ELE617E
Lectures

Prof. Dr. Müştak E. Yalçın

Istanbul Technical University

mustak.yalcin@itu.edu.tr

Unfolding

Unfolding is the process of unfolding a loop so that several iterations are unrolled into the same iteration (reveal hidden concurrency).

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

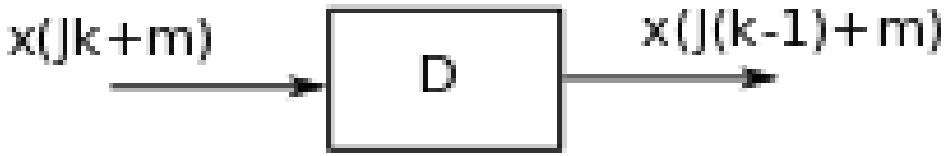
Unfolding factor J creates J consecutive iterations of original program.

Applications:

- Reducing sampling period to achieve iteration bound (desired throughput rate).
- Parallel (block processing) to execute several iterations concurrently.
- Digit-serial or bit-serial processing

Unfolding

In a J Unfolded system, each delay is $J-$slow!



$x(Jk-J+m)$

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

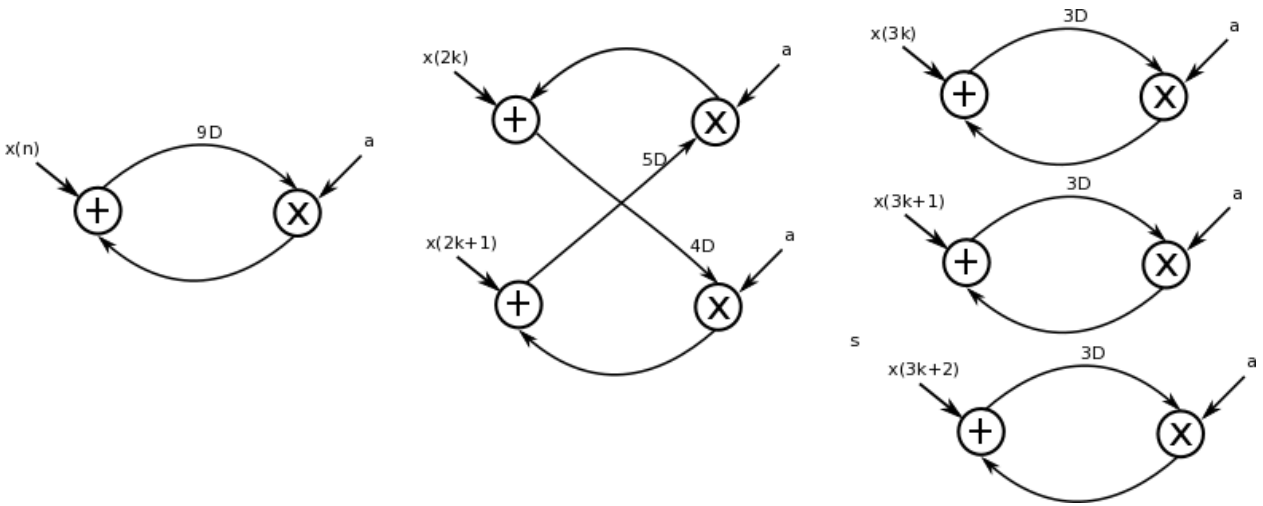
Unfolding factor J creates J consecutive iterations of original program.
$J-$unfolded version of the system ($n = 2k$):

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

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

Unfolding

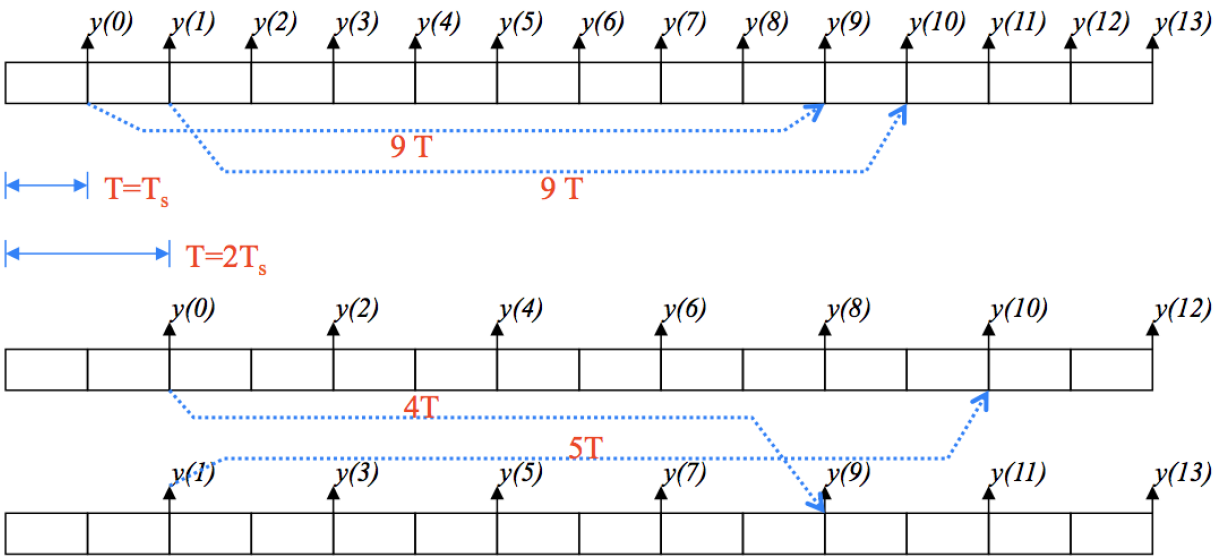
$$y(2k) = ay(2(k-5)+1) + x(2k)$$
$$y(2k+1) = ay(2(k-4)+0) + x(2k+1)$$



