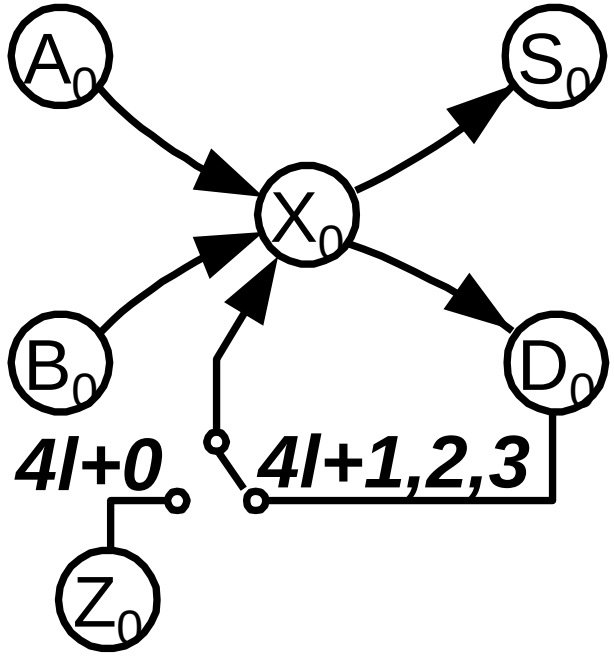


For each edge $U \rightarrow V$ with w delays in the original DFG,

draw the J edges $U_i \rightarrow V_{(i+w)\%J}$ with $\lfloor (i+w)/J \rfloor$ delays for $i = 0, 1, \ldots, J-1$

X⇨D for i=0 ⇨ $X_0 \rightarrow D_1$ with 0 delays and X⇨D for i=1 ⇨ $X_1 \rightarrow D_0$ with 1 delays

