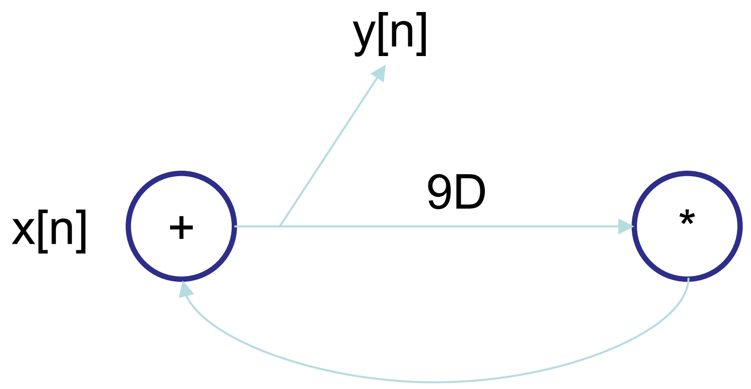
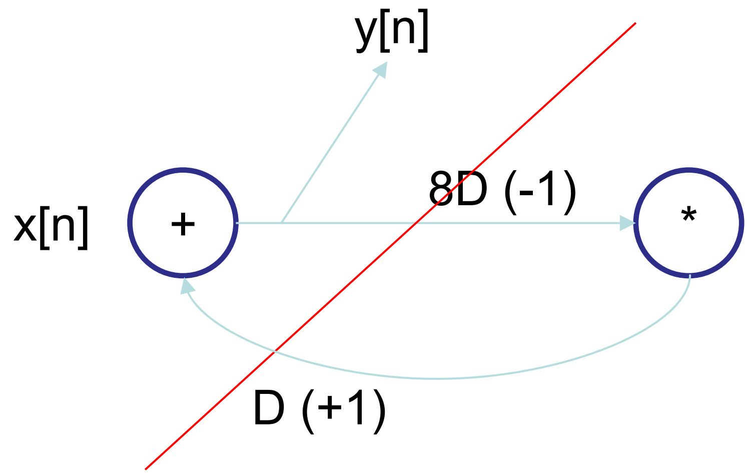
Unfolding

Vassilis Paliouras, Integrated System Design , ECE Dept, U. Patras

Example

$$y[n] = a\, y[n-9] + x[n]$$





$y[n] = a\,y[n-9] + x[n]$

$y[0]\ y[1],\quad y[2]\ y[3],\ y[4]\ y[5],$

$k = \quad 0\ , \qquad\qquad 1 \qquad\qquad ,\ 2,\ \dots$

$y[2k]\ y[2k+1]$

$y[2k] = a\,y[2k-9] + x[2k] = a\,y[2k-10+1] + x[2k]$

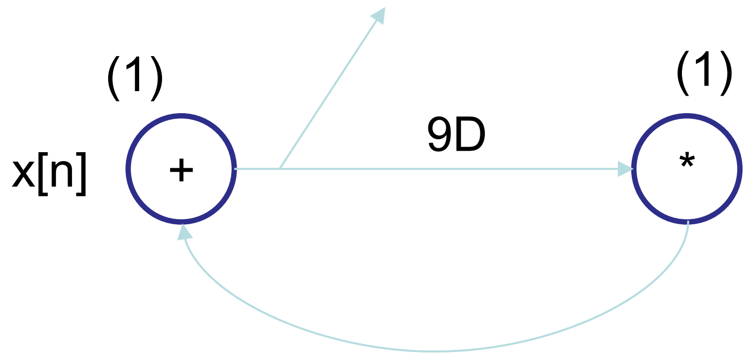
$= a\,y[2(k-5)+1] + x[2k]$

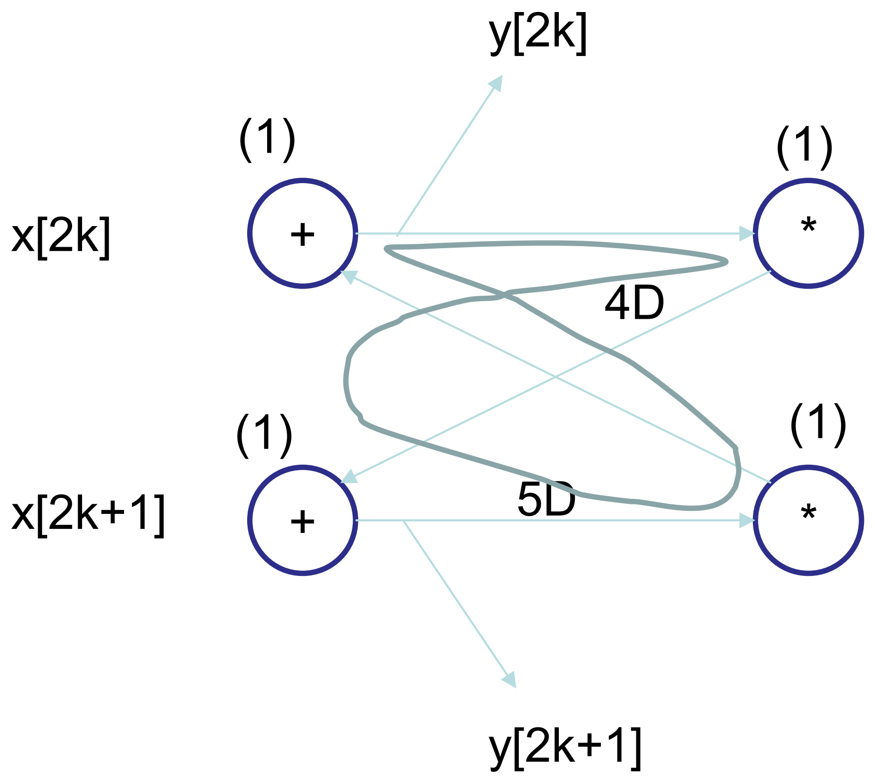
$y[2k+1] = a\,y[2k+1-9] + x[2k+1] = a\,y[2k-8\ ] + x[2k+1] =$