$$y(3k) = ay(3(k-3)) + x(3k)$$
$$y(3k+1) = ay(3(k-3)+1) + x(3k+1)$$
$$y(3k+2) = ay(3(k-3)+2) + x(3k+2)$$

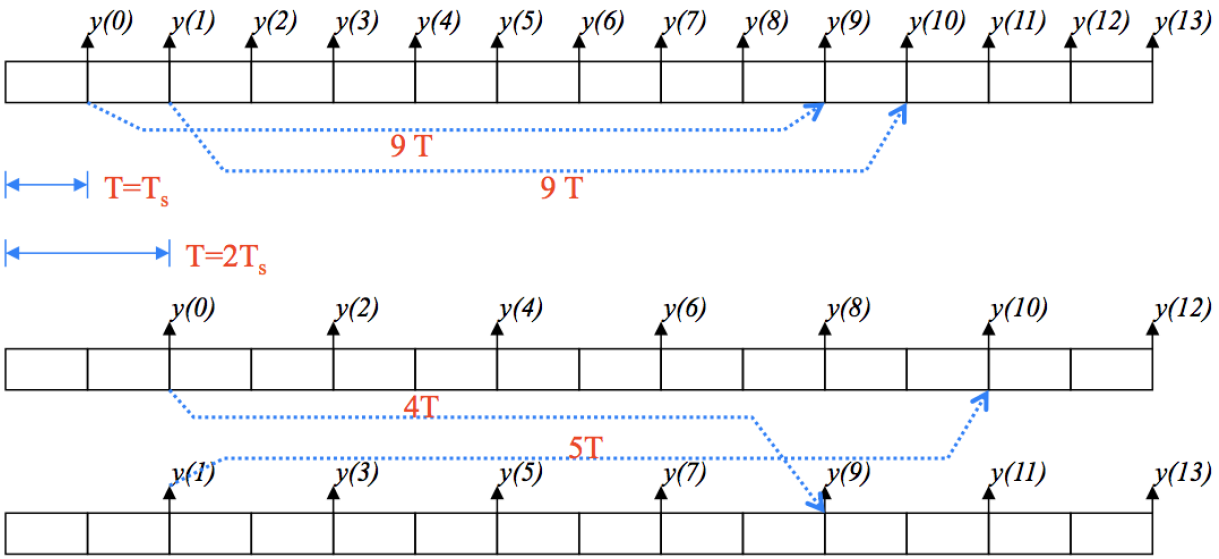
Unfolding

$$y(2k) = ay(2(k-5)+1) + x(2k)$$
$$y(2k+1) = ay(2(k-4)+0) + x(2k+1)$$



Unfolding

$$y(3k) = ay(3(k-3)) + x(3k)$$
$$y(3k+1) = ay(3(k-3)+1) + x(3k+1)$$
$$y(3k+2) = ay(3(k-3)+2) + x(3k+2)$$



Definitions

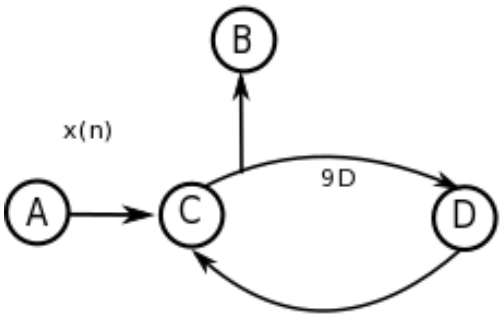
- $\lfloor x \rfloor$ is the floor of x, largest integer $\leq x$
- $\lceil x \rceil$ is the ceiling of x, smallest integer $\geq x$
- $a\%b$ remainder after $\frac{a}{b}$