Unfold the Switch, J=2



Z ⇨ X

4l+0 → 2(2l+0)+0

D ⇨ X

4l+1 → 2(2l+0)+1

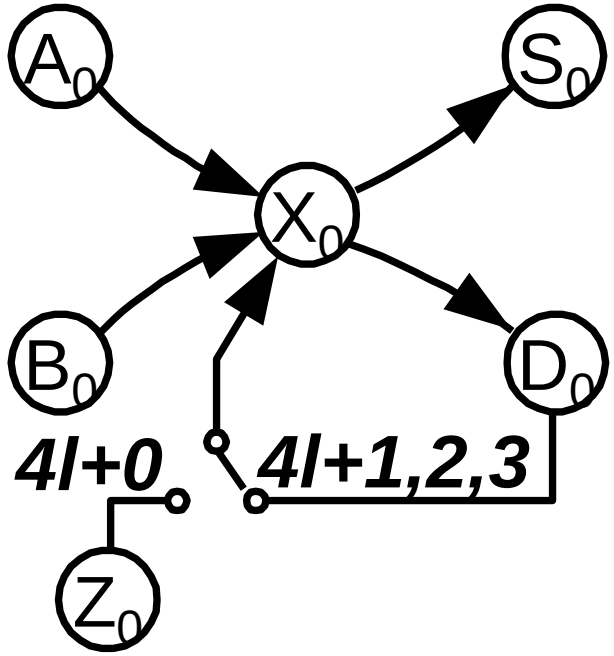
4l+2 → 2(2l+1)+0

4l+3 → 2(2l+1)+1

Write the switching instance as

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

Unfold the Switch, J=2



Z ⇨ X

4l+0 → 2(2l+0)+0

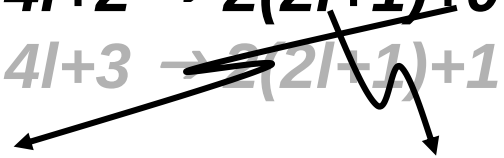
$Z_0 \rightarrow X_0$ at time *2l+0*

D ⇨ X

4l+1 → 2(2l+0)+1

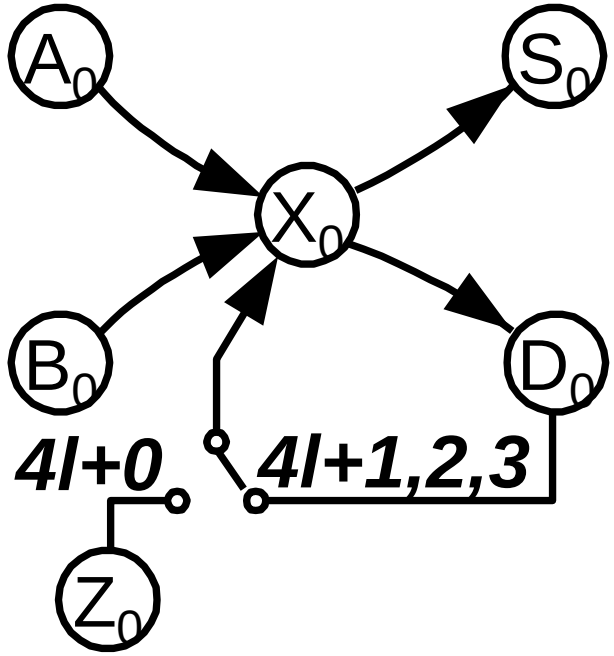
4l+2 → 2(2l+1)+0

4l+3 → 2(2l+1)+1



$D_0 \rightarrow X_0$ at time *2l+1*

Unfold the Switch, J=2



Z ⇨ X

4l+0 → 2(2l+0)+0

D ⇨ X

4l+1 → 2(2l+0)+1

4l+2 → 2(2l+1)+0

4l+3 → 2(2l+1)+1

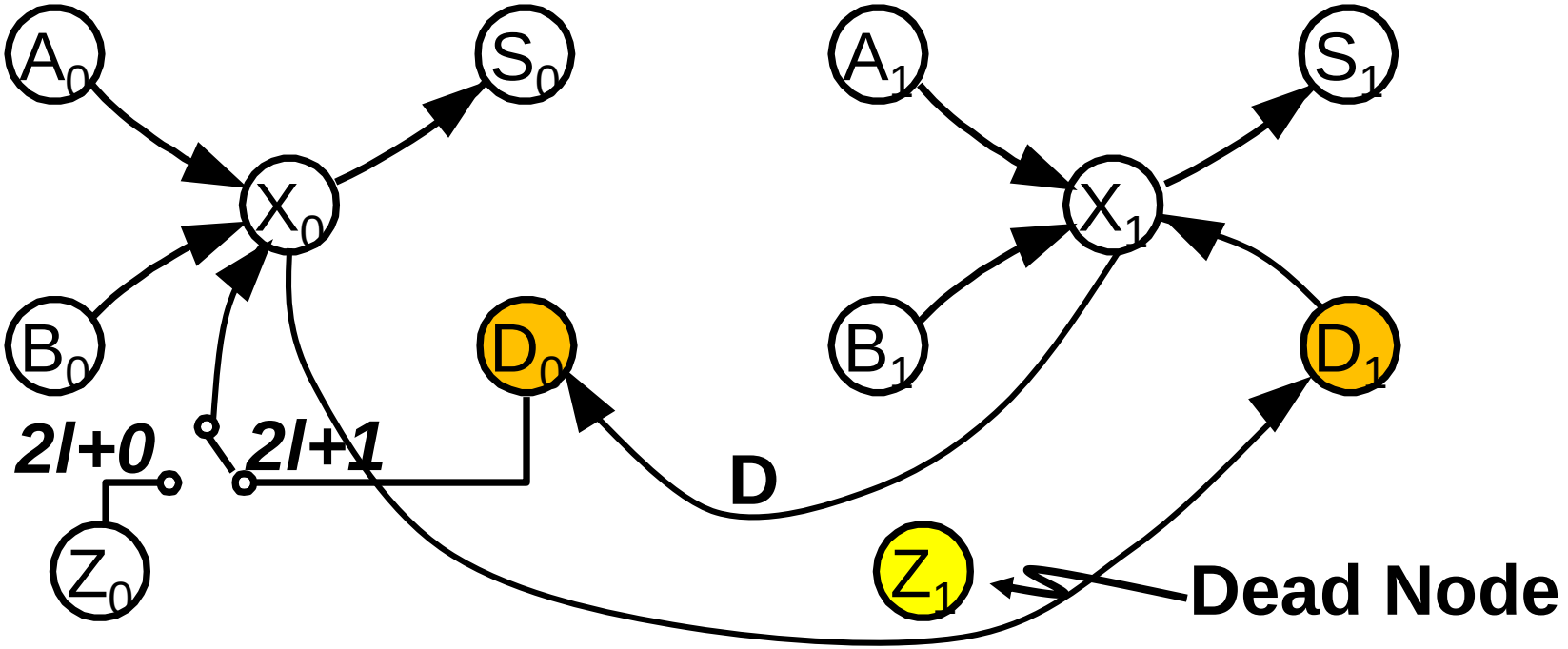
$Z_0 \rightarrow X_0$ at time *2l+0*

$D_0 \rightarrow X_0$ at time *2l+1*

$D_1 \rightarrow X_1$ at time *2l+0,1*

i.e. always closed

Unfold the Switch, J=2

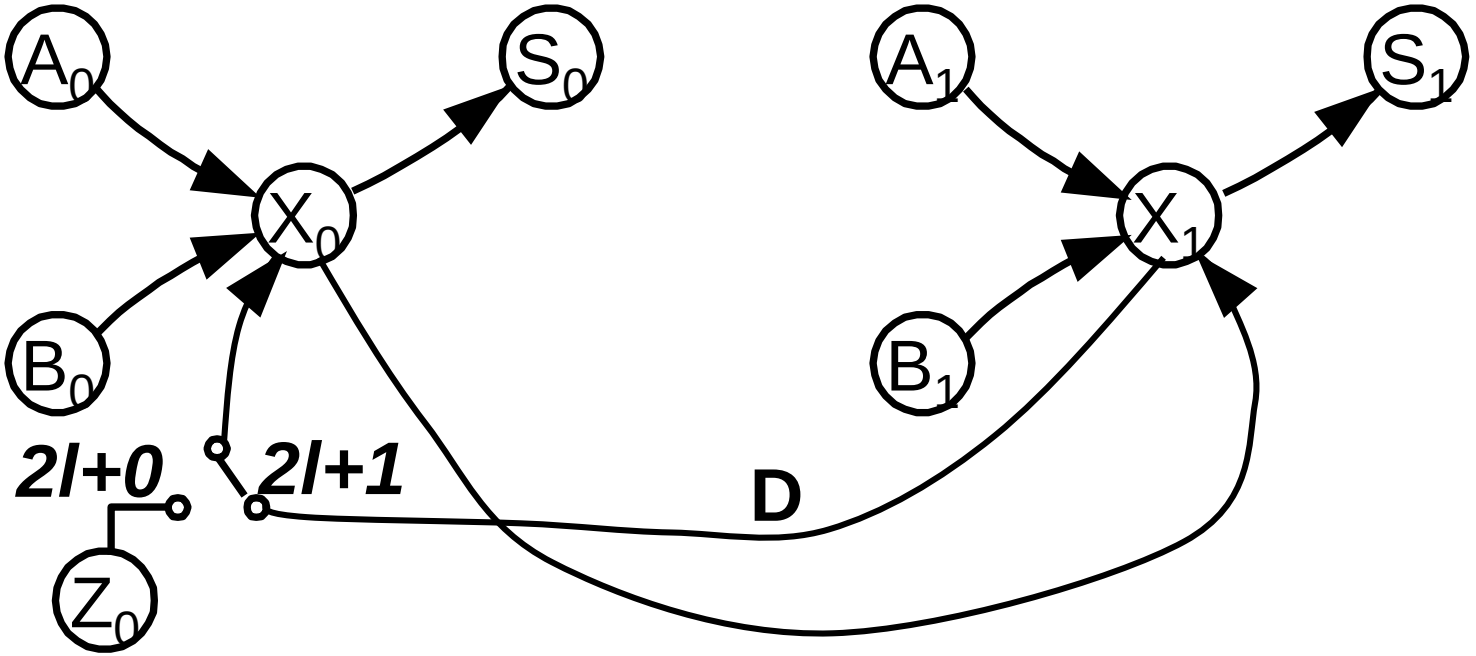


$Z_0 \rightarrow X_0$ at time *2l+0*

$D_0 \rightarrow X_0$ at time *2l+1*

$D_1 \rightarrow X_1$ at time *2l+0,1*
i.e. always closed

Remove Dead and Dummy Nodes

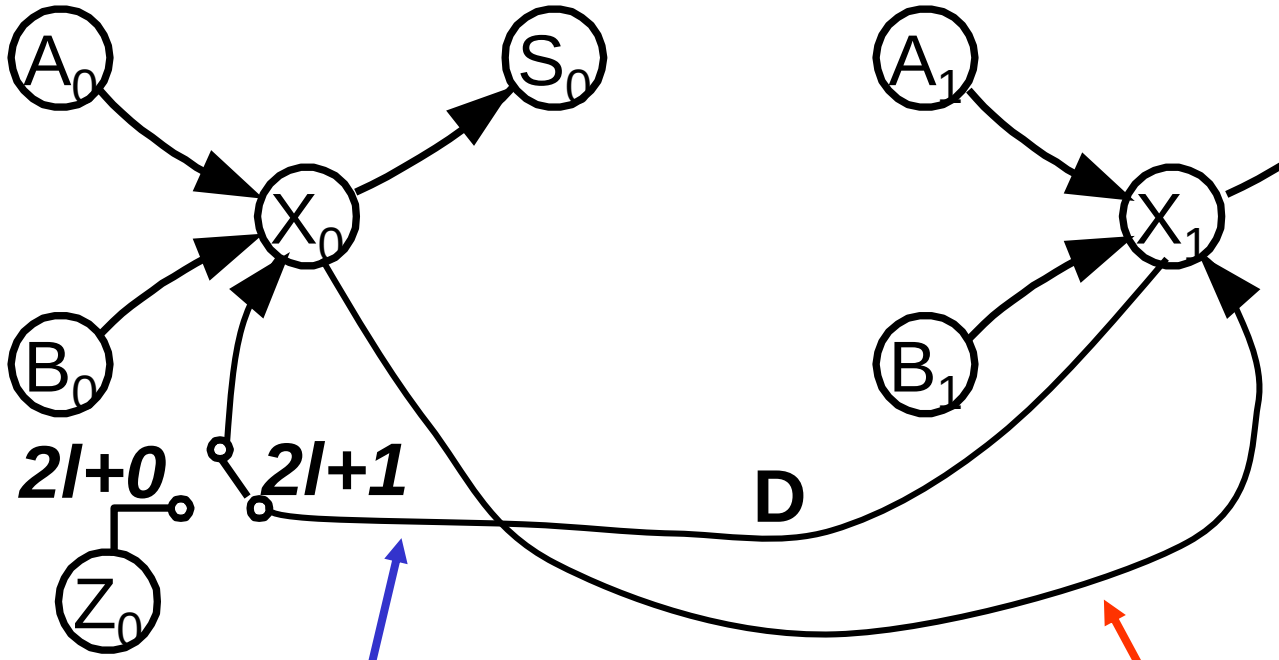


$Z_0 \rightarrow X_0$ at time *2l+0*

$D_0 \rightarrow X_0$ at time *2l+1*

$D_1 \rightarrow X_1$ at time *2l+0,1*
i.e. always closed

The Digit Serial Adder

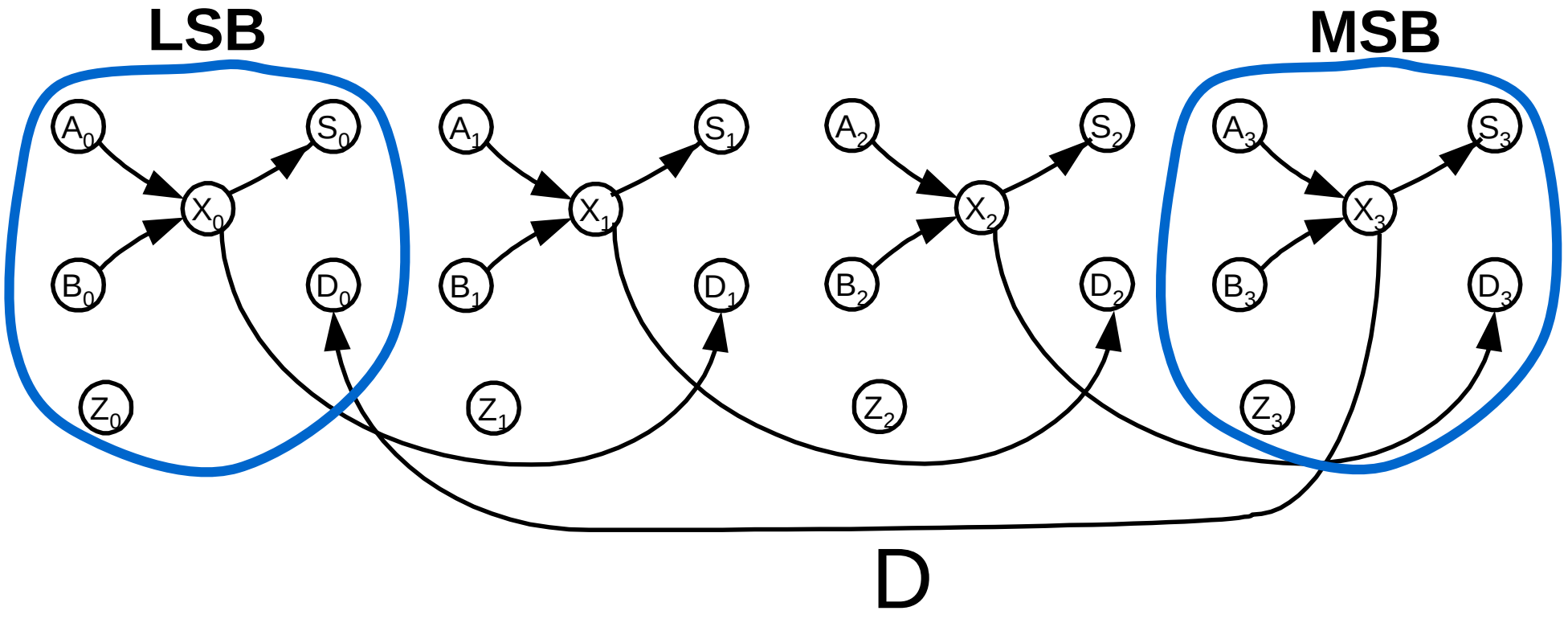


digit2	digit1
a_3 a_2	a_1 a_0
b_3 b_2	b_1 b_0
s_3 s_2	s_1 s_0

Carry next iteration
D=1

Carry within iteration

Fully Parallel Adder, i.e.J=4

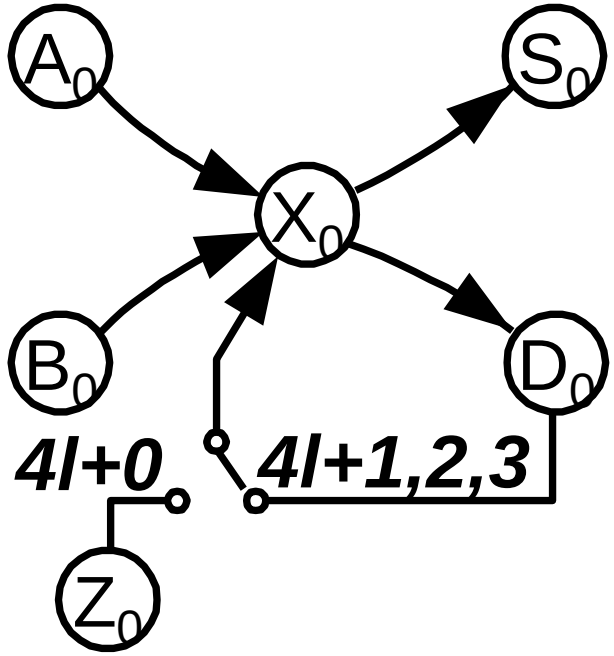


For each node U in the original DFG, draw J nodes U_0 , U_1 , U_2 ,…, U_{J-1}

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, …, J-1

Unfold the Switch, J=4



Z ⇨ X

4l+0 → 4(1l+0)+0

D ⇨ X

4l+1 → 4(1l+0)+1

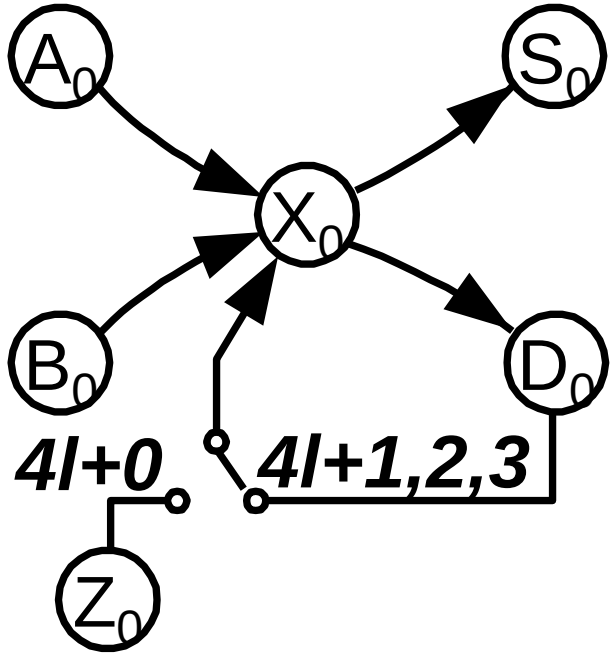
4l+2 → 4(1l+0)+2

4l+3 → 4(1l+0)+3

Write the switching instance as

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

Unfold the Switch, J=4



Z ⇨ X

$4l+0 \rightarrow 4(1l+0)+0$

D ⇨ X

$4l+1 \rightarrow 4(1l+0)+1$

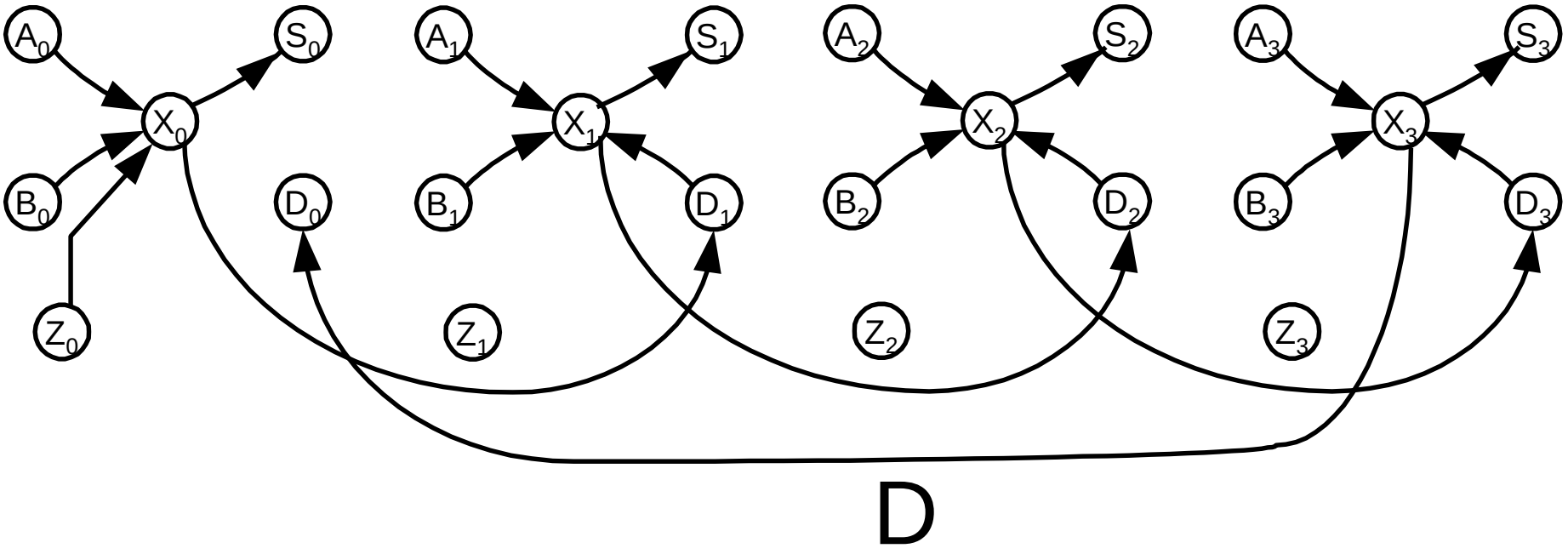
$4l+2 \rightarrow 4(1l+0)+2$

$4l+3 \rightarrow 4(1l+0)+3$

Only 1 time instance 0, i.e. fully parallel

$Z_0 \rightarrow X_0$, $D_1 \rightarrow X_1$, $D_2 \rightarrow X_2$ and $D_3 \rightarrow X_3$

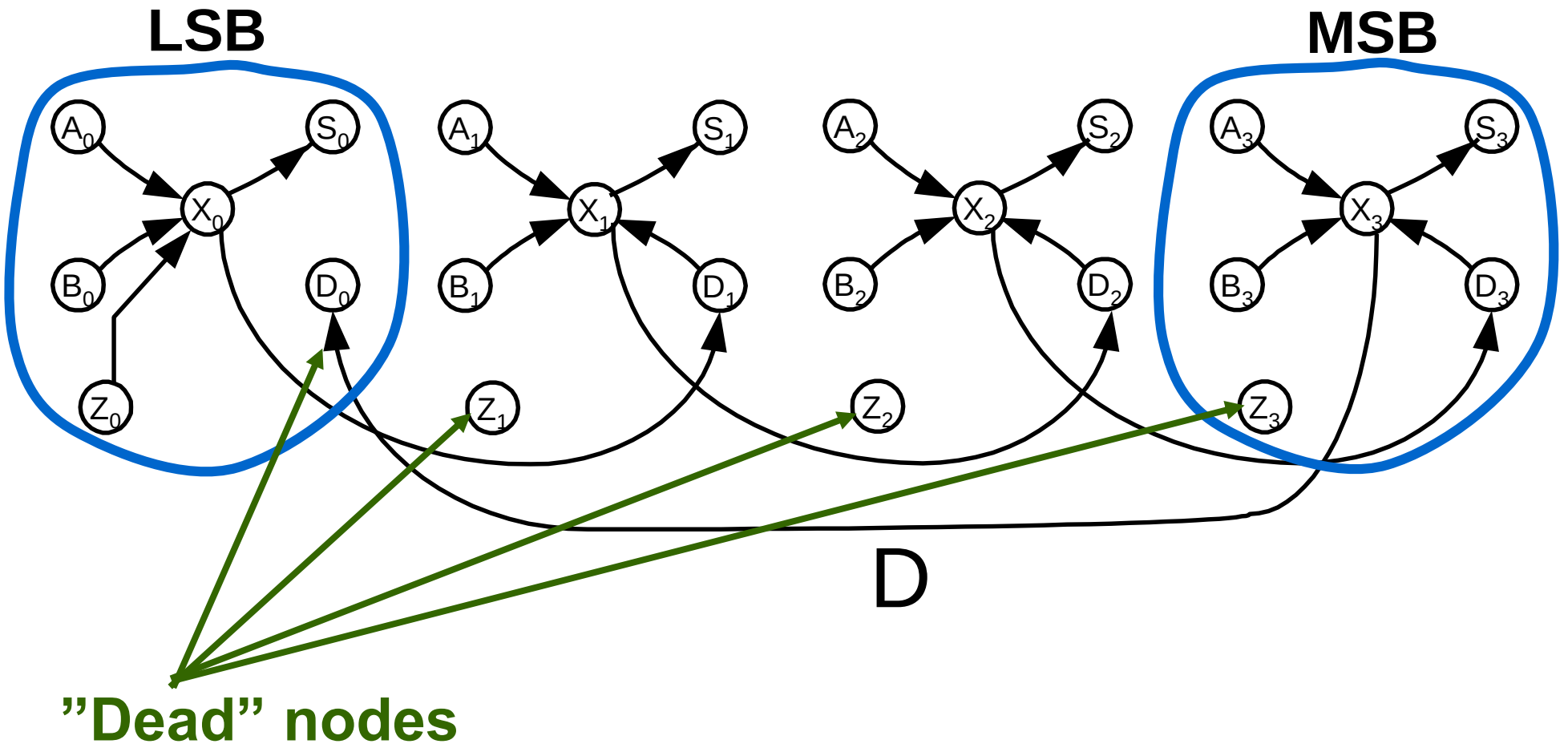
Bit-parallel Adder



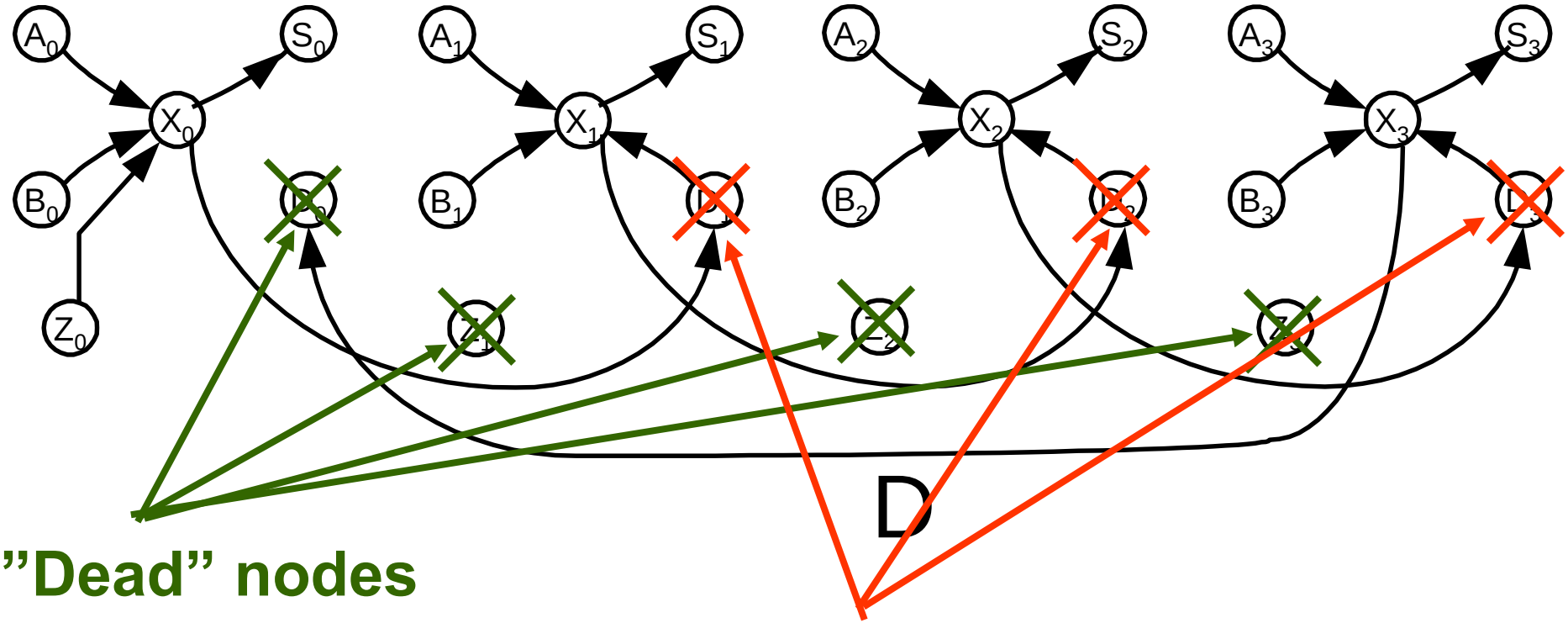
Only 1 time instance 0, i.e. fully parallel

$Z_0 \rightarrow X_0, D_1 \rightarrow X_1, D_2 \rightarrow X_2$ and $D_3 \rightarrow X_3$