$= a\,y[2(k-4)\ ] + x[2k+1]$

y[n] = a y[n-9] + x[n]

y[2k] = a y[2(k-5)+1] + x[2k]
y[2k+1] = a y[2(k-4)] + x[2k+1]





y[9k] = …

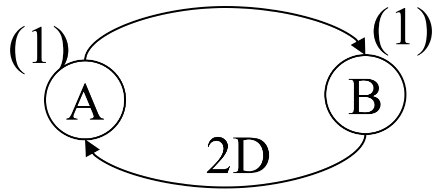
y[9k+1] = …

x[9k+2] =…
…

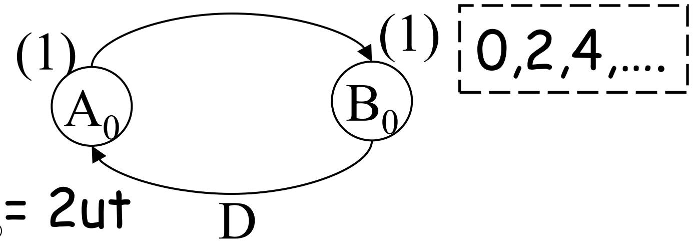
X[9k+8] =…

Unfolding = Parallel Processing

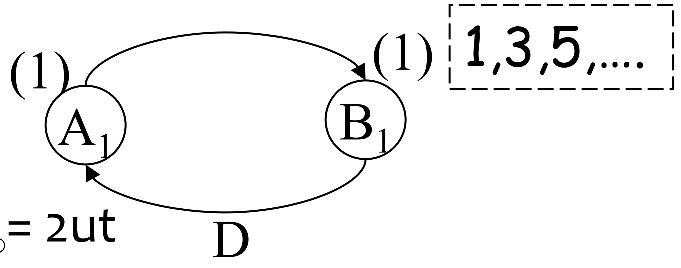
2-unfolded



2 nodes & 2 edges
$T_\infty = (1+1)/2 = 1$ut

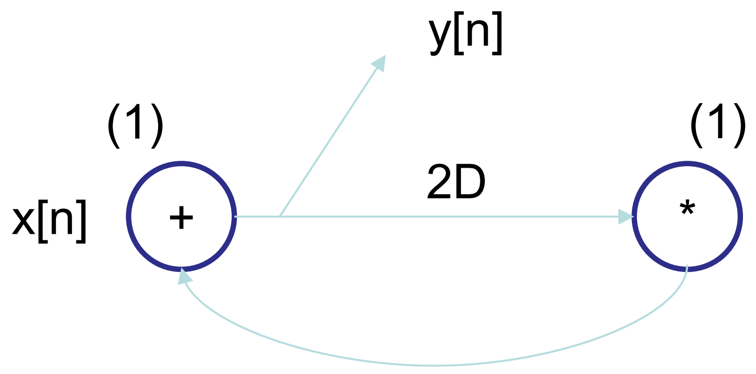


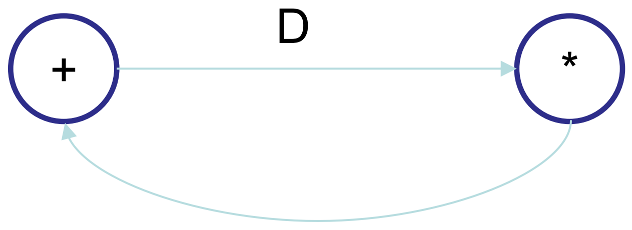
$T'_\infty = 2$ut



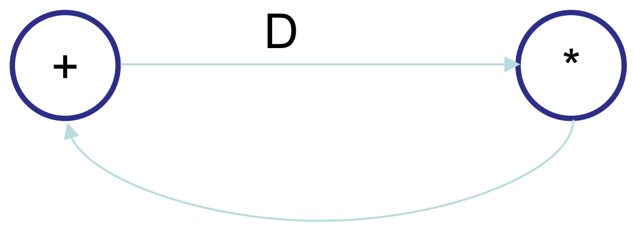
$T'_\infty = 2$ut

- In a '*J*' unfolded system each delay is ***J-slow*** =>
- if input to a delay element is the signal $x[kJ + m]$,
- then the output is $x[(k\text{-}1)J + m] = x[kJ + m - J]$.