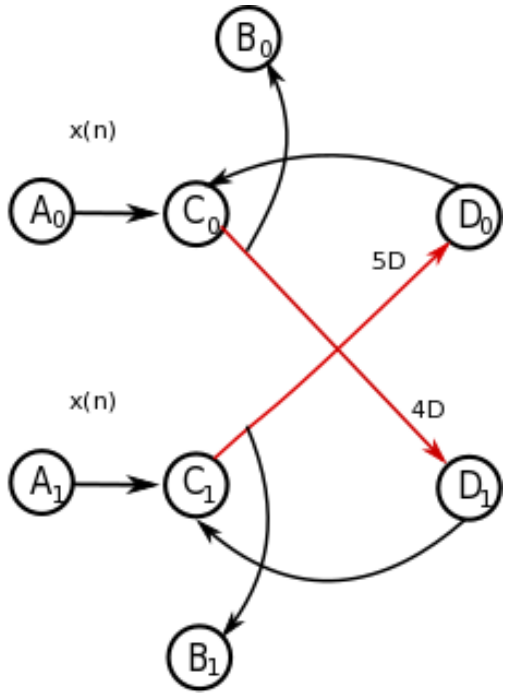
Unfolding Alg.

- For each node U, draw the J nodes $U_0, ..., U_{J-1}$.
- For each node $U \rightarrow V$ with w delays, draw J edges $U_i \rightarrow V_{(i+w)\%J}$ with $\lfloor \frac{i+w}{J} \rfloor$ delays for $i = 0, ...J-1$.

Unfolding Alg.

- For each node U, draw the J nodes $U_0, ..., U_{J-1}$.
- For each node $U \rightarrow V$ with w delays, draw J edges $U_i \rightarrow V_{(i+w)\%J}$ with $\lfloor \frac{i+w}{J} \rfloor$ delays for $i = 0, ...J-1$.





$C_0 \xrightarrow{\lfloor \frac{0+9}{2} \rfloor = 4} D_{(0+9)\%2=1}$ and $C_1 \xrightarrow{\lfloor \frac{1+9}{2} \rfloor = 5} D_{(1+9)\%2=0}$

Examples: Fig. 5.3 and 5.4

Properties of Unfolding

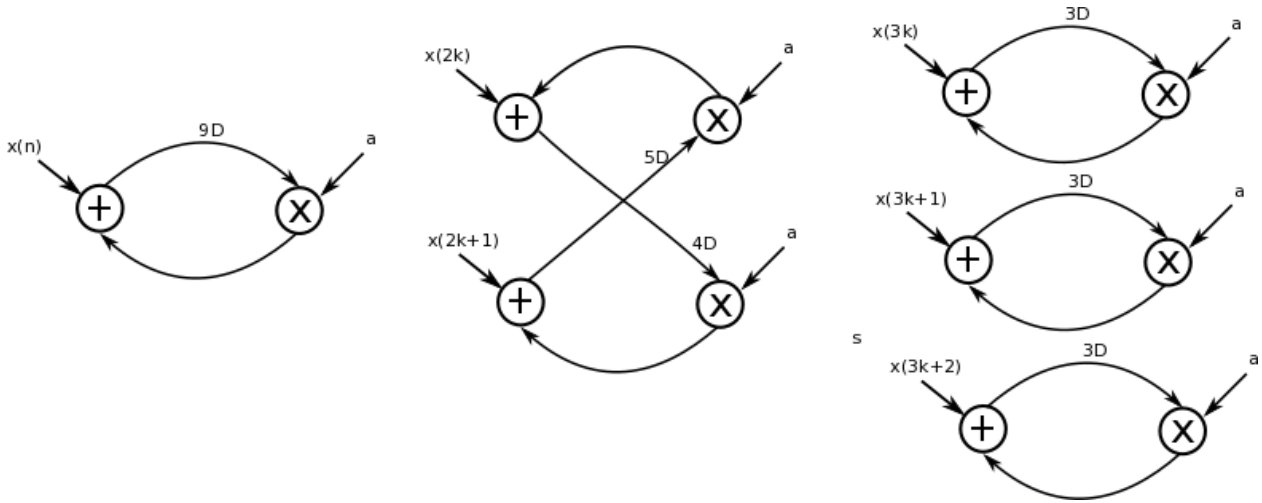
- Unfolding preserves the number of delays in a DFG.

$$\sum_{i=0}^{J-1} \lfloor \frac{i+w}{J} \rfloor = w$$

- Unfolding preserves precedence constraints. $U_i \rightarrow V_{(i+w)\%J}$ with $\lfloor \frac{i+w}{J} \rfloor$ delays in the unfloded DFG corresponds to $U \rightarrow V$ with w delays.
- J-unfolding of a loop with w_k delays in the original DFG a $gcd(w_k, J)$ loops in the unfolded DFG. Each loop contains $w_k/gcd(w_k, J)$ delays and $J/gcd(w_k, J)$ copies of each node.