Bit-parallel Adder



Remove ”Dead” and Dummy Nodes

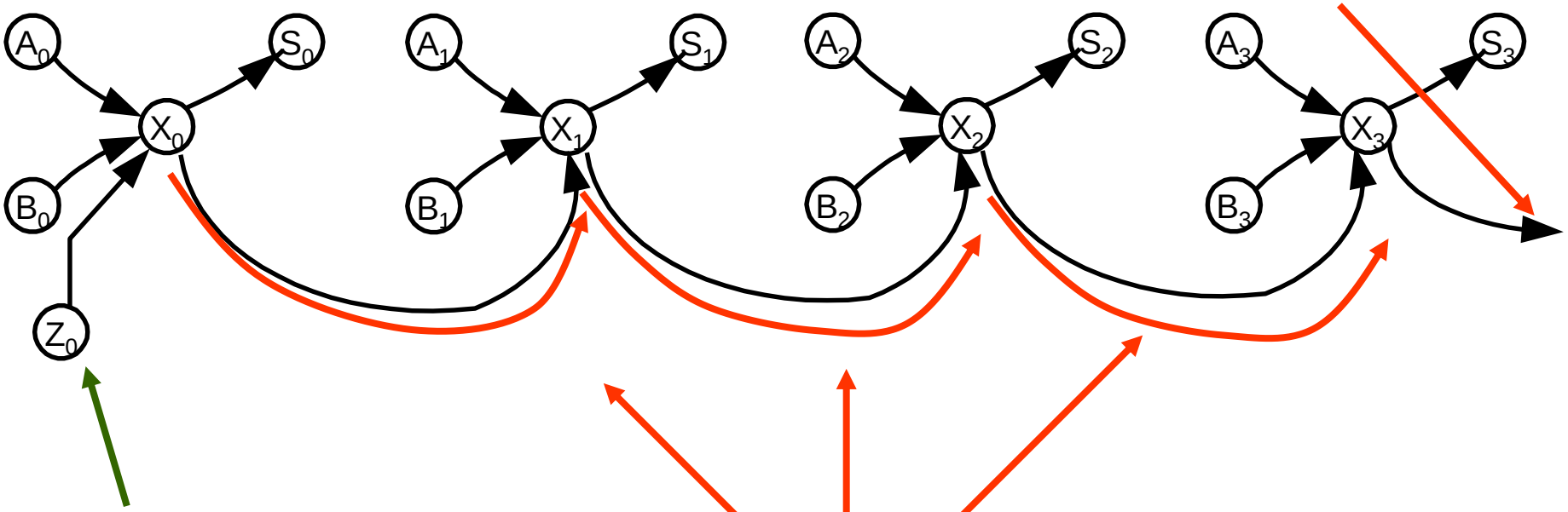


”Dead” nodes
can be removed

Dummy nodes
can be removed

Bit-parallel Adder

Carry from MSB as overflow or if to be used as a 4-bit module



Switch if to be used as a 4-bit module

Carry ≠ 0

Carry Ripple Adder

$$\begin{array}{r} a_3\ a_2\ a_1\ a_0 \\ b_3\ b_2\ b_1\ b_0 \\ \hline s_3\ s_2\ s_1\ s_0 \end{array}$$

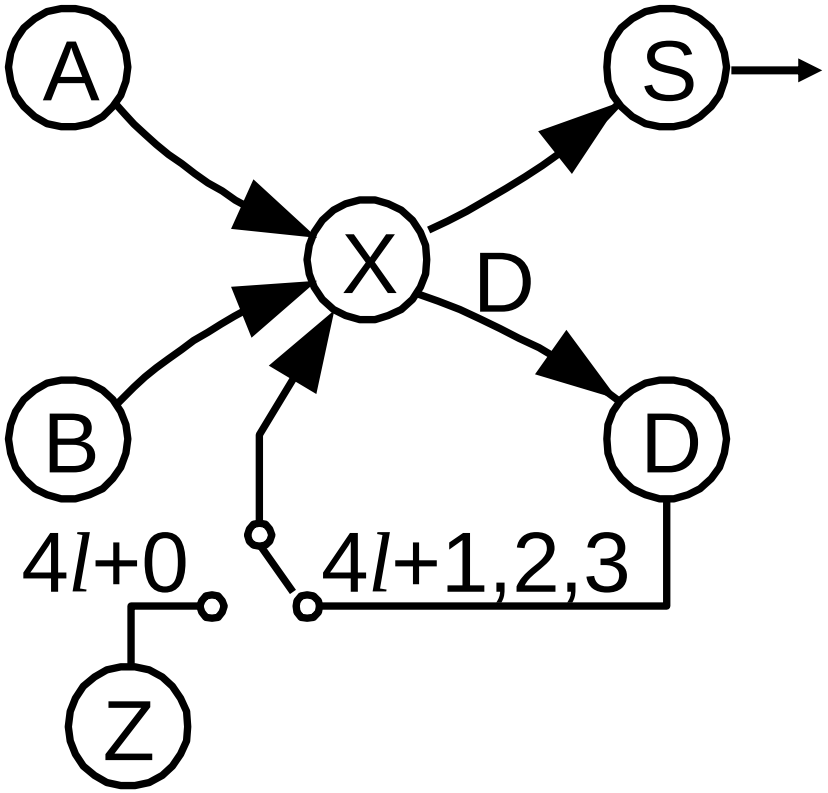
If Wordlength is not a multiple of J

- **determine** $L=lcm\{W,J\}$, lcm = least common multiple
- **replace switching instance** $Wl+u$ **with** L/W **instances**

  $Ll+u+wW$, for $w= 0..L/W-1$

  i.e. the switching periodicity has been changed from W **to** L
- **perform the unfolding as previously**
- **identify the correspondence between original instances and expanded instances**

Example: Unfold Bit-serial Adder by J=3



Wordlength $W=4$ not a multiple of the the unfolding factor $J=3$.

Determine

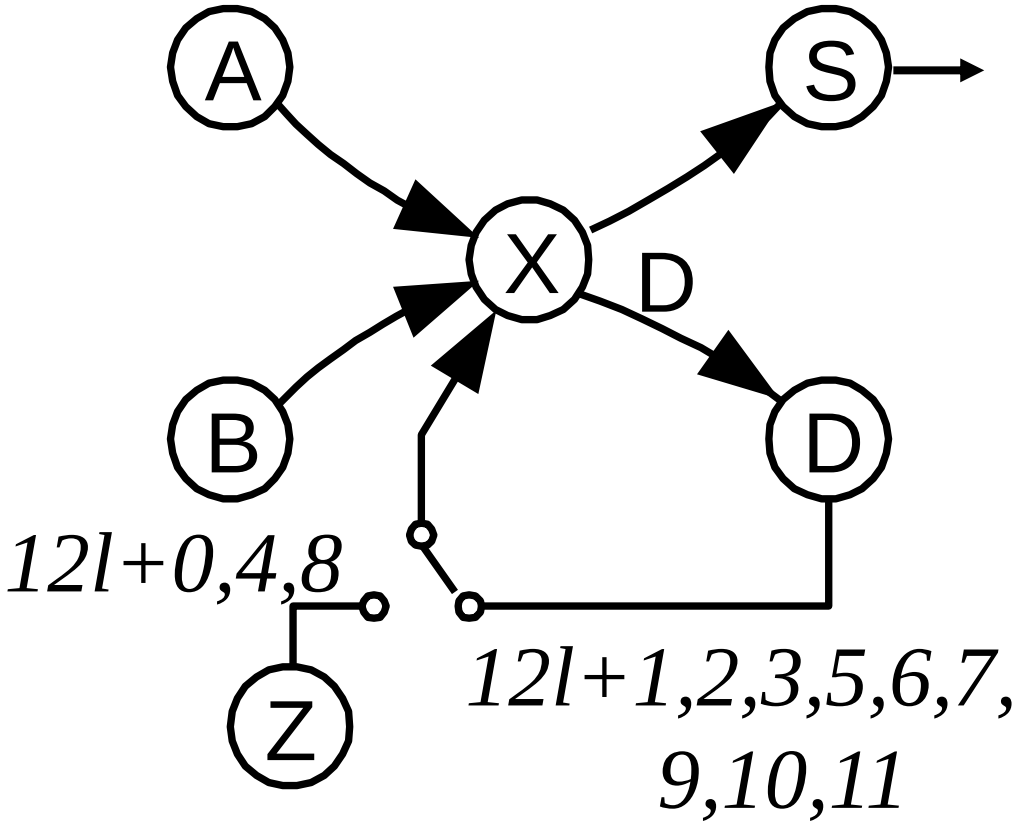
$L=lcm\{W,J\}=lcm\{4,3\}=12$

Replace

$Wl+u$ **with** $Ll+u+wW$

for $w=0$ **to** $L/W-1$

Example: Unfold Bit-serial Adder by J=3



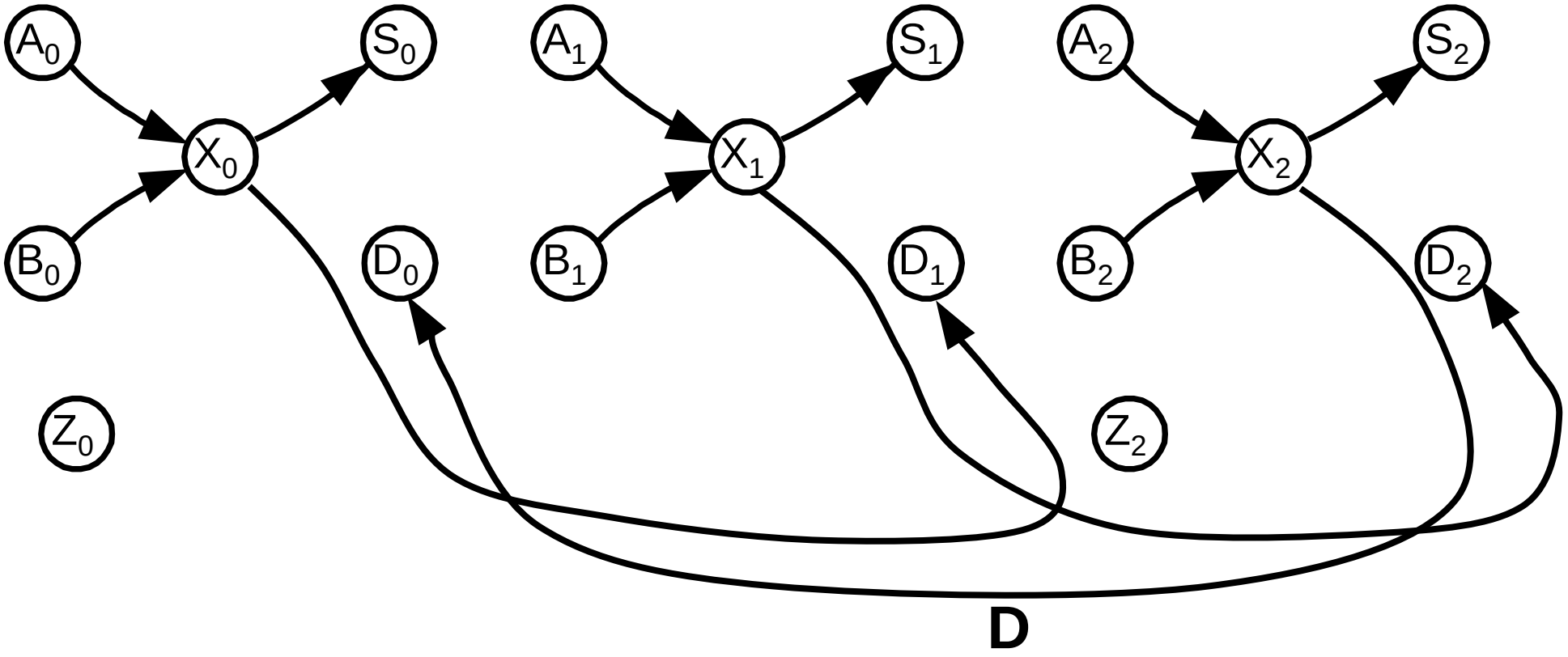
Replace
Wl+u **with** ***Ll+u+wW***
for ***w= 0*** **to** ***L/W-1***

4l+0 **is now equivalent to** ***12l+0, 12l+4*** **and** ***12l+8***

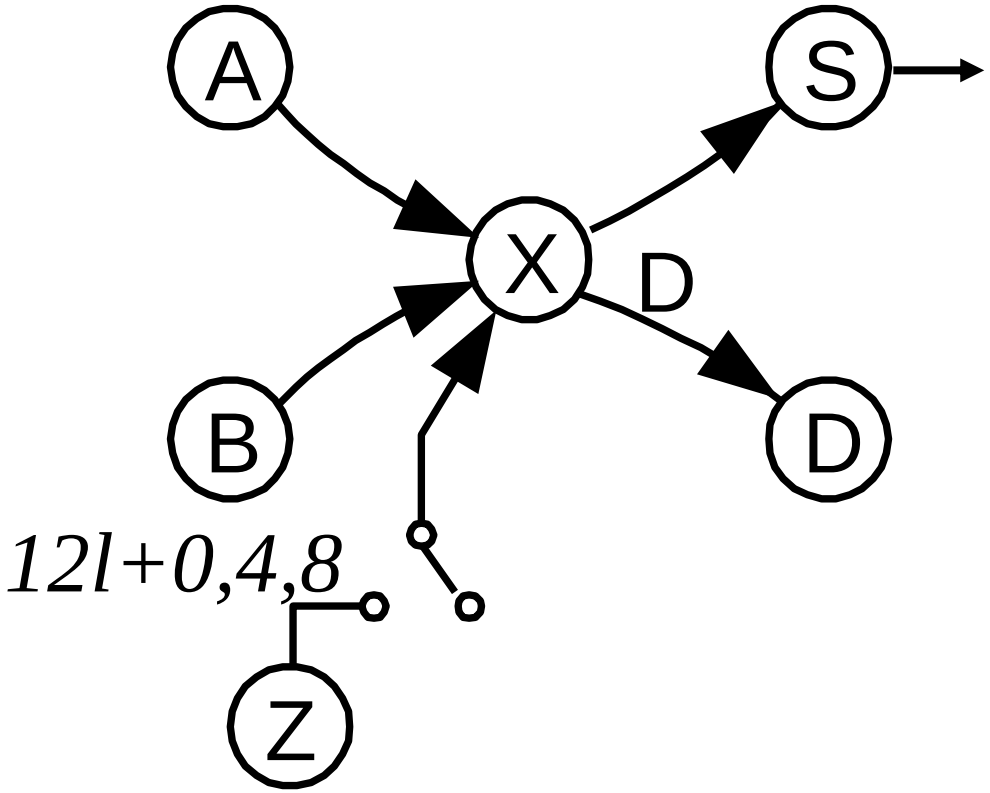
and so on…

Unfold as for the regular case.

Example: Unfold Bit-serial Adder by J=3



Example: Unfold Bit-serial Adder by J=3



Write the switching instance as

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

$$12l+0 = 3(4l+0)+0 \quad Z_0 \rightarrow X_0$$

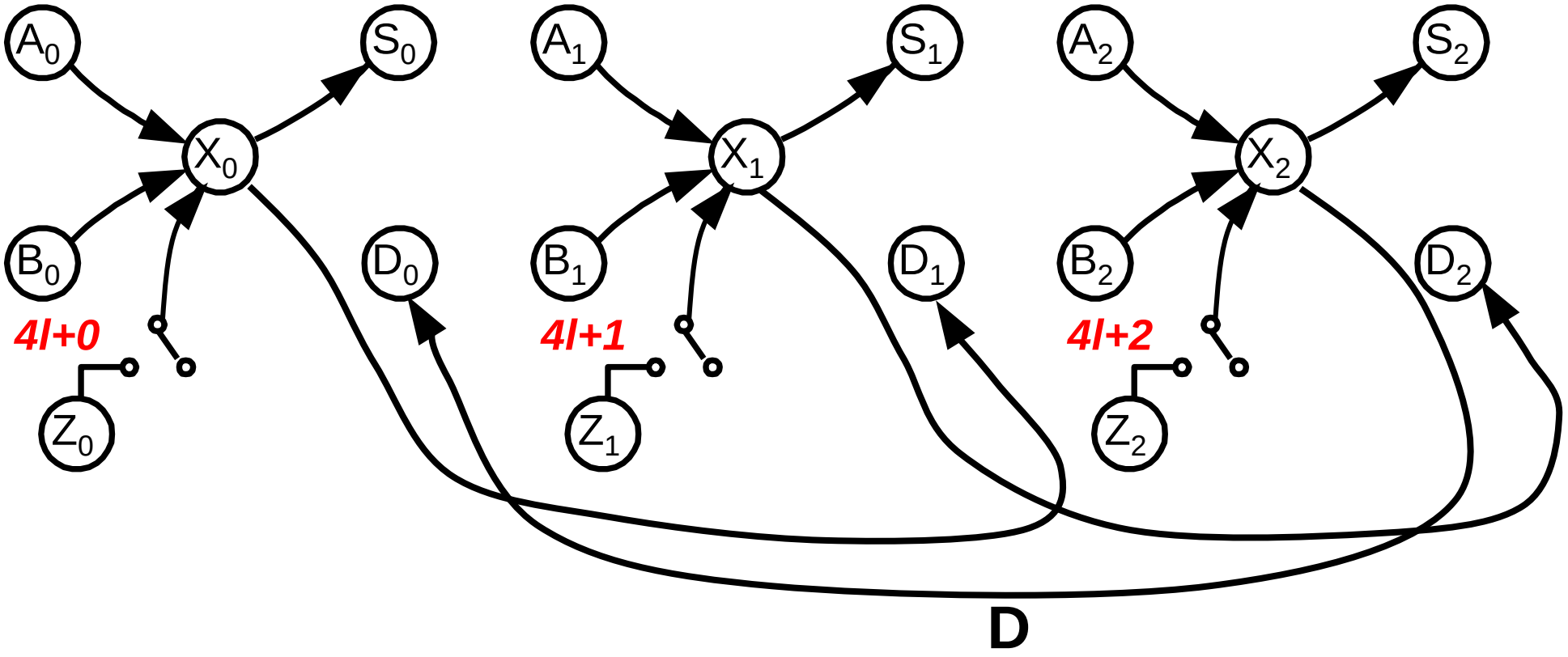
$$12l+4 = 3(4l+1)+1 \quad Z_1 \rightarrow X_1$$