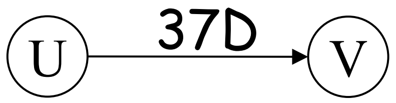
y[2k] = a y[2(k-1)] + x[2k]



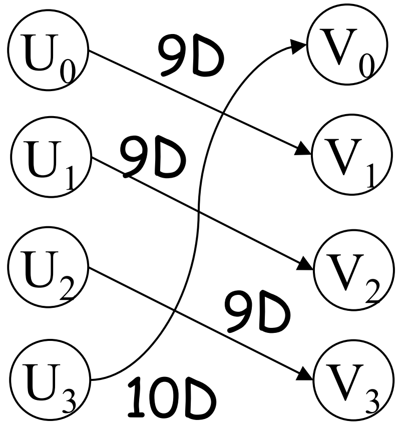
y[2k+1] = a y[2k+1-2] + x[2k+1]
= a y [2(k-1)+1] + x[2k+1]

Algorithm for unfolding

- 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 $\left\lfloor \frac{i+w}{J} \right\rfloor$ delays for $i = 0,1,\dots,J-1$

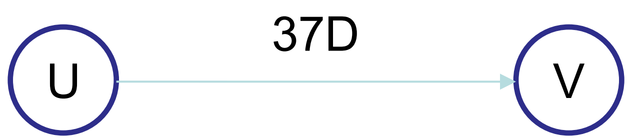


$$w = 37$$
$$\left\lfloor \frac{i+w}{4} \right\rfloor = 9, i = 0,1,2$$
$$\left\lfloor \frac{i+w}{4} \right\rfloor = 10, i = 3$$



- Unfolding of an edge with w $(w < J)$ delays in the original DFG produces
  $J - w$ edges with no delays and
  w edges with 1 delay in the J-unfolded DFG

- Unfolding preserves ***precedence constraints*** of a DSP program.



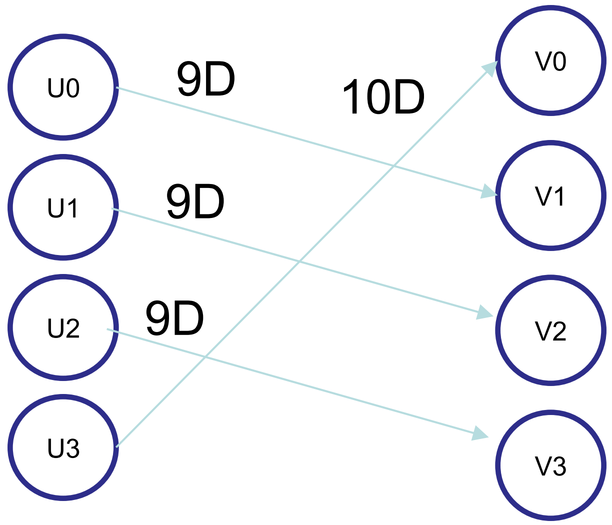
Ui -> V(i+w)%J,   d = floor((i+w)/J)

U0 -> V(0+37)%4 = V1, 9

U1 - > V(1+37)%4 = V2, 9

U2 - > V(3+37)%4 = V3, 9

U3 -> V0, 10



J=4, U1,k, 4k+1 original DFG

V2, 4 k + 4*9 + 2 = 4k + 38 original DFG

- $U_i \rightarrow V_{(i+w)\%J}$, with $\left\lfloor\frac{i+w}{J}\right\rfloor$ delays
- k =0: 0 1, k=1: 2 3, k=2: 4 5,

- $U \rightarrow V$, w delays
- U_i: executes step $J\,k + i$ of original DFG

- $V_{(i+w)\%J}$ executes step

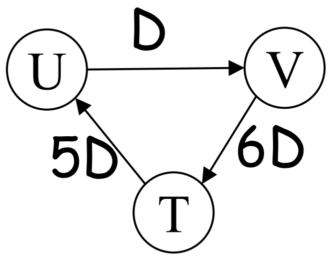
$$J\left(k + \left\lfloor\frac{i+w}{J}\right\rfloor\right) + (i+w)\%J = Jk + J\left\lfloor\frac{i+w}{J}\right\rfloor + (i+w)\%J$$