Properties of Unfolding

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

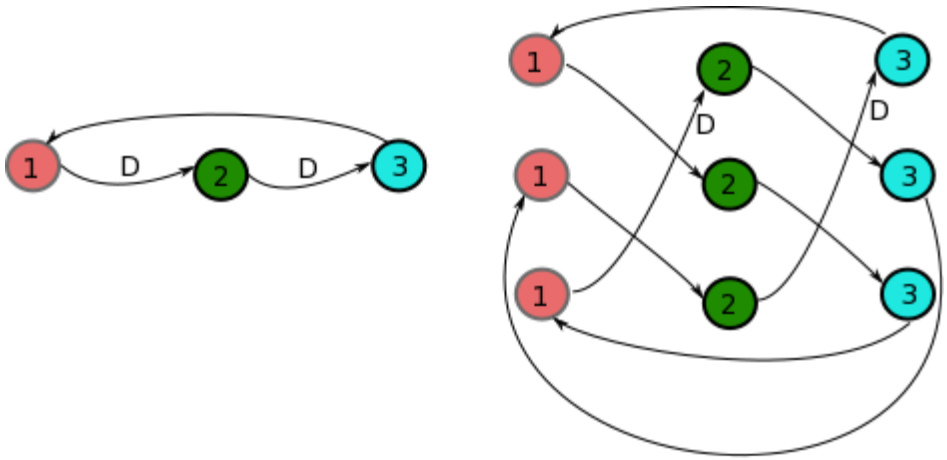


- Unfolding a DFG with iteration bound T_∞ results in a J-unfolded DFG with iteration bound JT_∞.

$$T_\infty = \frac{T_M + T_A}{9} \quad T_{\infty,2} = \frac{2(T_M + T_A)}{9} \quad T_{\infty,3} = \frac{(T_M + T_A)}{3}$$

Unfolding & The Critical Path

- if edge with $w < J$ then $(J - w)$ paths with zero delay and w paths with 1 delay.



Can lead to increased **critical path**!

- if the edge $w \geq J$: no new critical path.

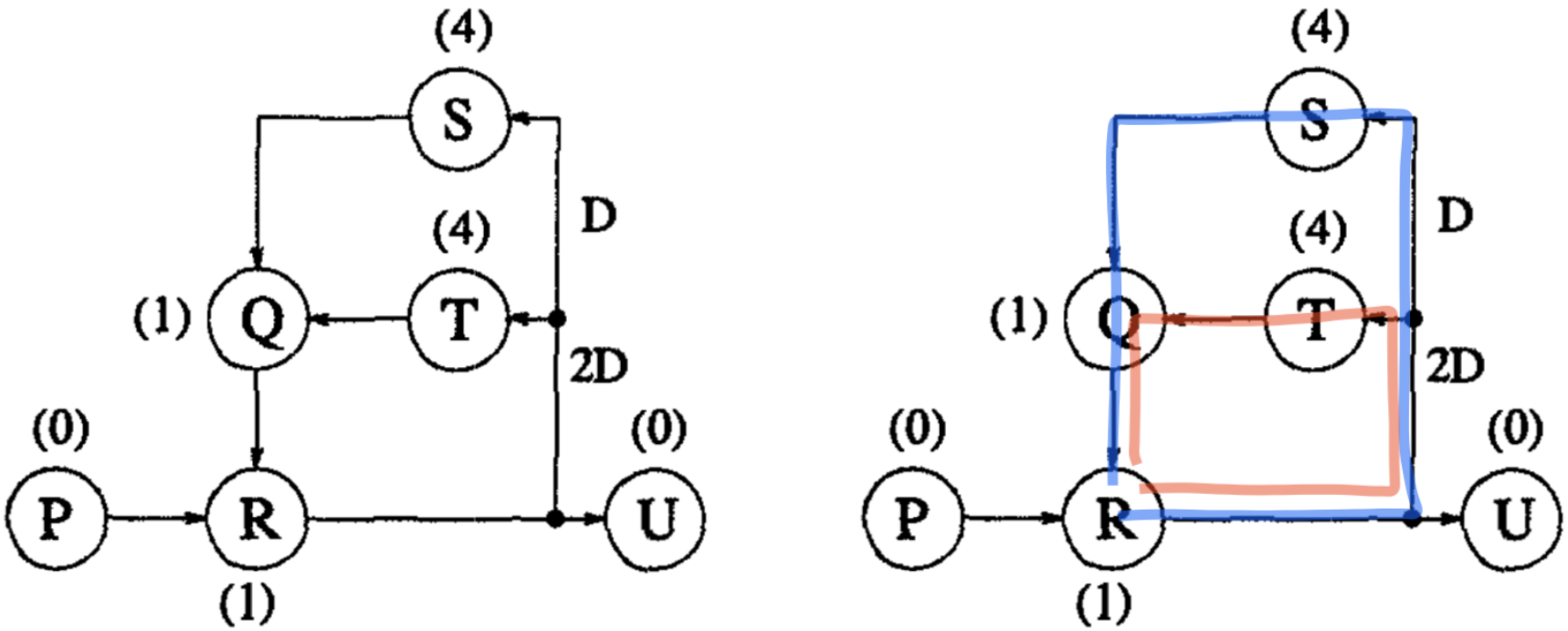
Sample Period Reduction

T_∞ is the lower bound on ▸ the iteration period of a recursive DSP program. Without unfolding: DSP program cannot imp. with iteration period equal to T_∞ (in same case).