$$12l+8 = 3(4l+2)+2 \quad Z_2 \rightarrow X_2$$

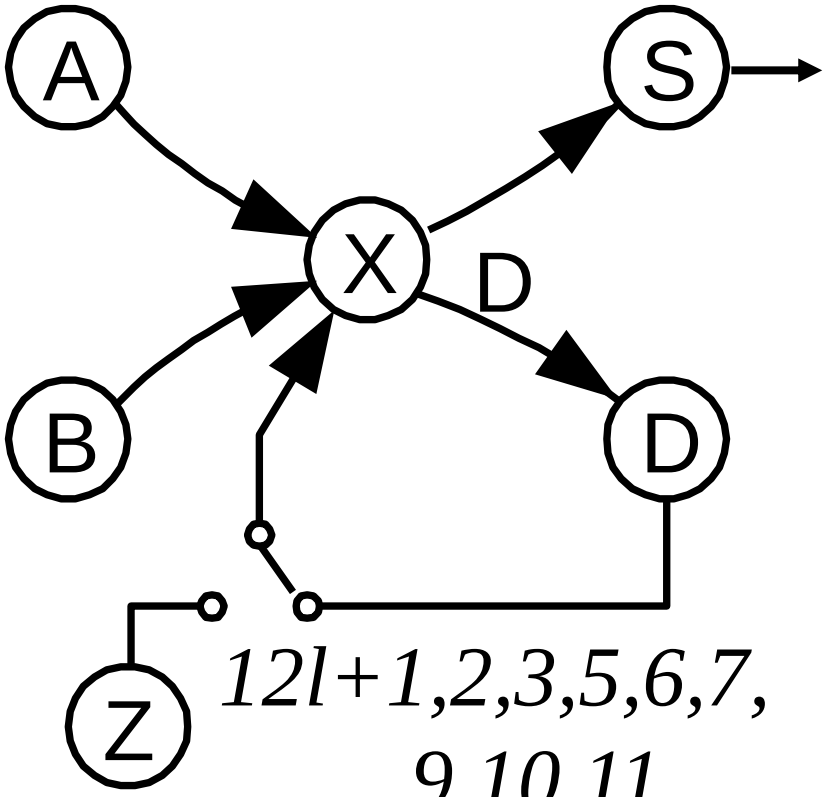
Draw an edge from node $U_{u\%J} \Rightarrow V_{u\%J}$, which is switched at time instance

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

Example: Unfold Bit-serial Adder by J=3



Example: Unfold Bit-serial Adder by J=3

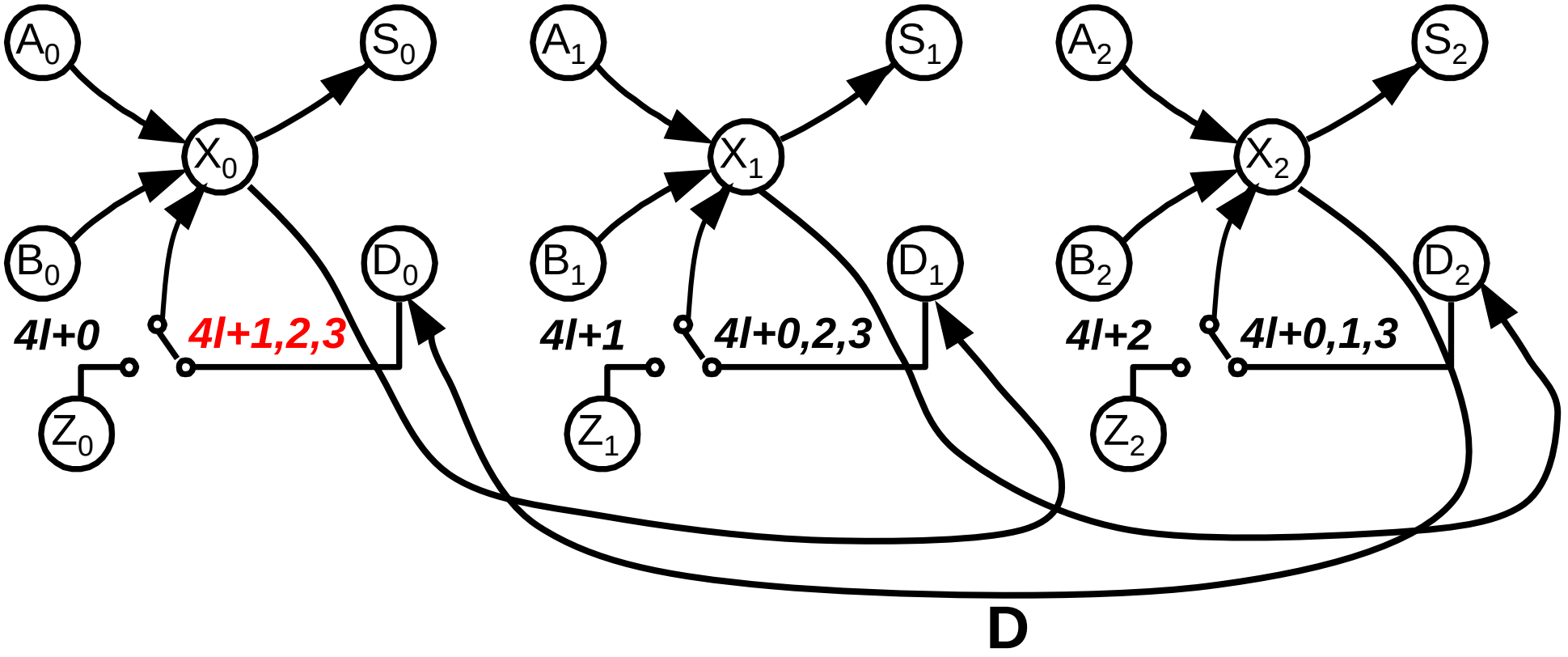


Write the switching instance as

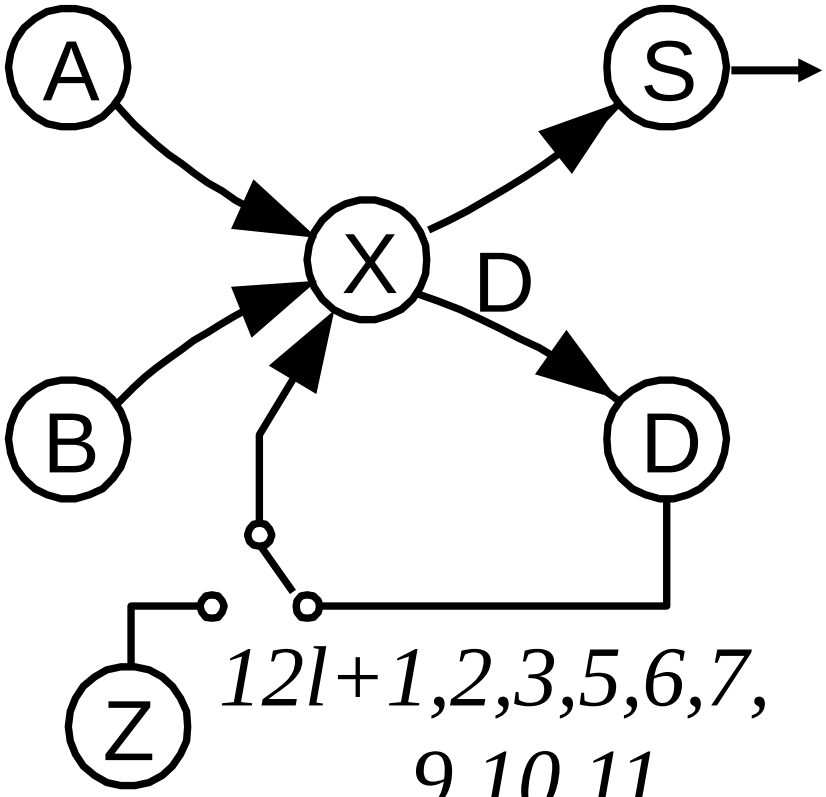
$$Wl + u = J(W'l + \lfloor u/J \rfloor) + (u\%J)$$

$12l+1$	=	$3(4l+0)+1$	
$12l+2$	=	$3(4l+0)+2$	
$12l+3$	=	$3(4l+1)+0$	$D_0 \rightarrow X_0$
$12l+5$	=	$3(4l+1)+2$	
$12l+6$	=	$3(4l+2)+0$	$D_0 \rightarrow X_0$
$12l+7$	=	$3(4l+2)+1$	
$12l+9$	=	$3(4l+3)+0$	$D_0 \rightarrow X_0$
$12l+10$	=	$3(4l+3)+1$	
$12l+11$	=	$3(4l+3)+2$	

Example: Unfold Bit-serial Adder by J=3



Example: Unfold Bit-serial Adder by J=3



Write the switching instance as

Wl + u = J(W'l + ⌊u/J⌋) + (u%J)

$12l+1 = 3(4l+0)+1$ $D_1 \rightarrow X_1$

$12l+2 = 3(4l+0)+2$

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

$12l+5 = 3(4l+1)+2$

$12l+6 = 3(4l+2)+0$

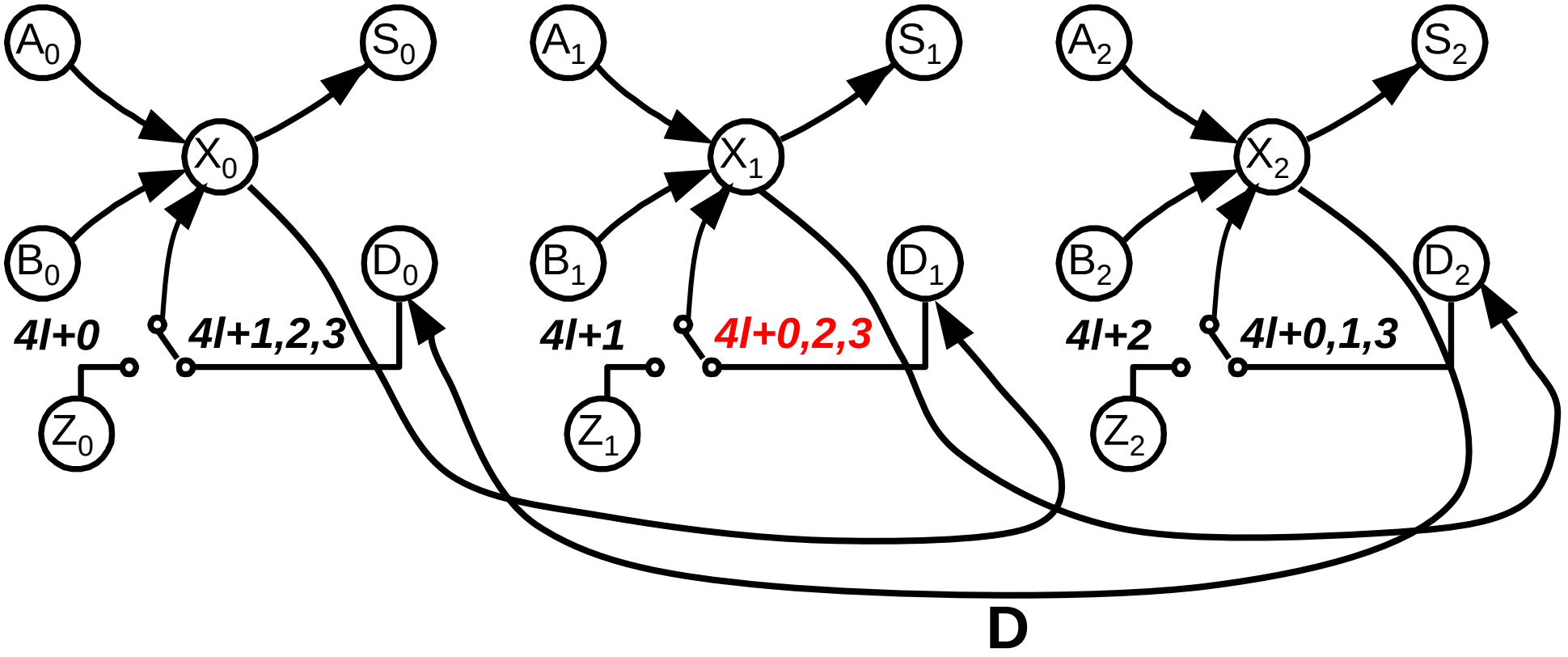
$12l+7 = 3(4l+2)+1$ $D_1 \rightarrow X_1$

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

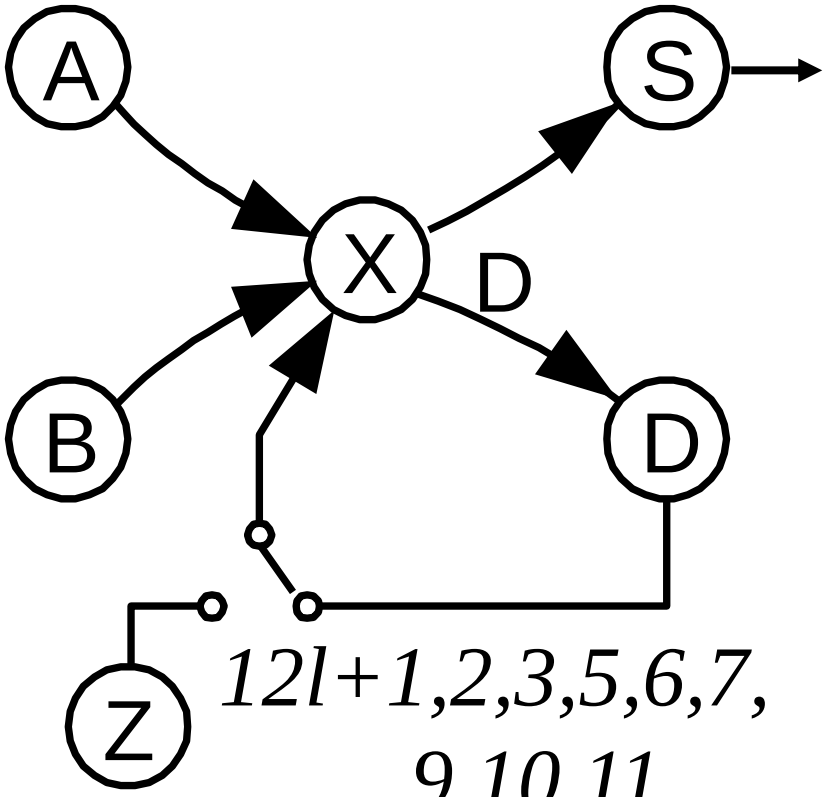
$12l+10 = 3(4l+3)+1$ $D_1 \rightarrow X_1$

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

Example: Unfold Bit-serial Adder by J=3



Example: Unfold Bit-serial Adder by J=3



Write the switching instance as

Wl + u = J(W'l + ⌊u/J⌋) + (u%J)