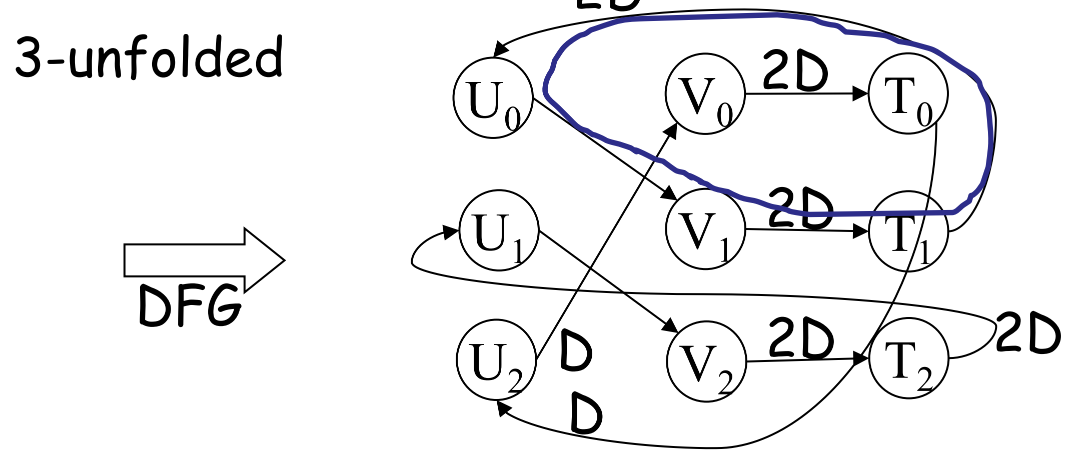
$$= J\,k + i + w$$

- 7 / 3 = 2
- 3 * floor(7/3) + (7 % 3) = 7

Properties of unfolding

floor((i+w)/J), i=0,1,…,J-1

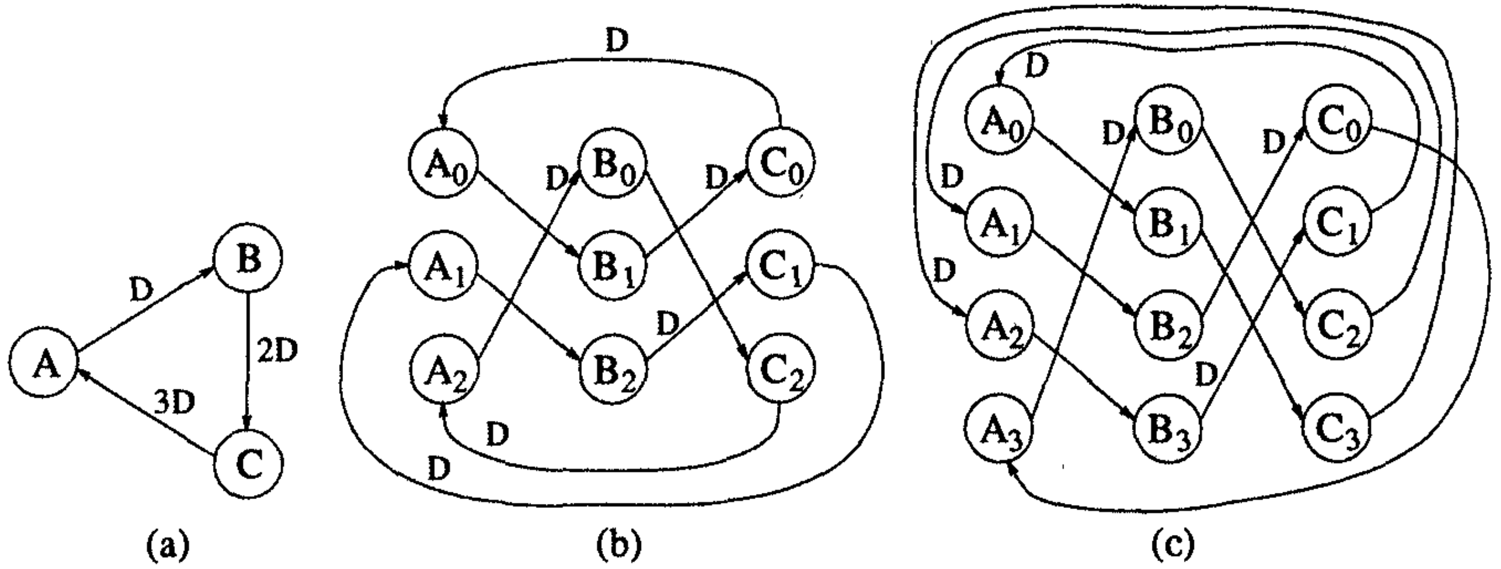




- Unfolding ***preserves*** the number of delays in a DFG.

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

- 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 $\gcd(w_l, J)$ loops contains
  - $\frac{w_l}{\gcd(w_l,J)}$ delays and
  - $\frac{J}{\gcd(w_l,J)}$ copies of each node that appears in l .
- Unfolding a DFG with iteration bound T_∞ results in a J-unfolded DFG with iteration bound JT_∞ .



A loop l gives $\gcd(w_l, J)$ loops

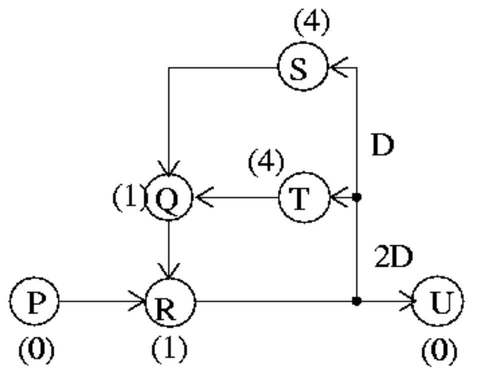
Applications of Unfolding

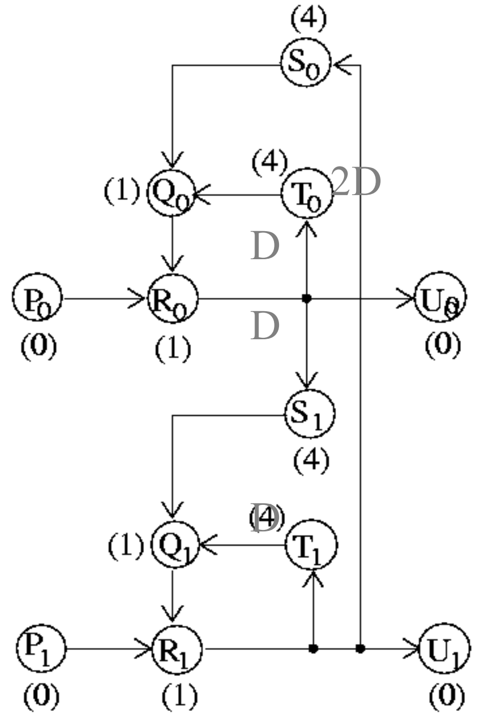
- Sample Period Reduction
- Parallel Processing

- 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 :
    - Longest node computation is larger than the iteration bound T_∞, and
    - T_∞ is not an integer.