- Case1: A node in the DFG having computation time greater than T_∞
- Case2: Iteration bound is not an integer
- Case3: Longest node computation is larger than the iteration T_∞, and T_∞ is not an integer

Sample Period Reduction

- Case1: A node in the DFG having computation time greater than T_∞

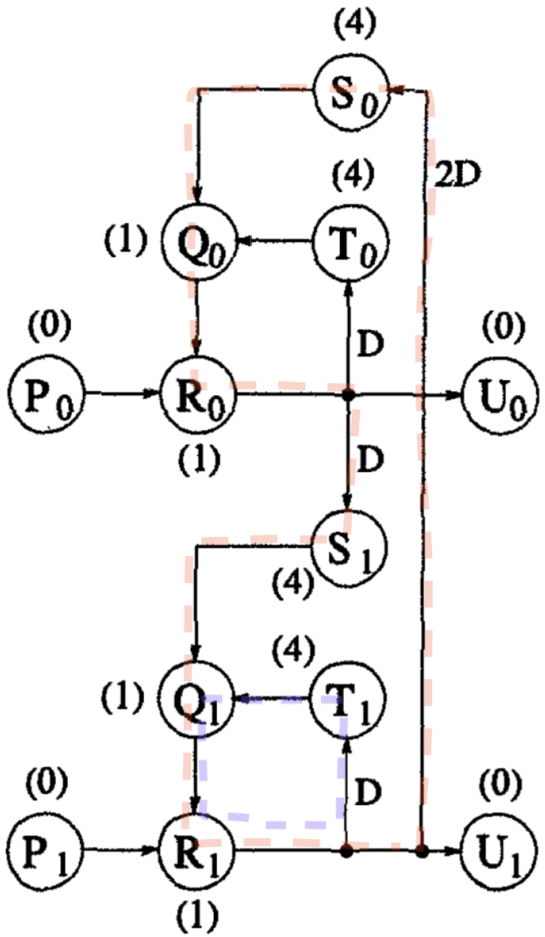


$$T_\infty = \max\{\frac{6}{2}, \frac{6}{3}\} \text{ and } d_{max} = 4 \text{ then } T_{\text{critical}} = 6$$

$$\text{Hence } T_\infty \neq T_s, \text{ unfolled } \lceil \frac{d_{max}}{T_\infty} \rceil \text{ unfolding}$$

Sample Period Reduction

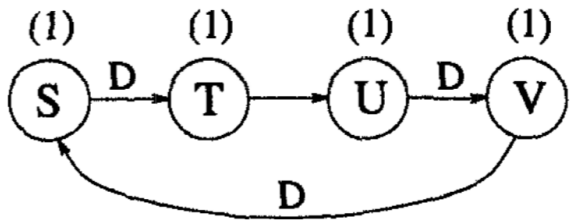
- Case1: A node in the DFG having computation time greater than T_∞



$T_\infty = \max\{\frac{6}{1}, \frac{12}{3}\}$ and $d_{max} = 4$ then $T_s = 6/J = 3$

Sample Period Reduction

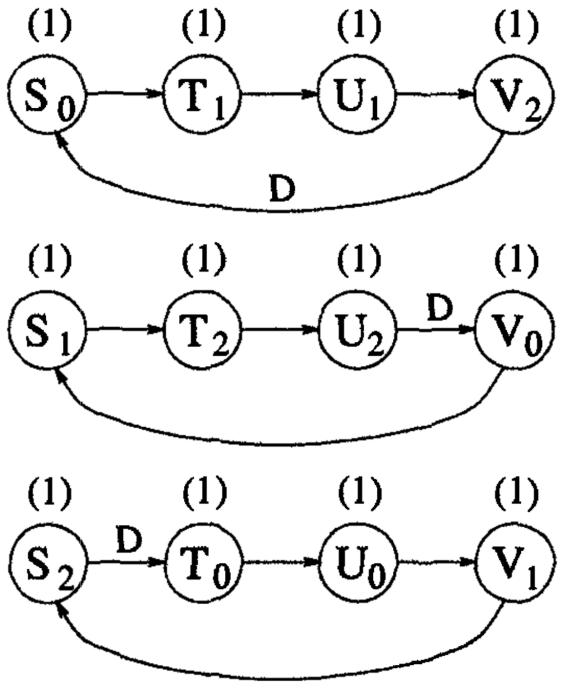
- Case2: Iteration bound is not an integer: if a critical loop is $\frac{d_l}{w_l}$ where d_l and w_l are mutually coprime, then w_l-unfolding should be used.