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

$12l+2 = 3(4l+0)+2$ $D_2 \rightarrow X_2$

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

$12l+5 = 3(4l+1)+2$ $D_2 \rightarrow X_2$

$12l+6 = 3(4l+2)+0$

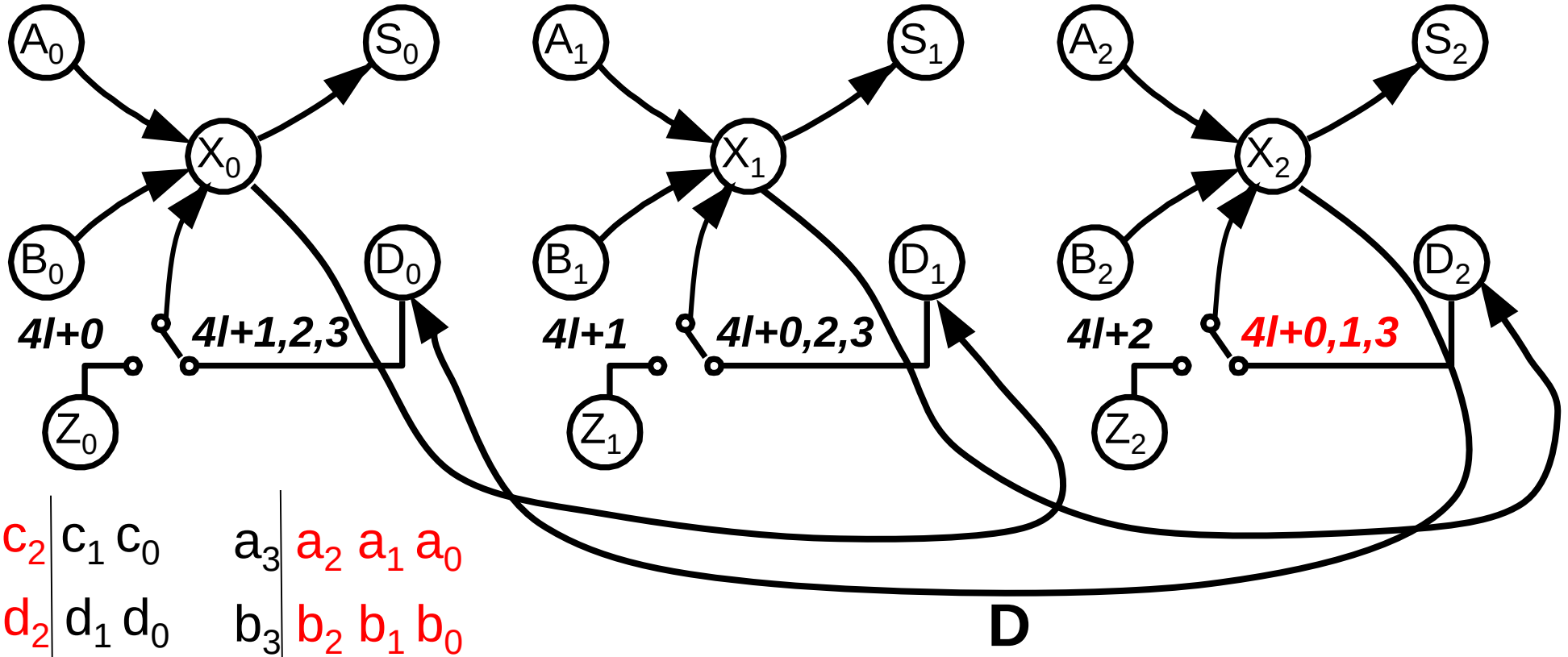
$12l+7 = 3(4l+2)+1$

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

$12l+10 = 3(4l+3)+1$

$12l+11 = 3(4l+3)+2$ $D_2 \rightarrow X_2$

Example: Unfold Bit-serial Adder by J=3



c_3	c_2	c_1	c_0	a_3	a_2	a_1	a_0
d_3	d_2	d_1	d_0	b_3	b_2	b_1	b_0
s'_3	s'_2	s'_1	s'_0	s_3	s_2	s_1	s_0

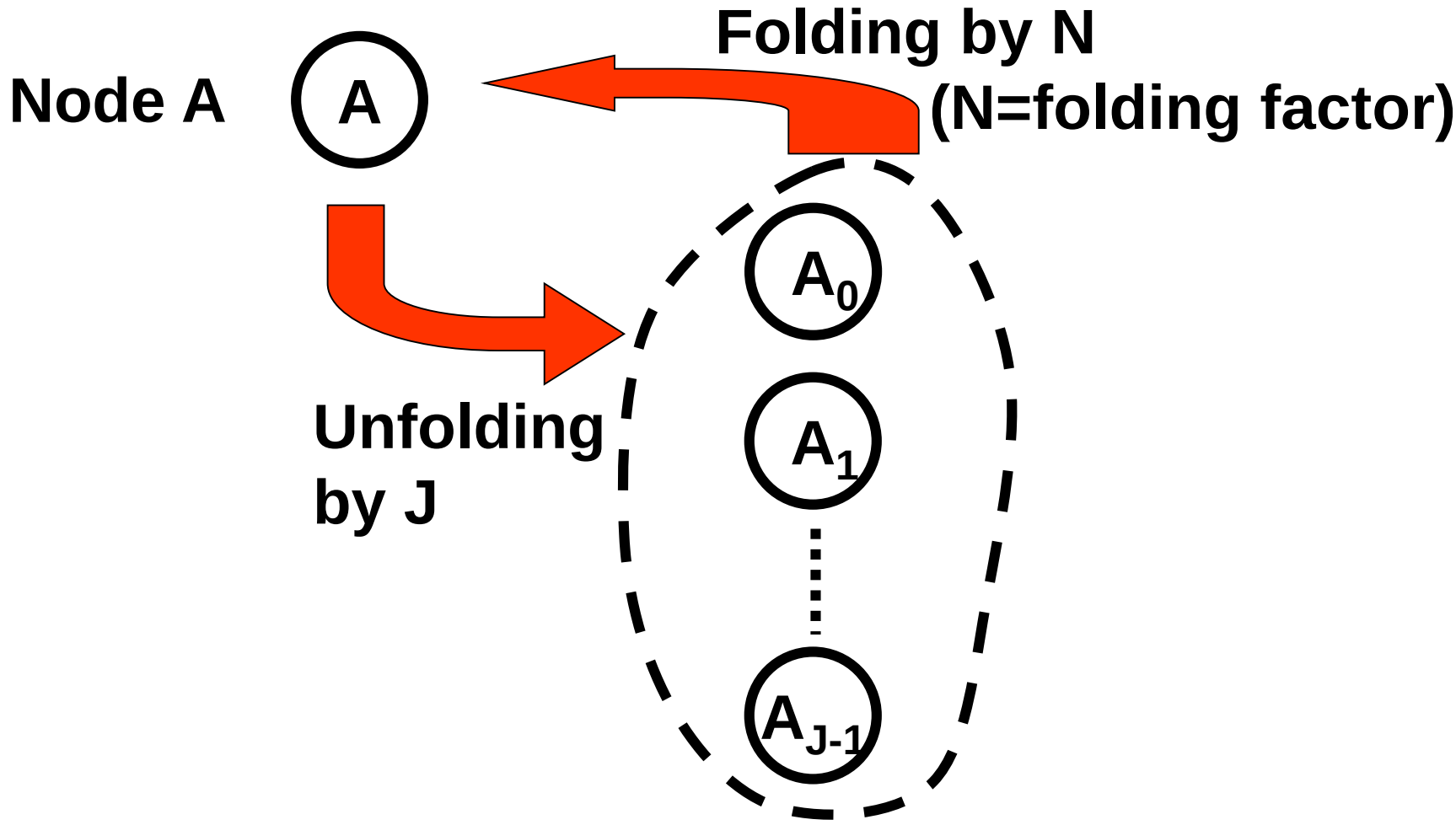
End of Unfolding

Folding

Chapter 6

What is folding??

Folding is the ”Inverse” of Unfolding



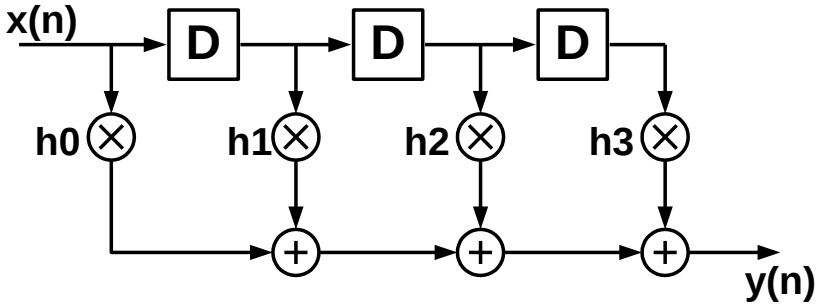
Folding?

- Used to minimize silicon area (trading area for time)!
- A way to systematically determine the control circuits in DSP architectures by folding transformation, where multiple algorithm operations are time-multiplexed to a single functional unit.
- Use for synthesis of DSP architectures that can be operated at single or multiple clocks.
- Use to reduce the number of hardware functional units (FUs) such as adders and mults by a factor of N at the expense of increasing computation time by a factor of N.

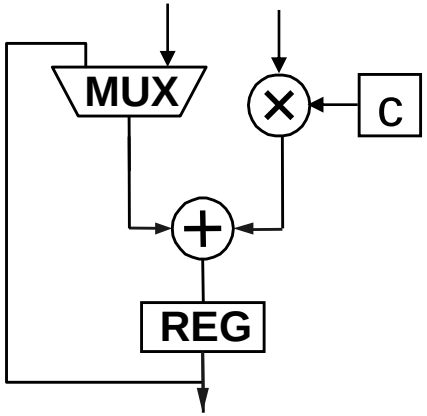
But Folding lead to an architecture that uses a large number of registers and thus a register minimization technique needs sometime to be applied.

Hardware Mapped vs. Time multiplexed

$$FIR: y(n) = \sum_{k=0}^{N-1} h(k)x(n-k)$$

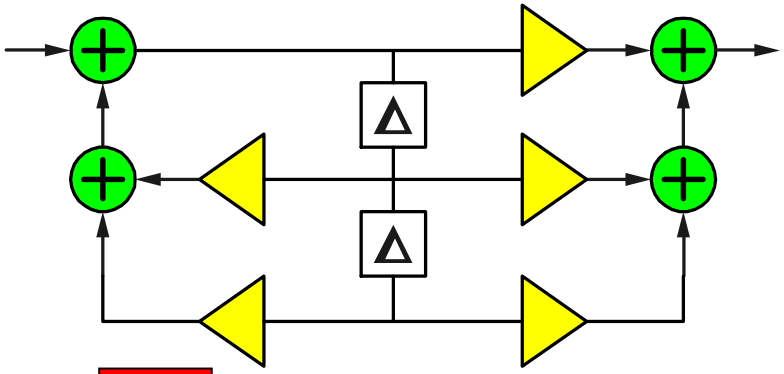


1	sample/cc
N	fixed multipliers
N-1	adders

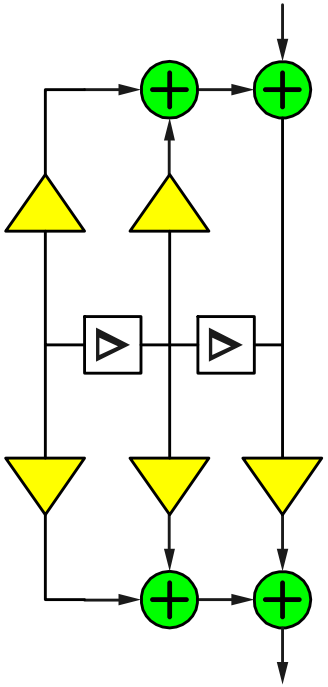


N	cc/sample
1	generalized multiplier
1	adders
1	coefficient memory
+	control

Hardware Mapped vs. Time multiplexed/Microcoded



Biquad Filter

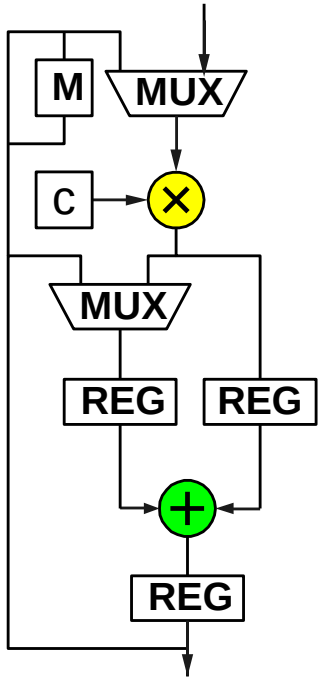


Hardware mapped

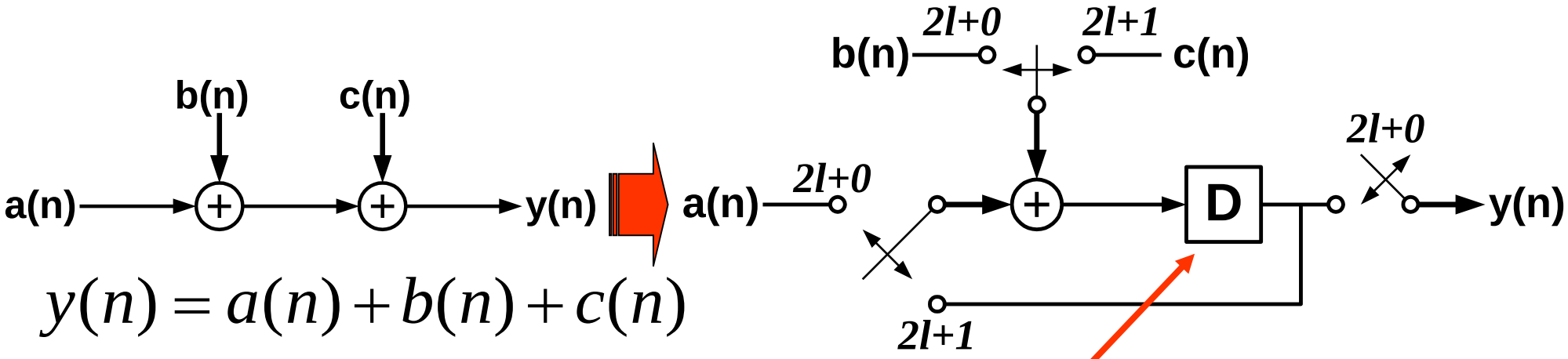
- 5 mult with fixed coeffecients
- 4 adders
- 2 delays
- Latency=1cc

Microcoded

- 1 mult
- 1 adder
- Latency=5cc
- Coeff Memory
- 3 Registers
- Controller



Folding – Time-shared Architecture



$$y(n) = a(n) + b(n) + c(n)$$

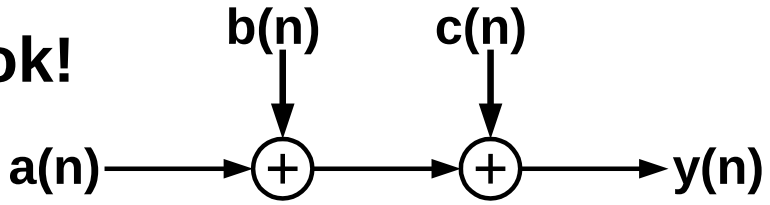
Folding is a technique to reduce the silicon area by time-multiplexing many operations into single functional units.

- The right figure shows a 2 times folded architecture where 2 additions are folded, or time-multiplexed, to a single adder

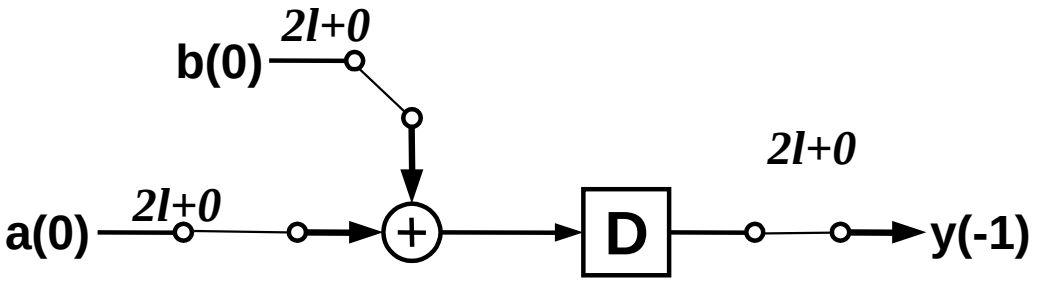
- Folding introduces registers/storage
- Computation time increased, e.g. one output sample every 2 cc (one input signal consumed every 2cc)

Folding Example – A more detailed look!