Case 1

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

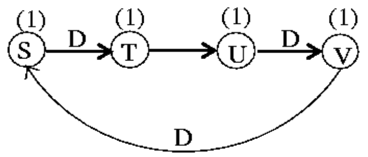


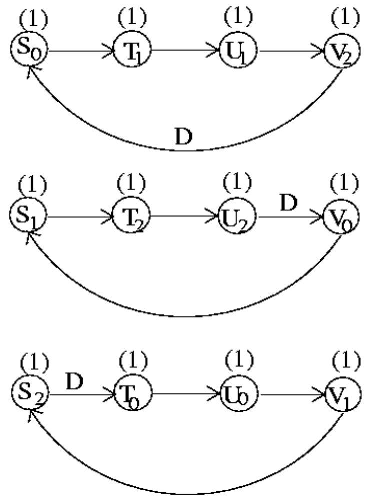


- If the computation time of a node ‘U’, t_u, is greater than the iteration bound T_∞, then $\lceil t_u/T_\infty \rceil$- unfolding should be used.

- In the example, $t_u = 4$, and $T_\infty = 3$, so $\lceil 4/3 \rceil$- unfolding i.e., 2-unfolding is used.

Case 2



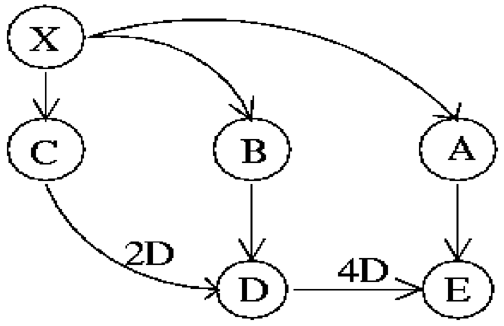


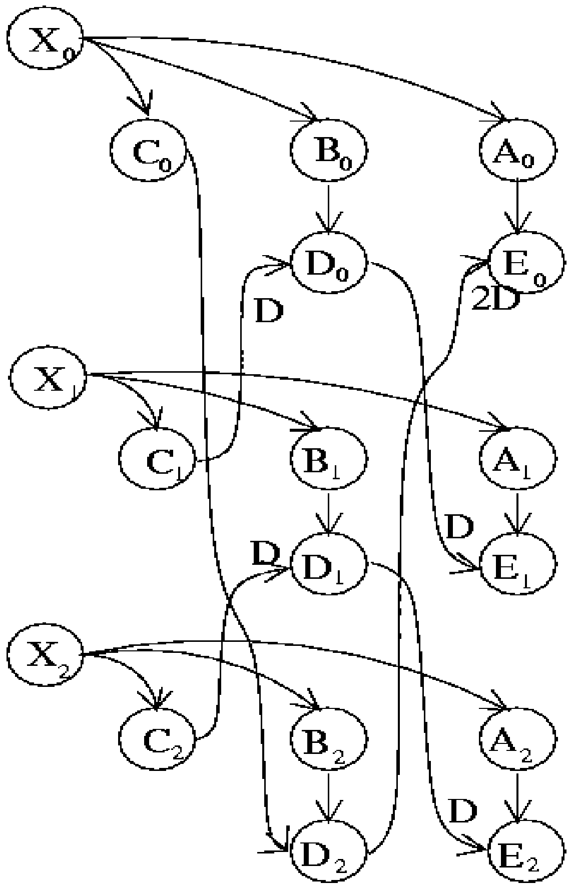
- The original DFG cannot have sample period equal to the iteration bound because the iteration bound is not an integer.

- 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.
- If, for example, $t_l = 60$ and $w_l = 45$, then t_l/w_l should be written as 4/3 and 3-unfolding should be used.

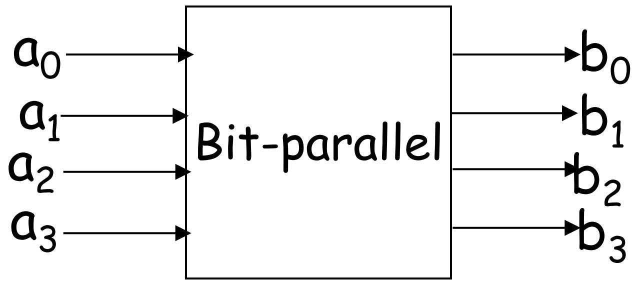
- **Case 3** :
- The minimum unfolding factor that allows the iteration period to equal the iteration bound is the min value of *J*, such that JT_∞ is an integer and is greater than the longest node computation time.