$$T_\infty = \max\{\frac{4}{3}\} \notin \mathbf{Z} \text{ and } T_{crt} = 2 \text{ then } T_s = 2$$
$$J = 3 \text{ unfolding should be used...}$$

Sample Period Reduction

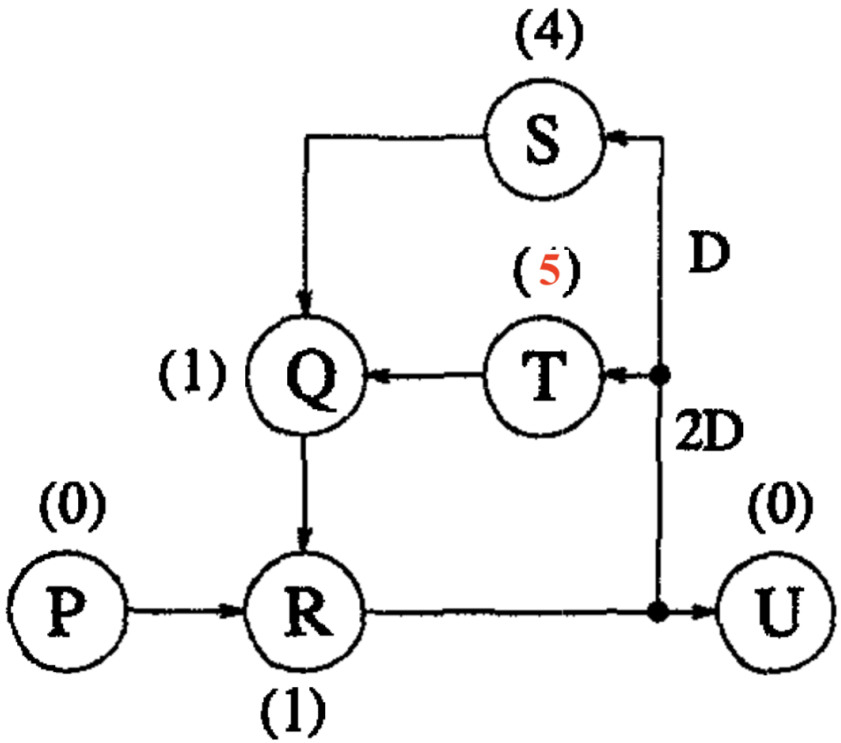
- Case2: Iteration bound is not an integer



$$T_\infty = \max\{\frac{4}{1}\} \text{ and } T_{crt} = 4 \text{ then } T_s = 4/3$$

Sample Period Reduction

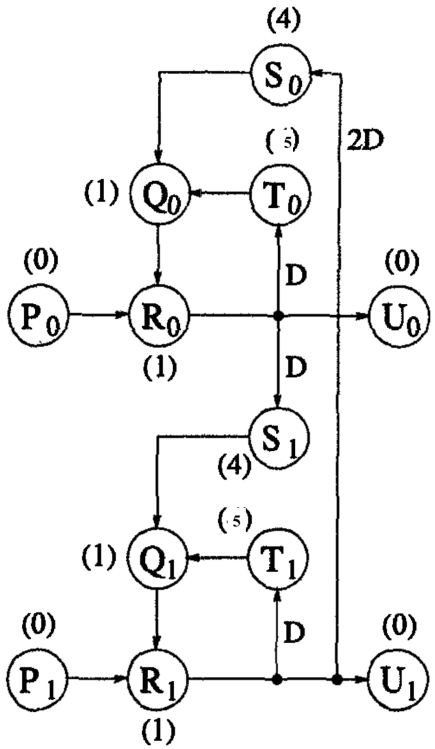
- Case3: Longest node computation is larger than the iteration T_∞, and T_∞ is not an integer



$$T_\infty = \max\{\frac{7}{2}, \frac{6}{3}\} \notin \mathbf{Z} \text{ and } T_{crt} = 7 > T_\infty,$$

Sample Period Reduction

- Case3: Longest node computation is larger than the iteration T_∞, and T_∞ is not an integer : the min. unfolding factor J such that $J \times T_\infty \in \mathbf{Z}$



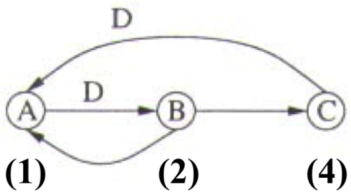
$$T_\infty = \max\{\frac{7}{1}, \frac{12}{3}\} \in \mathbf{Z} \text{ and } T_{crt} = 7 = T_\infty, \text{ then } T_s = 7/2$$

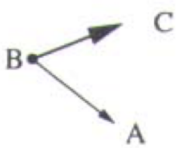
perfect-rate DFG

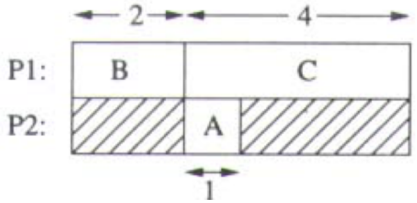
- perfect-rate DFG: the iteration periode is equal to the iteration bound, and a DFG is not perfect-rate, it can ve always be unfolded so the unfolded is perfect rate.
- Choosing J to be equal to the least common multiple of the number of loop delays guarantees that the unfolded DFG is perfect rate (not necessarily...).