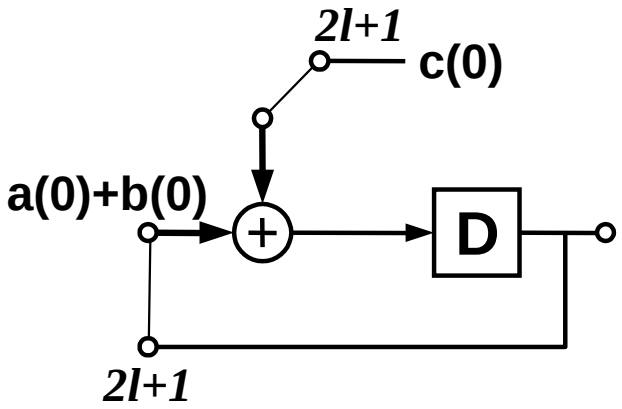
$$y(n) = a(n) + b(n) + c(n)$$



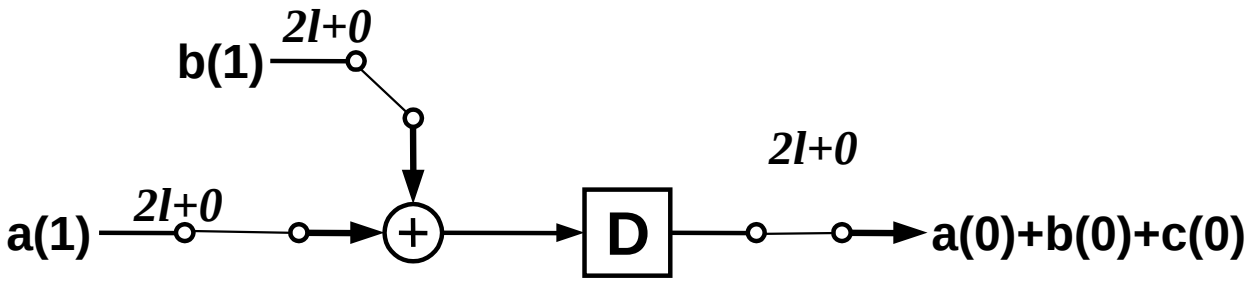
Cycle 0



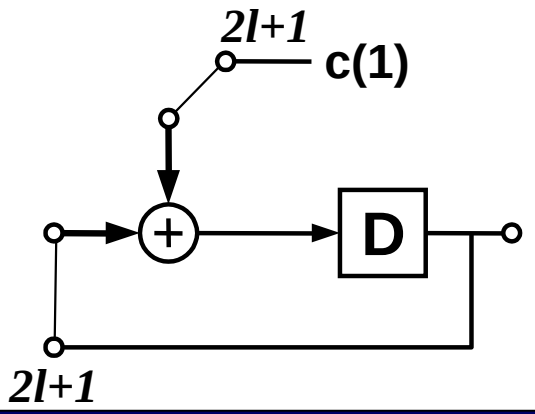
Cycle 1



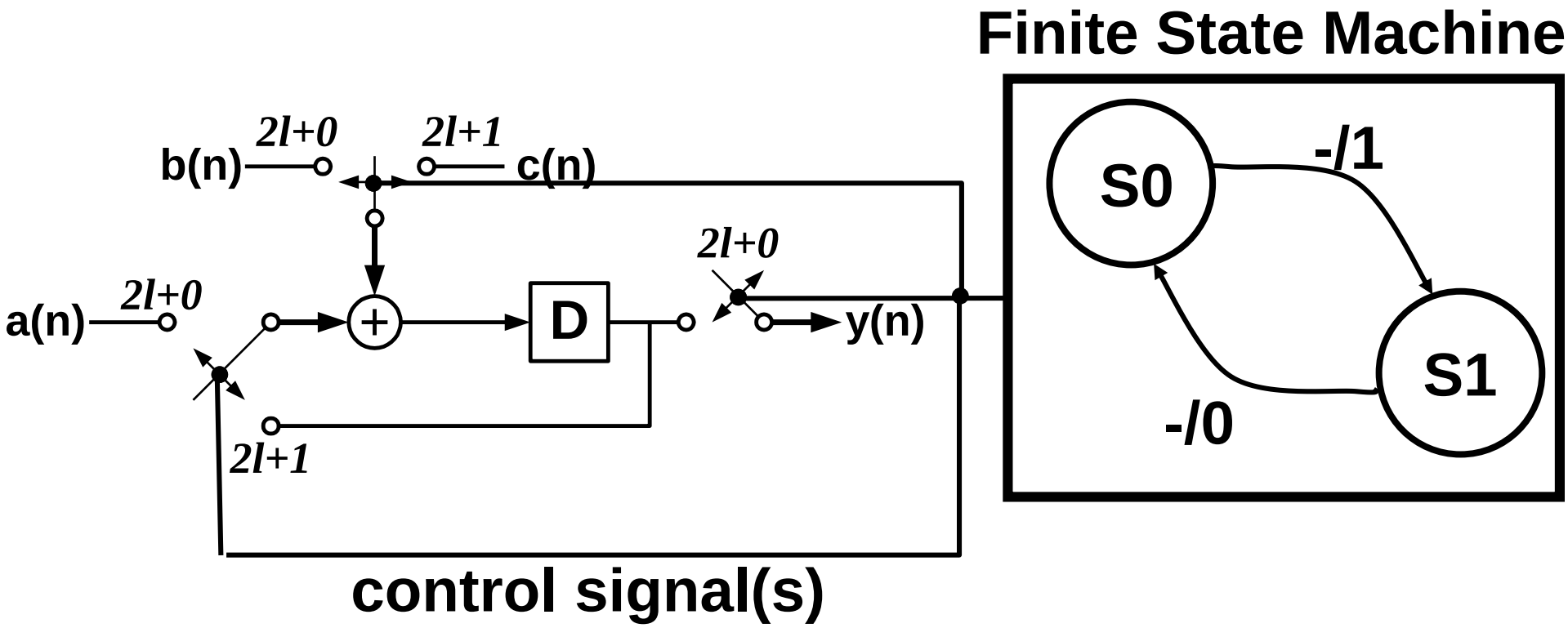
Cycle 2



Cycle 3



Control Unit



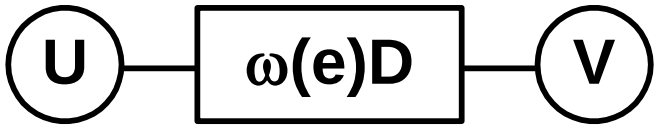
Control units can be complex in large systems!

Folding

- Reduce hardware by N-folding
- $T_{computation}$ increased by N ⇨ Latecy
- Extremes
  - Fully parallel
  - Time multiplexed = 1 unit per algorithmic operation
- Folding ⇨
  - extra registers, i.e. extra storage
  - a more complex control unit
  - more latency

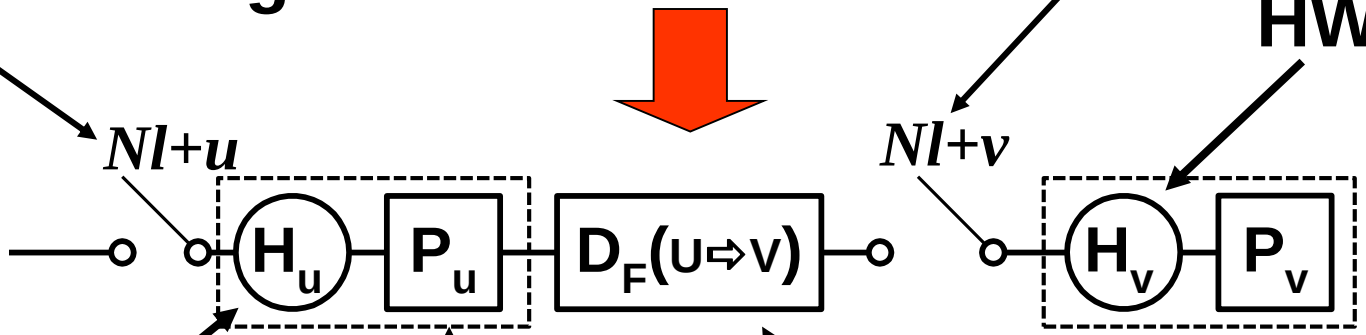
Folding Transformation

N=folding factor
Nr. of operations folded to a single unit



l = iteration

HW-unit V



HW-unit U

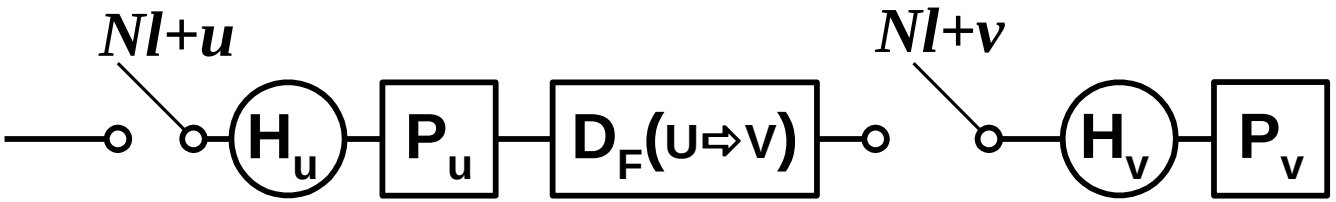
Level of Pipeline

Delays in folded graph

u and v are folding order, i.e. scheduled time

$$0 \le u, v \le N - 1$$

Folding Transformation



- H_u is pipelined by P_u stages and its output is available at $\mathbf{N}l + \mathbf{u} + \mathbf{P_u}$.

- Edge U→V has w(e) delays ⇒ the l-th iteration of U is used by $(l+\mathbf{w(e)})$ th iteration of node V, which is executed at $\mathbf{N}(l + \mathbf{w(e)}) + \mathbf{v}$.

So, the result should be stored for :

$$\mathbf{D_F(U \rightarrow V) = [N}(l + \mathbf{w(e)}) + \mathbf{v}] - [\mathbf{N}l + \mathbf{P_u} + \mathbf{u}]$$

$$\Rightarrow \mathbf{D_F(U \rightarrow V) = Nw(e) - P_u + v - u}$$ **(independent of *l*)**

Folding Set

- A folding set is an ordered set of operations to be executed on the same functional unit.
- The folding set are typically obtained from a scheduling and allocation algorithm (ref. Appendix B)
- The folding set represents underlying folding transformation
- Each set contain N entries, N=folding factor.

Folding order: 0 1 2 (… N-1)

$$S_1 = \{A_1, 0, A_2\}$$

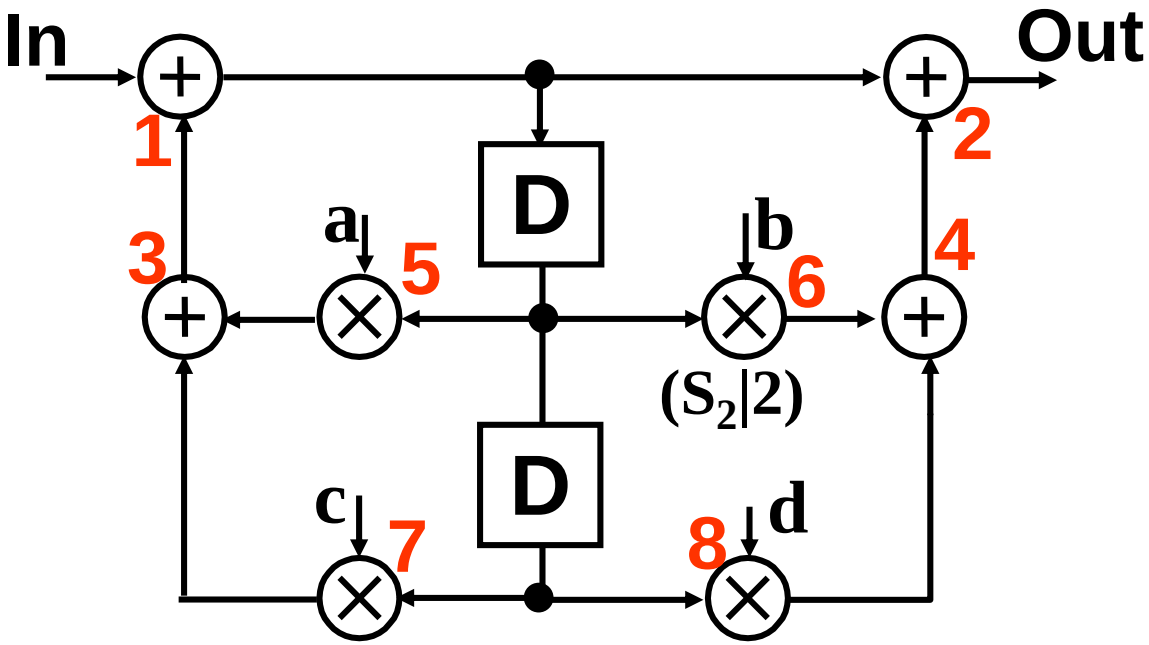
A_1 belongs to folding set S_1 with folding order 0