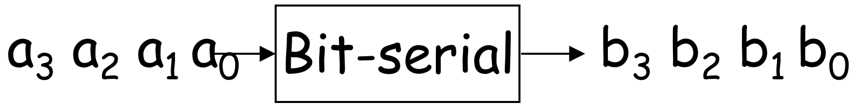
- Unfolding for Parallel Processing:
  - Word- Level Parallel Processing
  - Bit-Level Parallel processing
    - Bit-parallel processing
    - Bit-serial processing
    - Digit-serial processing

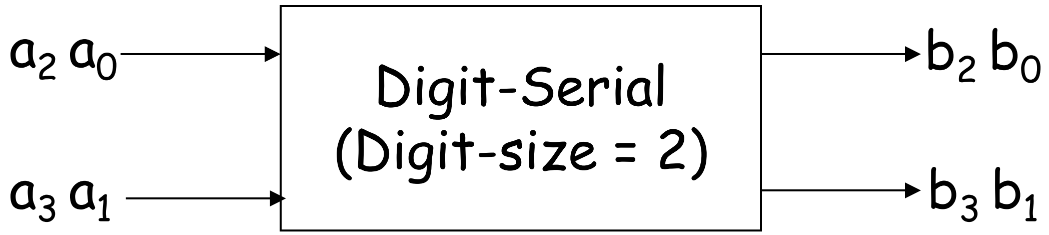


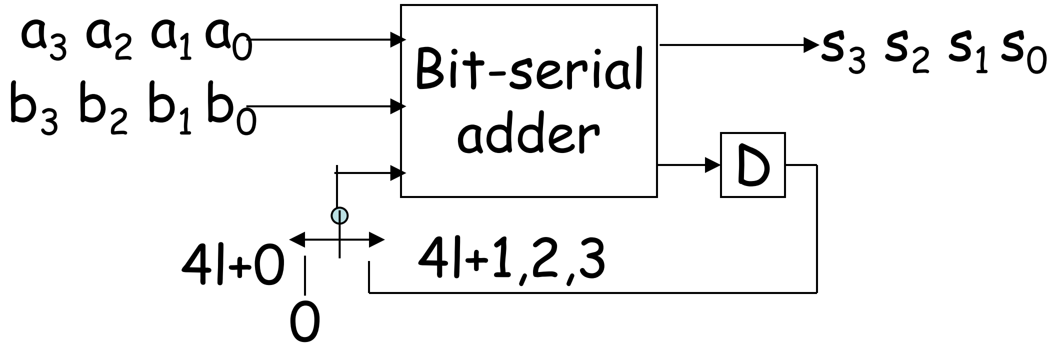


- Bit-Level Parallel Processing





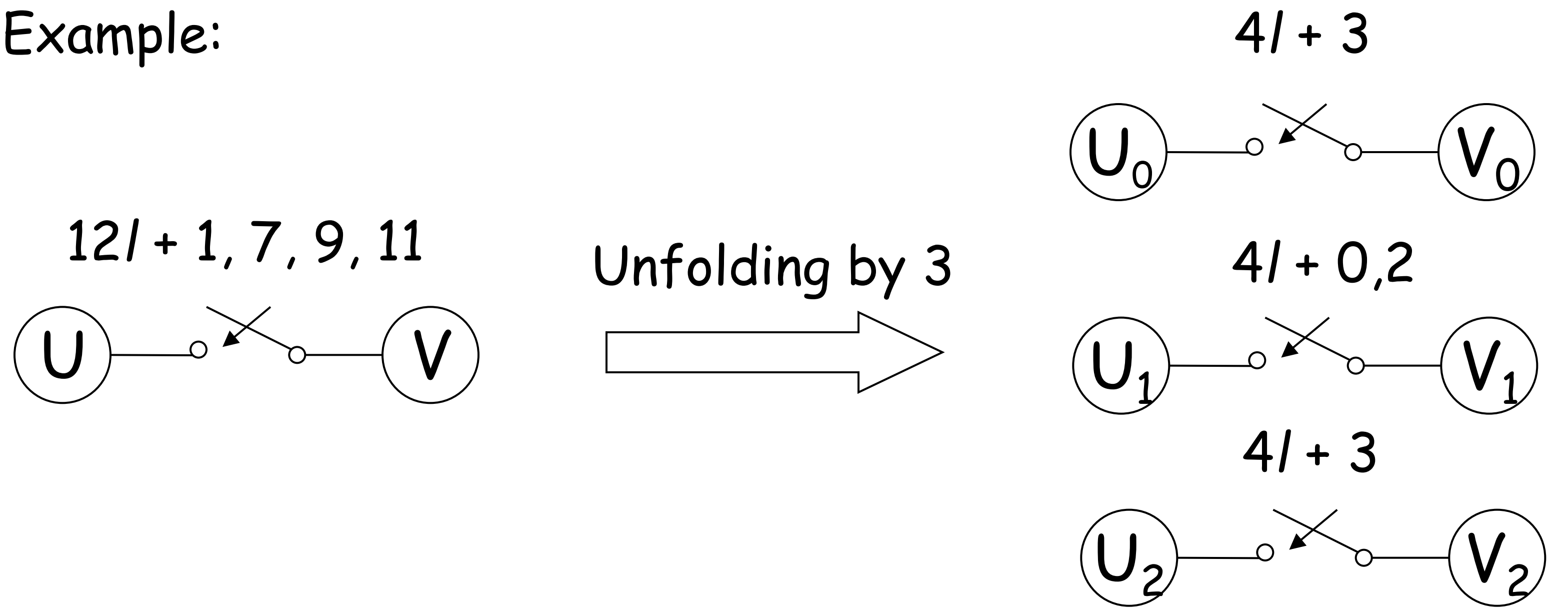




- Assumptions made when unfolding an edge $U \rightarrow V$:
  - The word length W is a multiple of the unfolding factor J, i.e. $W = W'J$.
  - All edges into and out of the switch have no delays.