Sample Period Reduction

- Unfolding can be exploited to reduce the iteration period in programmable DSPs.
- If the period of a periodic schedule for a J-unfolded DFG is T, then the average iteration period is T/J.
- In a non-overlapped schedule, the period of the schedule is the same as the critical path of the acyclic precedence graph.





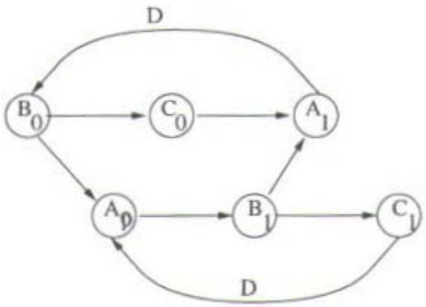


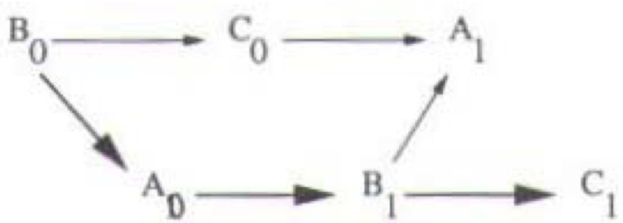
Periodic non-overlapped schedule with a period of 6 u.t.

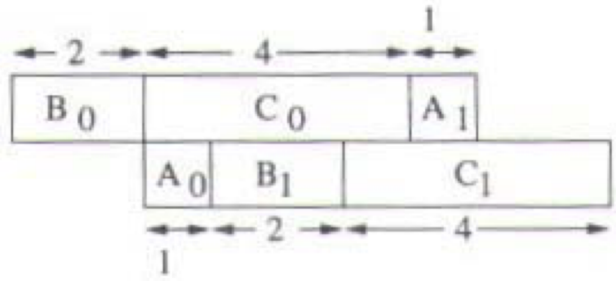
$$T_\infty = \max\{\frac{3}{1}, \frac{7}{2}\} \text{ and } T_{crt} = 6, \text{ then } T_s = 6$$

Sample Period Reduction

- Unfolding can be exploited to reduce the iteration period in programmable DSPs.
- If the period of a periodic schedule for a J-unfolded DFG is T, then the average iteration period is T/J.
- In a overlapped schedule, the period of the schedule is the smaller than the critical path of the acyclic precedence graph.



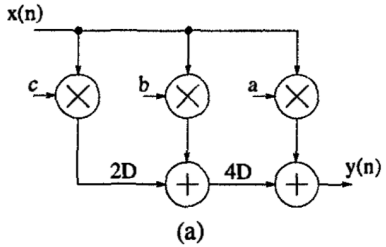




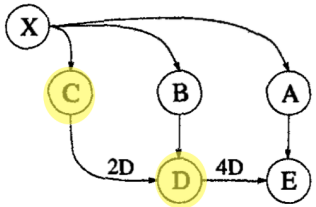
$$T_\infty = \max\{\frac{7}{2}, \frac{6}{2}, \frac{7}{2}\} \text{ and } T_{crt} = 7, \text{ then } T_s = 7/2$$

Parallel Processing

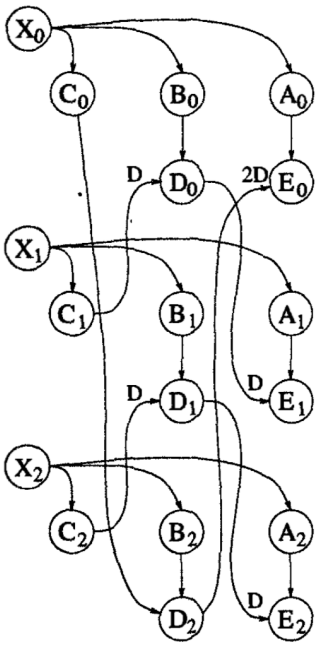
- Word-Level Parallel Processing
- Bit-Level Parallel Processing



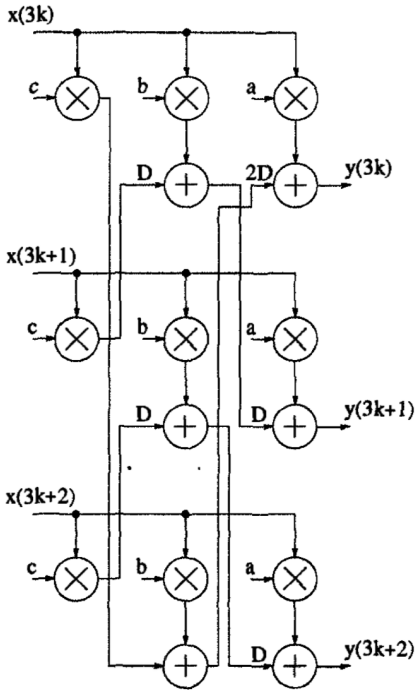
(a)