$(S_1|0)$

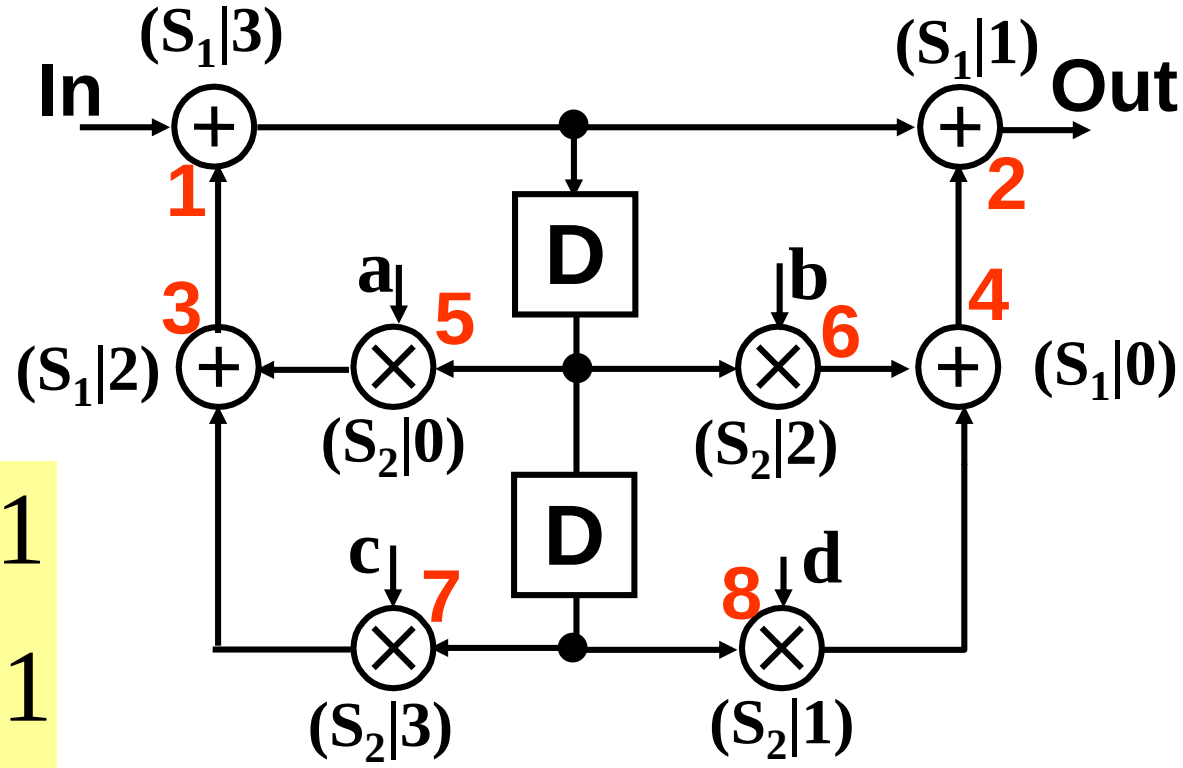
N=3

Null operation

Ex. Folding of Biquad filter



Ex. Folding of Biquad filter



$$T_{adder} = 1$$

$$P_{adder} = 1$$

$$T_{mult} = 2$$

$$P_{mult} = 2$$

Additions

$$S_1 = \{4,2,3,1\}$$

Multiplication

$$S_2 = \{5,8,6,7\}$$

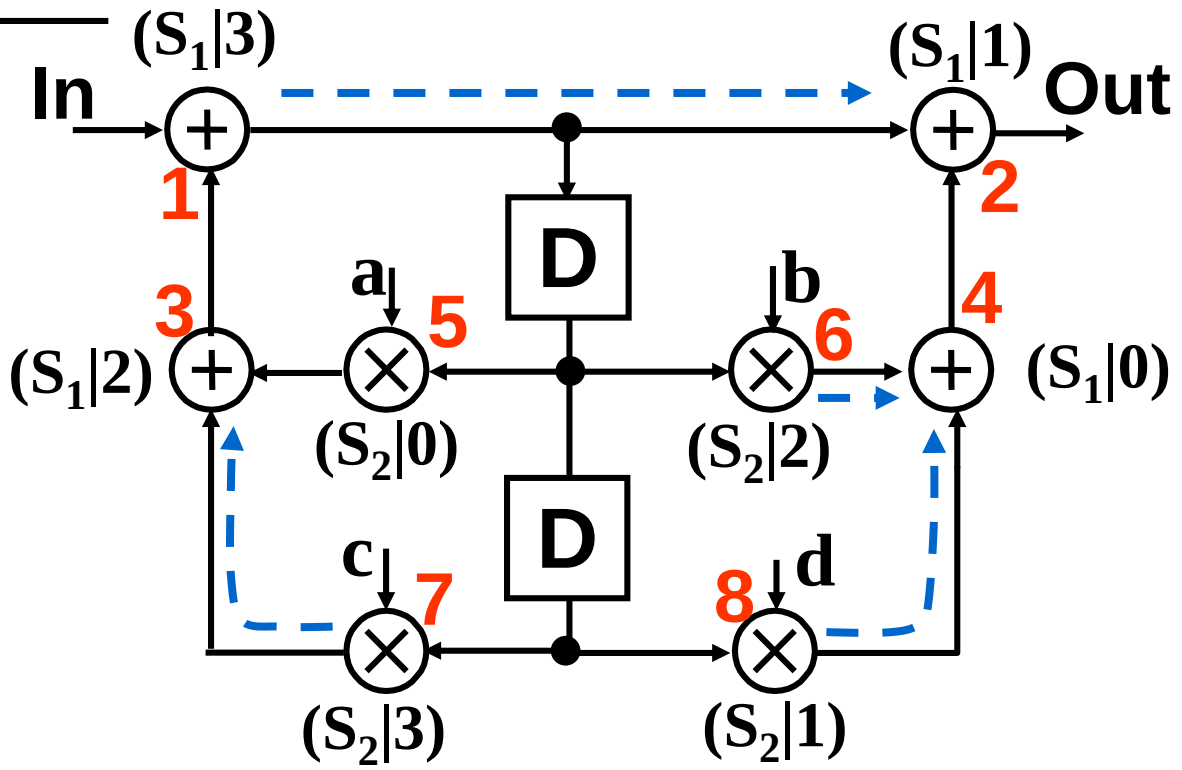
Folding of Biquad filter, N=4

receive send

$$D_F(U \rightarrow V) = Nw(e) - P_u + v - u$$

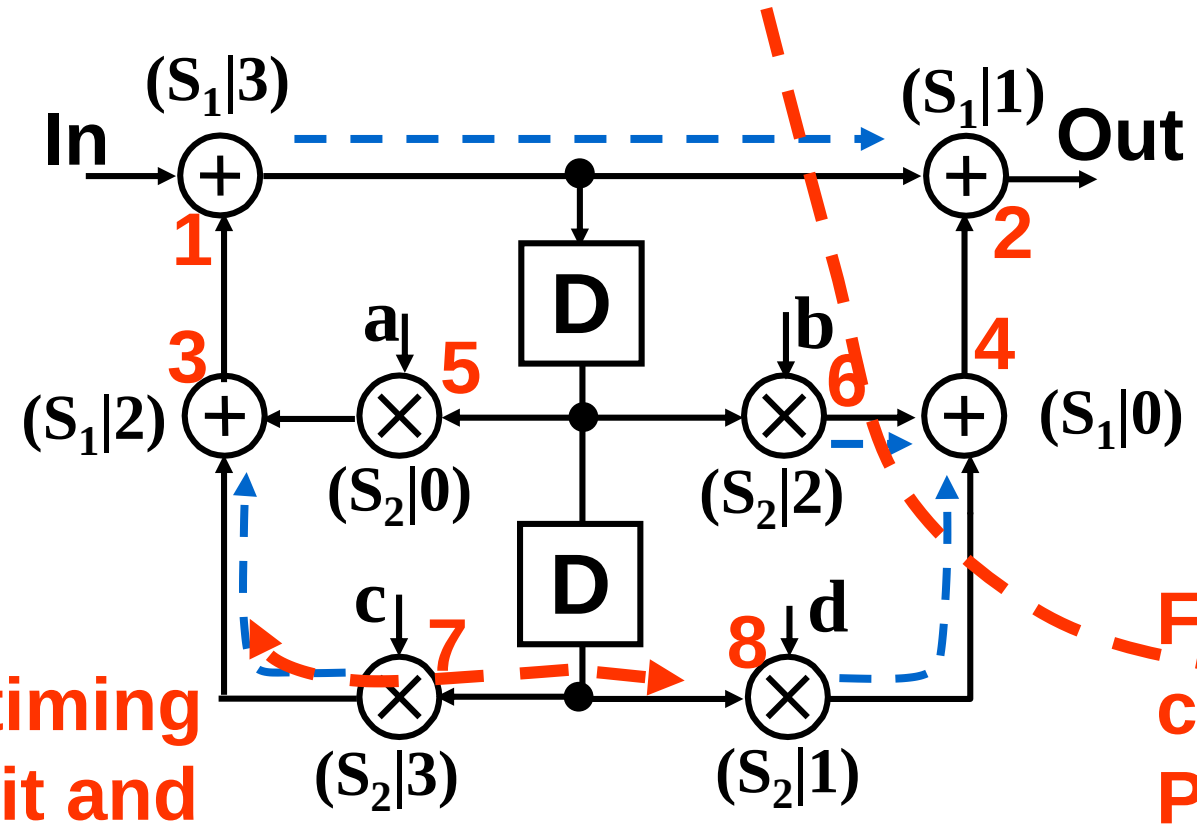
A delay between two edges can not be negative!

$D_F(1 \rightarrow 2) = -3$
$D_F(1 \rightarrow 5) = 0$
$D_F(1 \rightarrow 6) = 2$
$D_F(1 \rightarrow 7) = 7$
$D_F(1 \rightarrow 8) = 5$
$D_F(3 \rightarrow 1) = 0$
$D_F(4 \rightarrow 2) = 0$
$D_F(5 \rightarrow 3) = 0$
$D_F(6 \rightarrow 4) = -4$
$D_F(7 \rightarrow 3) = -3$
$D_F(8 \rightarrow 4) = -3$



$D_F(U \rightarrow V) < 0$ ➡ **Not Valid folding**

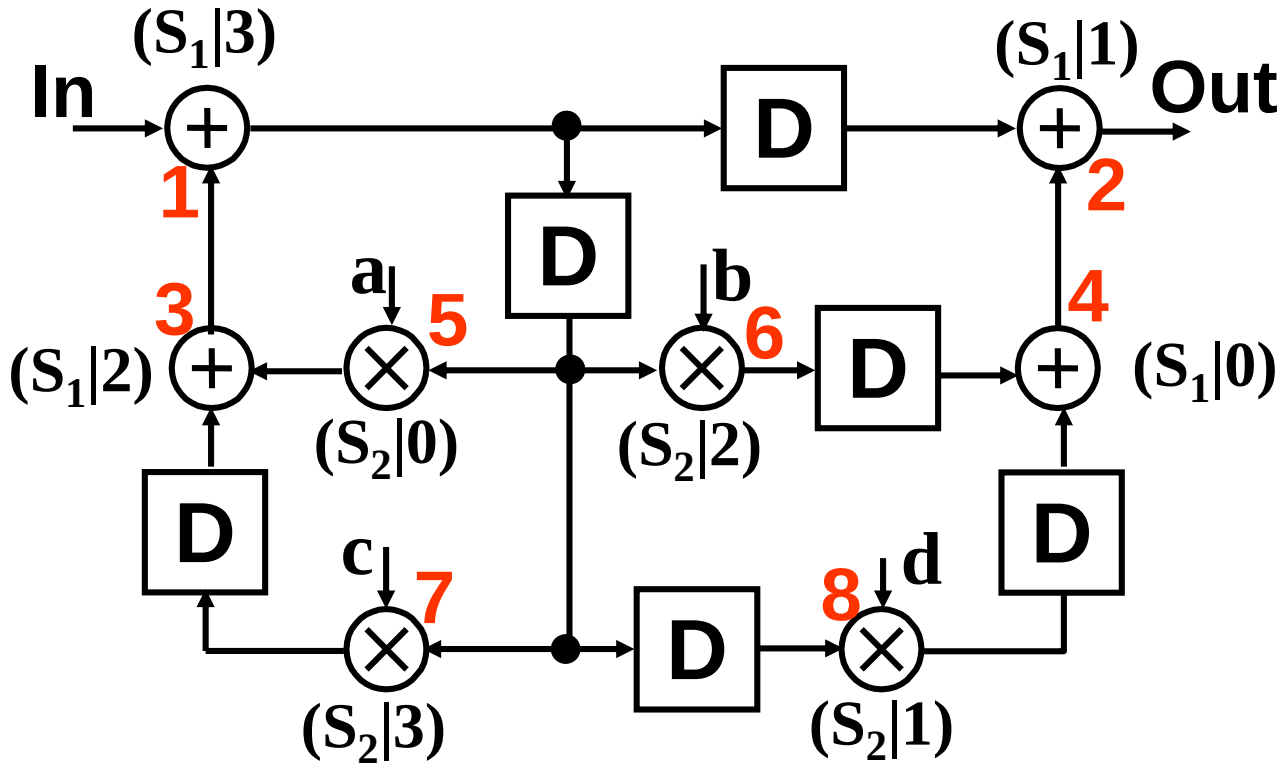
Retiming: Folding of Biquad filter, N=4



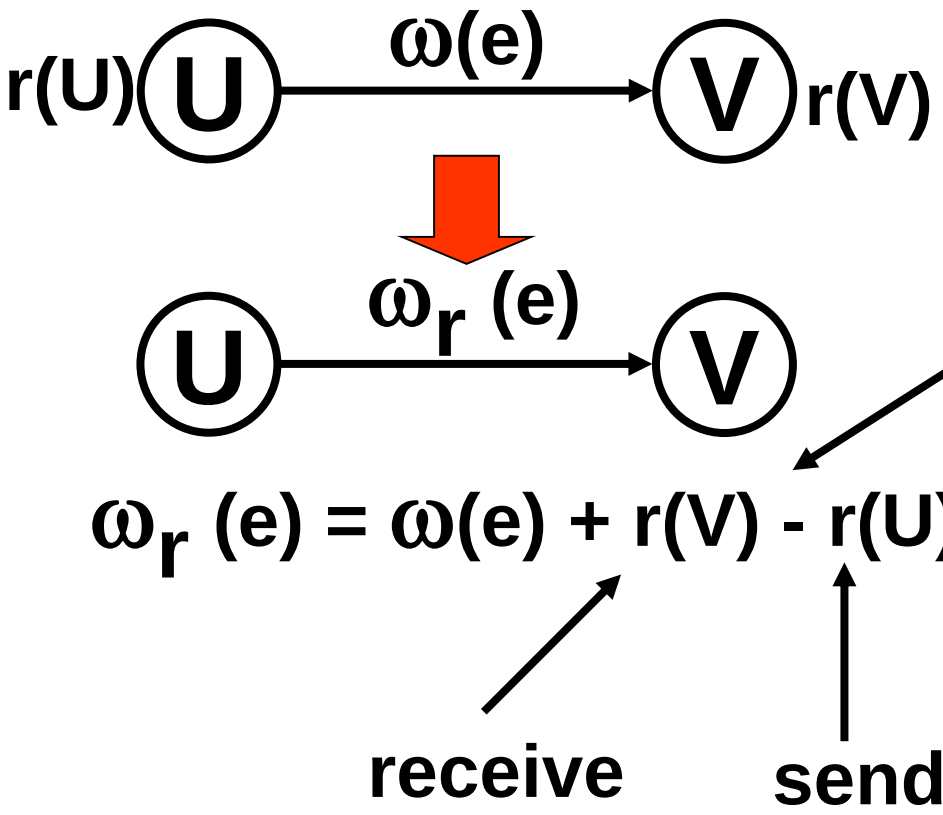
Retiming Split and move delay

Feedforward cutset ⇨ Pipelining

Folding of retimed Biquad filter



Systematic way of: Retiming for Folding



If $D'_F(U \rightarrow V)$ is the folded delays of the edge $U \rightarrow V$ for the retimed graph then $D'_F(U \rightarrow V) \geq 0 \Rightarrow$

$$Nw_r(e) - P_U + v - u \geq 0$$

$$\Rightarrow N(w(e) + r(V) - r(U)) - P_U + v - u \geq 0$$

$$\Rightarrow N(r(U) - r(V)) \leq \underline{Nw(e) - P_U + v - u}$$

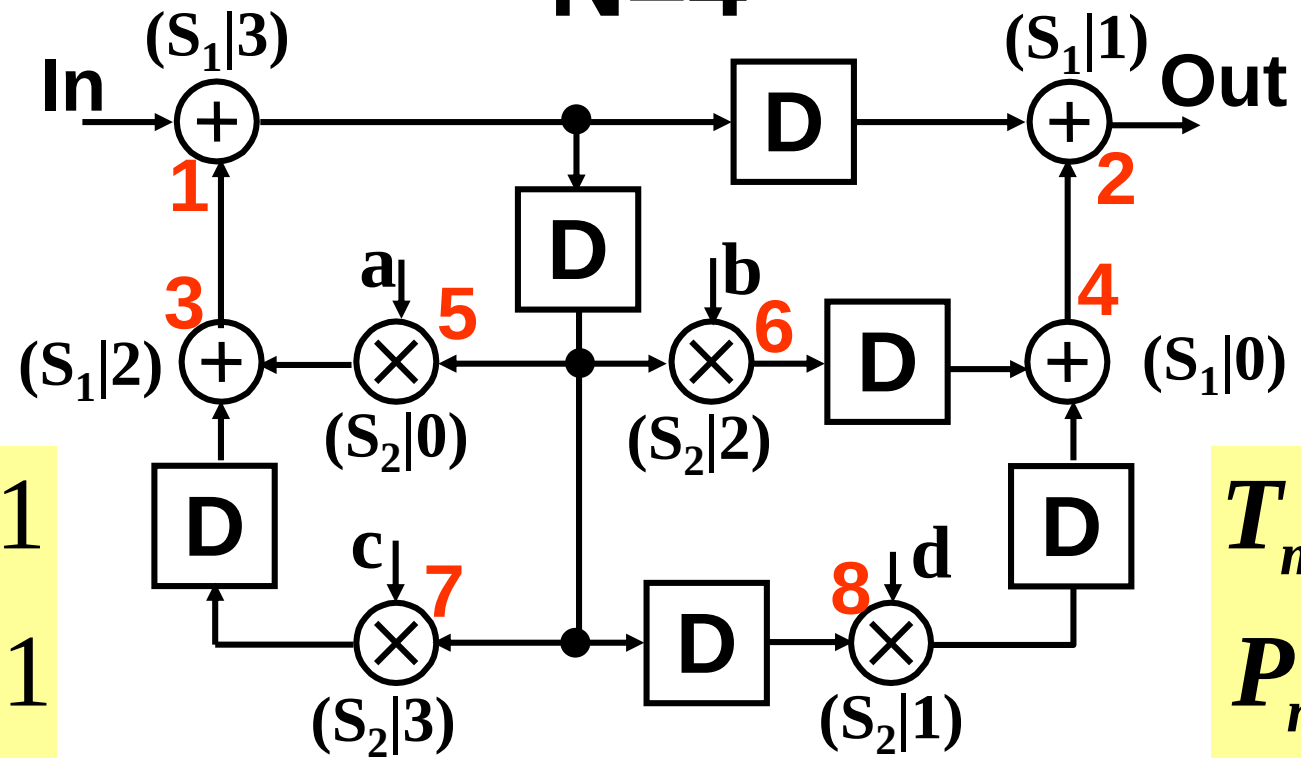
$$\Rightarrow r(U) - r(V) \leq D_F(U \rightarrow V) / N$$

$$\Rightarrow r(U) - r(V) \leq \lfloor D_F(U \rightarrow V) / N \rfloor$$

$\Rightarrow$(floor since retiming values are integers)

Then solve the the system of inequalities!