- With the above two assumptions, an edge $U \rightarrow V$ can be unfolded as follows :
  - Write the switching instance as
    $$Wl + u = J(\ W'l + \lfloor u/J \rfloor\ ) + (u\%J)$$
  - Draw an edge with no delays in the unfolded graph from the node $U_{u\%J}$ to the node $V_{u\%J}$, which is switched at time instance $(\ W'l + \lfloor u/J \rfloor\ )$.

Example:



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

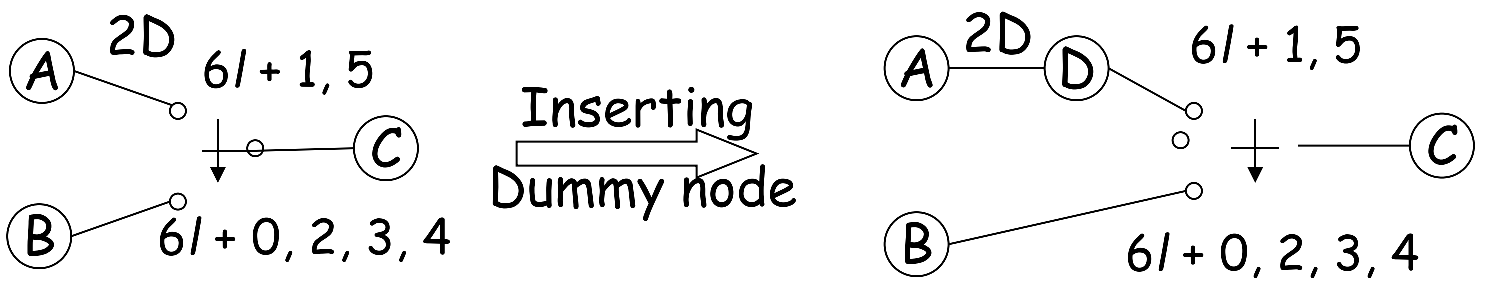
$12l + 1 = 3(4l + 0) + 1$

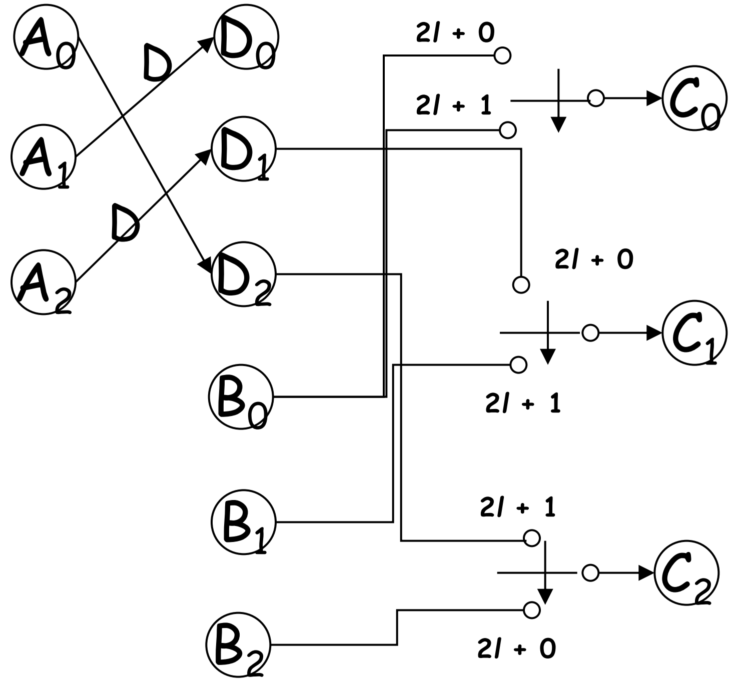
$12l + 7 = 3(4l + 2) + 1$

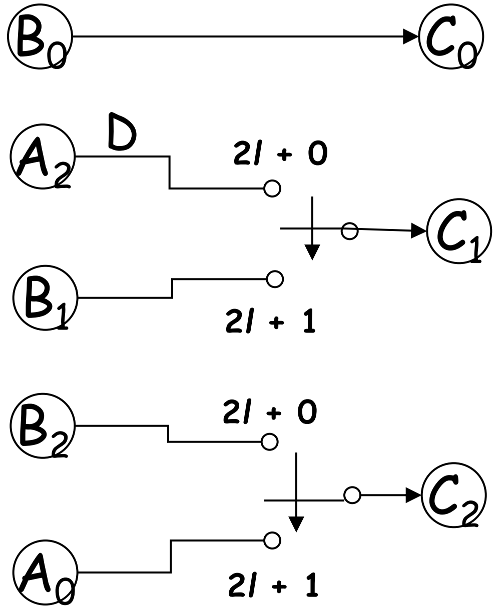
$12l + 9 = 3(4l + 3) + 0$

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

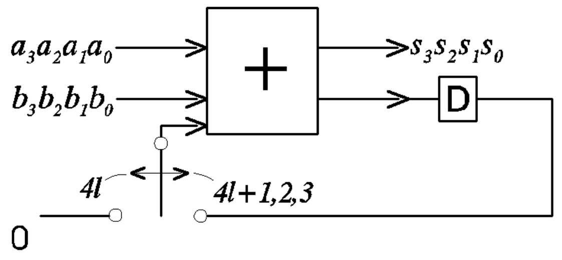
- Unfolding a DFG containing an edge having a switch and a positive number of delays is done by introducing a dummy node.