Folding of retimed Biquad filter, N=4



$$T_{adder} = 1 \qquad P_{adder} = 1$$

$$T_{mult} = 2 \qquad P_{mult} = 2$$

Additions $S_1 = \{4,2,3,1\}$

Multiplication $S_2 = \{5,8,6,7\}$

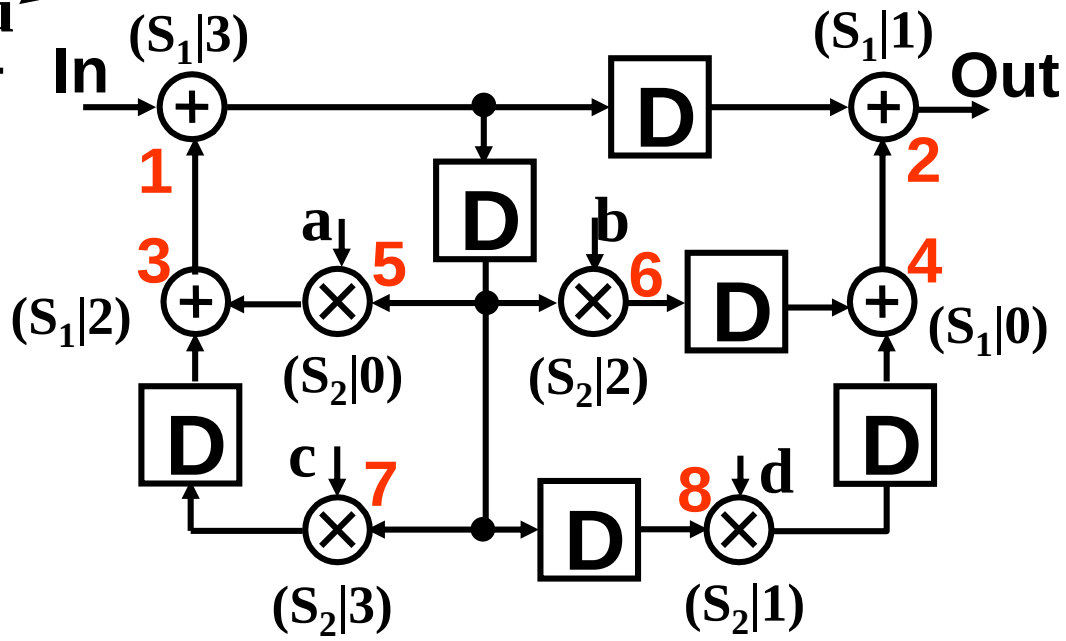
Folding of retimed Biquad filter, N=4

receive

send

$$D_F(U \rightarrow V) = Nw(e) - P_u + v - u$$

$D_F(1 \rightarrow 2) = 4(1) - 1 + 1 - 3 = 1$
$D_F(1 \rightarrow 5) = 4(1) - 1 + 0 - 3 = 0$
$D_F(1 \rightarrow 6) = 4(1) - 1 + 2 - 3 = 2$
$D_F(1 \rightarrow 7) = 4(1) - 1 + 3 - 3 = 3$
$D_F(1 \rightarrow 8) = 4(2) - 1 + 1 - 3 = 5$
$D_F(3 \rightarrow 1) = 4(0) - 1 + 3 - 2 = 0$
$D_F(4 \rightarrow 2) = 4(0) - 1 + 1 - 0 = 0$
$D_F(5 \rightarrow 3) = 4(0) - 2 + 2 - 0 = 0$
$D_F(6 \rightarrow 4) = 4(1) - 2 + 0 - 2 = 0$
$D_F(7 \rightarrow 3) = 4(1) - 2 + 2 - 3 = 1$
$D_F(8 \rightarrow 4) = 4(1) - 2 + 0 - 1 = 1$



Valid folding

$$D_F(U \rightarrow V) \geq 0$$

Folding of retimed Biquad filter, N=4

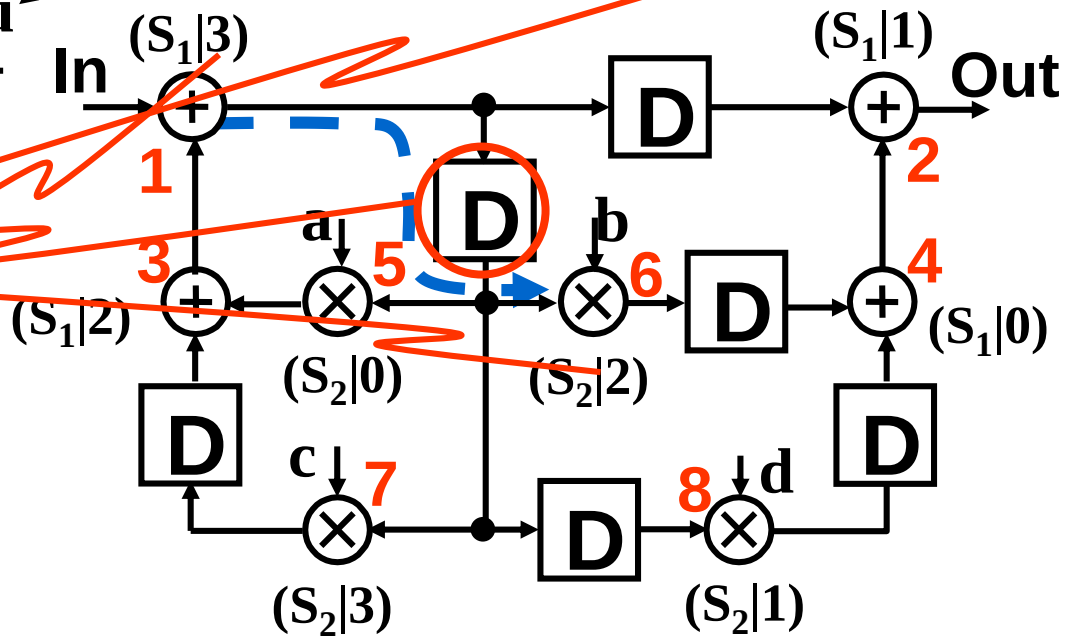
receive

send

$$D_F(U \rightarrow V) = Nw(e) - P_u + v - u$$

$D_F(1 \rightarrow 2) = 4(1) - 1 + 1 - 3 = 1$
$D_F(1 \rightarrow 5) = 4(1) - 1 + 0 - 3 = 0$
$D_F(1 \rightarrow 6) = 4(1) - 1 + 2 - 3 = 2$
$D_F(1 \rightarrow 7) = 4(1) - 1 + 3 - 3 = 3$
$D_F(1 \rightarrow 8) = 4(2) - 1 + 1 - 3 = 5$
$D_F(3 \rightarrow 1) = 4(0) - 1 + 3 - 2 = 0$
$D_F(4 \rightarrow 2) = 4(0) - 1 + 1 - 0 = 0$
$D_F(5 \rightarrow 3) = 4(0) - 2 + 2 - 0 = 0$
$D_F(6 \rightarrow 4) = 4(1) - 2 + 0 - 2 = 0$
$D_F(7 \rightarrow 3) = 4(1) - 2 + 2 - 3 = 1$
$D_F(8 \rightarrow 4) = 4(1) - 2 + 0 - 1 = 1$

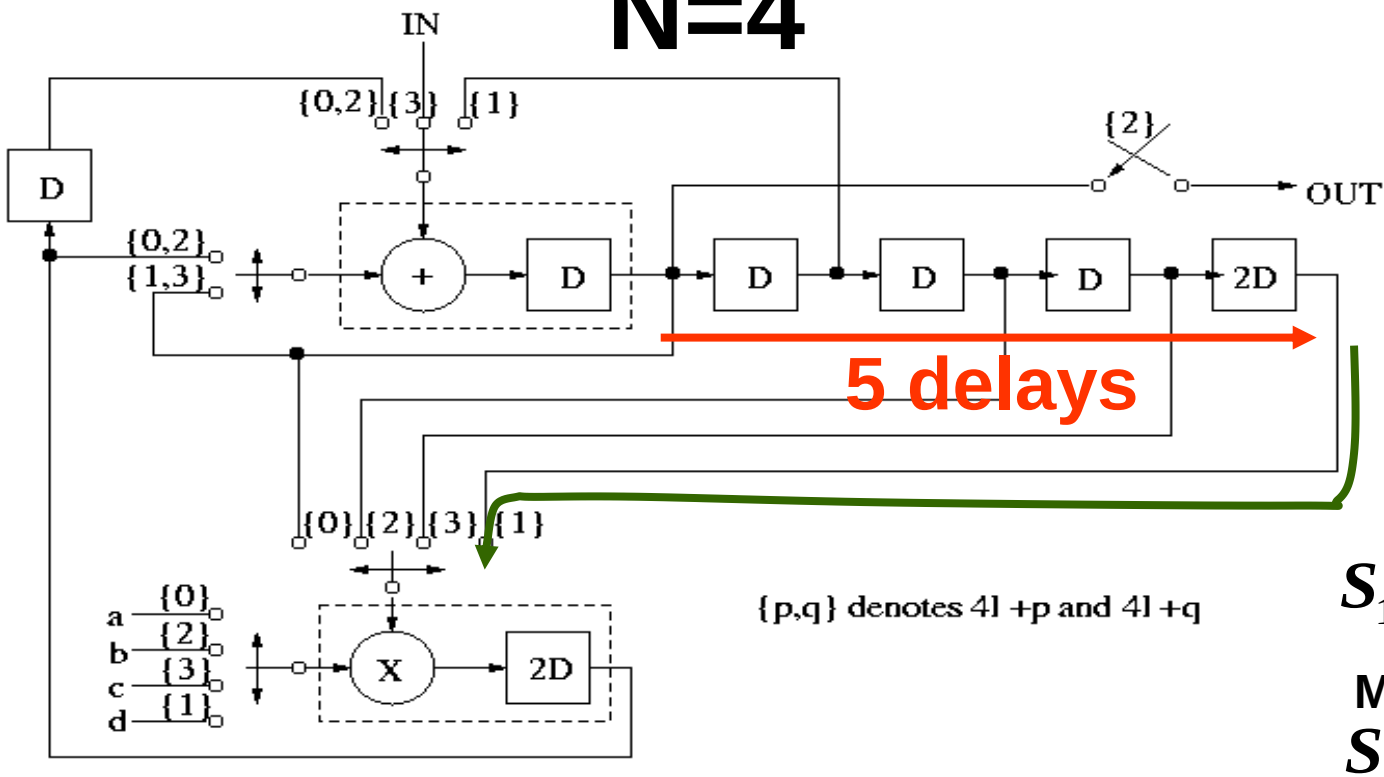
$P_{adder} = 1$



Valid folding

$$D_F(U \rightarrow V) \geq 0$$

Folding of retimed Biquad filter, N=4



Additions

$$S_1 = \{4,2,3,1\}$$

Multiplication

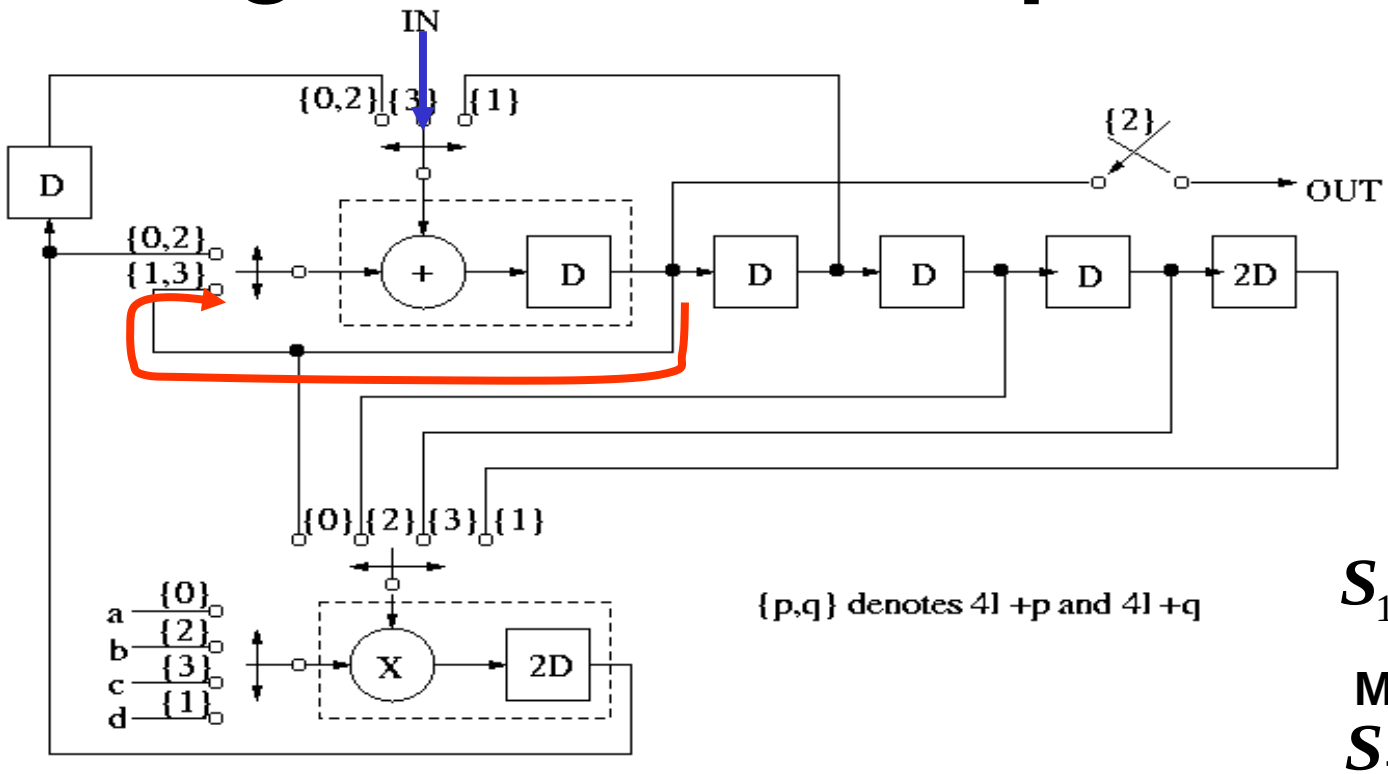
$$S_2 = \{5,8,6,7\}$$

$$D_F(U \rightarrow V) = Nw(e) - P_u + v - u$$

$D_F(1 \rightarrow 8) = 4(2) - 1 + 1 - 3 = 5$ ⇨ path from add to mult with 5D

Node 8 has folding order 1 ⇨ switch close at 1

Folding of retimed Biquad filter



Additions

$$S_1 = \{4,2,3,1\}$$

Multiplication

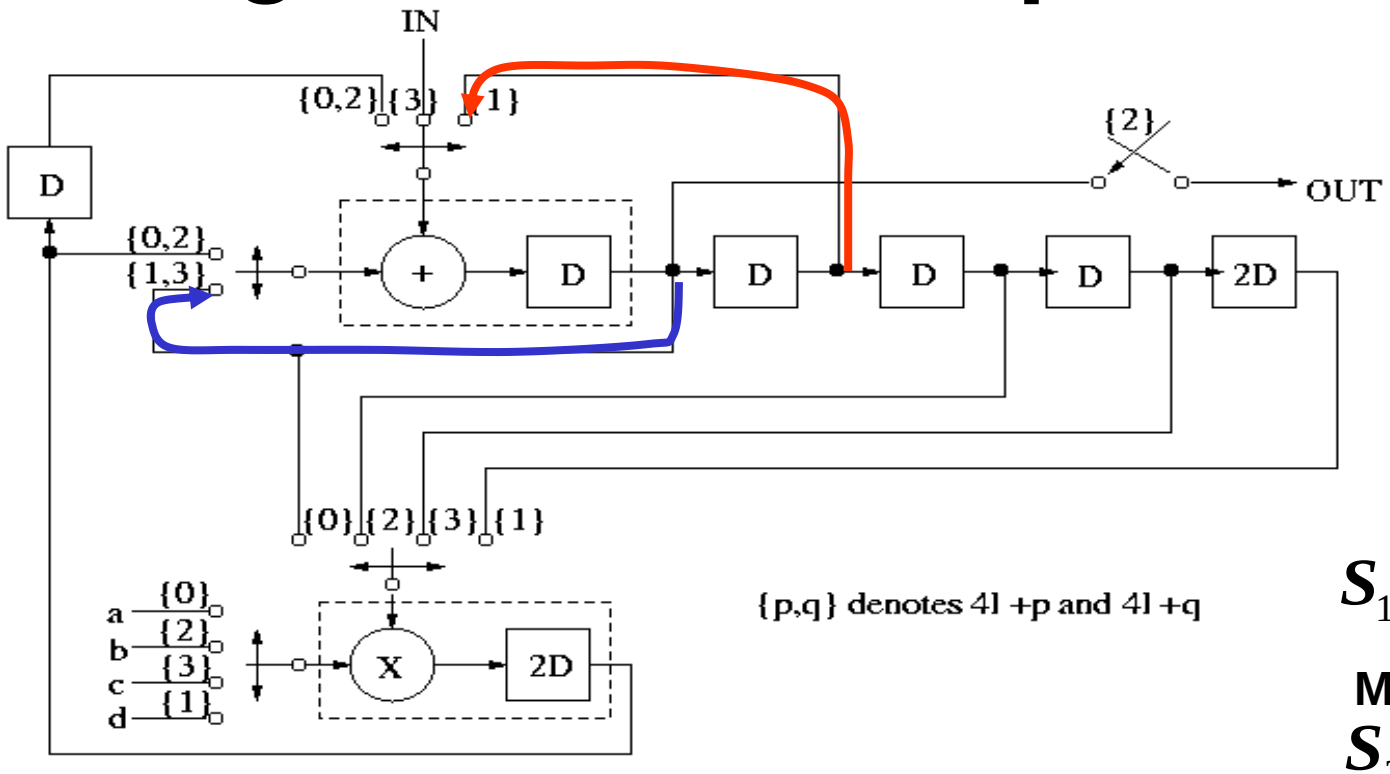
$$S_2 = \{5,8,6,7\}$$

$D_F(3 \rightarrow 1) = 4(0) - 1 + 3 - 2 = 0$ ⇨ path from add to add with 0D

Node 1 has folding order 3 ⇨ switch close at 3 →

Node 1 is also connected to the input →

Folding of retimed Biquad filter



Additions

$$S_1 = \{4,2,3,1\}$$

Multiplication

$$S_2 = \{5,8,6,7\}$$

Execution of node 2 (input from node 1 and 4) :

$D_F(1 \rightarrow 2) = 4(1) - 1 + 1 - 3 = 1$ ⇨ path from add to add with 1D

Node 2 has folding order 1 ⇨ switch close at 1 →

$D_F(4 \rightarrow 2) = 4(0) - 1 + 1 - 0 = 0$ ⇨ path from add to add with 0D →

End of Lecture