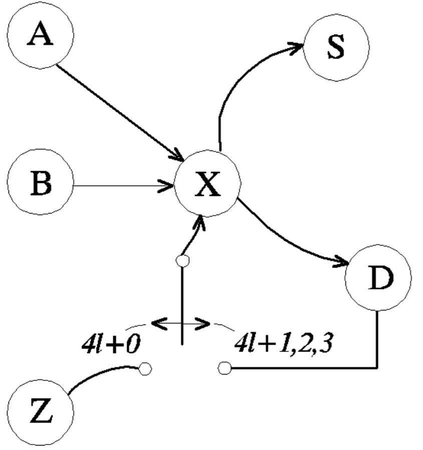




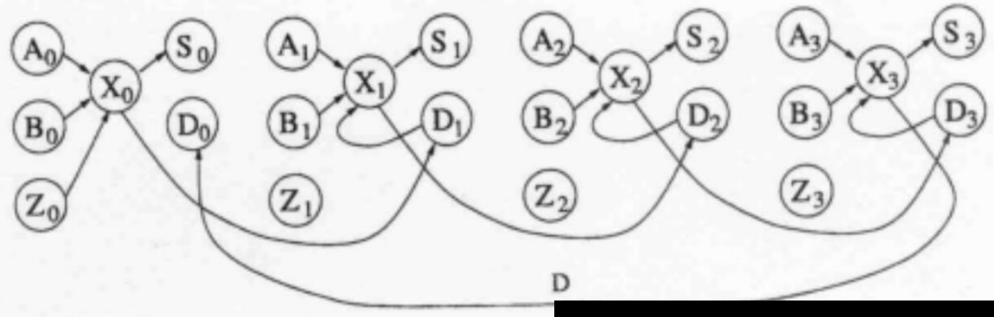


Example: Bit-serial adder

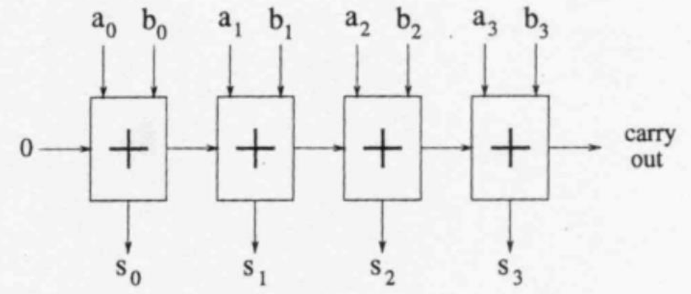




Bit-parallel adder (J=4)

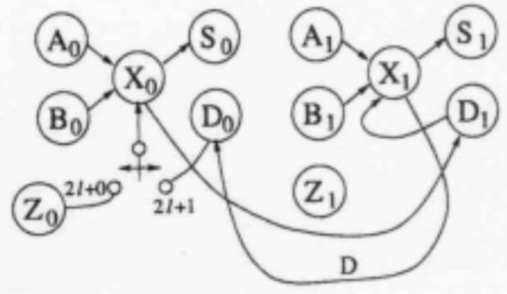

A_0 B_0 Z_0 X_0 S_0 D_0
A_1 B_1 Z_1 X_1 S_1 D_1
A_2 B_2 Z_2 X_2 S_2 D_2
A_3 B_3 Z_3 X_3 S_3 D_3
D

(a)

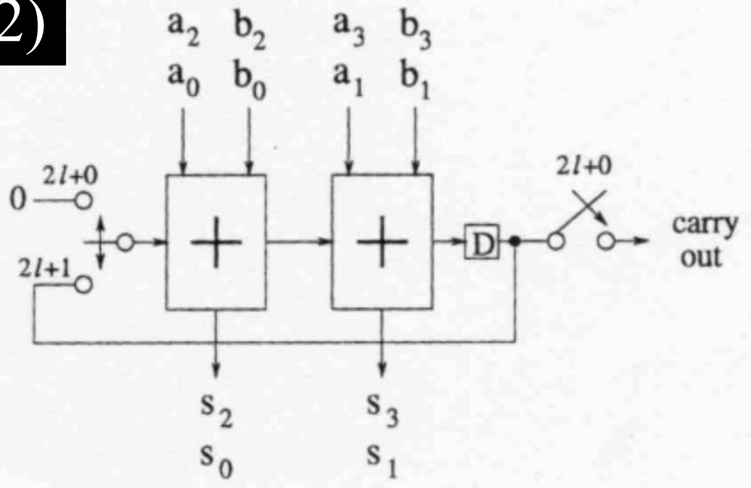


(a)

Digit-serial adder (J=2)


A_0 B_0 X_0 S_0 D_0 Z_0
2l+0
2l+1
A_1 B_1 X_1 S_1 D_1 Z_1
D

(b)



(b)