(b)



(c)



(d)

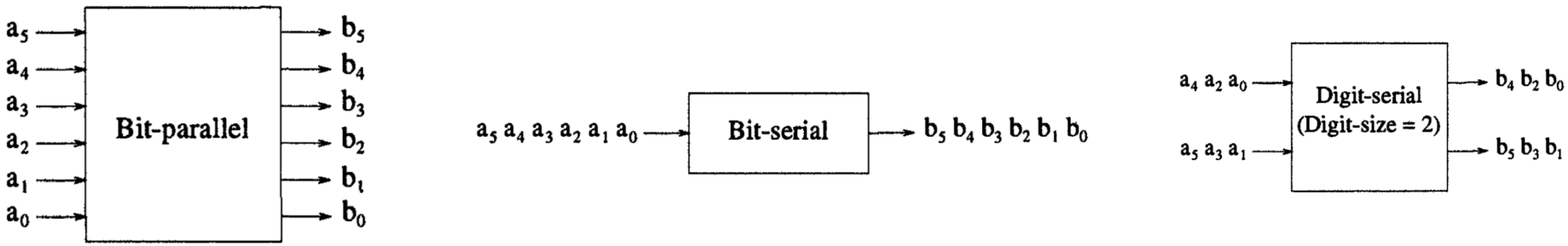
$$C_0 \xrightarrow{\lfloor \frac{0+2}{3} \rfloor = 0} D_{(0+2)\%3=2}$$

$$C_1 \xrightarrow{\lfloor \frac{1+2}{3} \rfloor = 1} D_{(1+2)\%3=0}$$

$$C_2 \xrightarrow{\lfloor \frac{2+2}{3} \rfloor = 1} D_{(2+2)\%3=1}$$

Parallel Processing

(a) Bit-parallel processing, (b) Bit-serial processing,(c) Digit-serial processing.

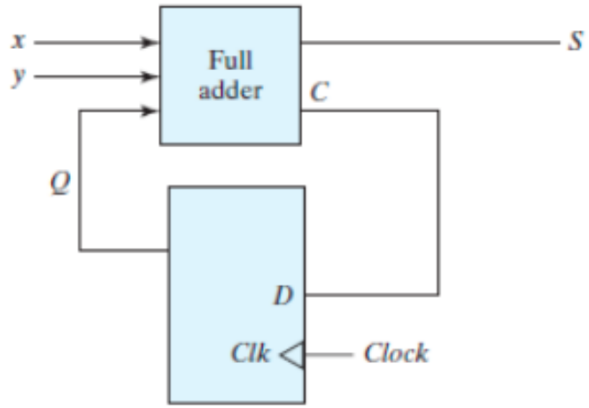


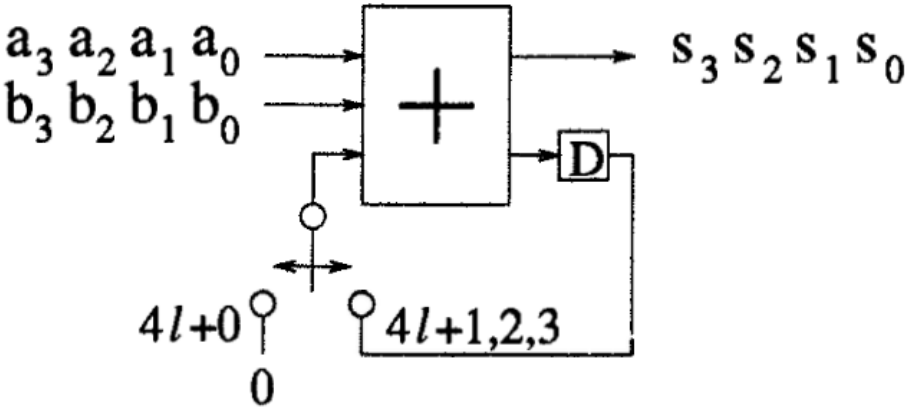
Bit-Level Parallel Processing

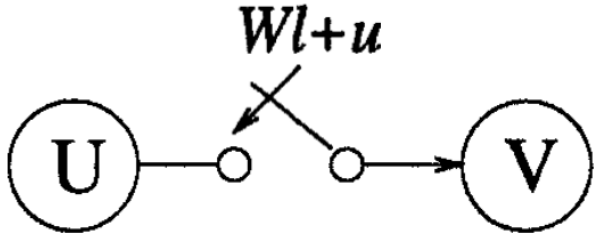
- Bit level parallel processing can increase the speed (reduce the sample time) be processing more than one bit at a time.
- Digit serial parallel processing is when we process W bits at a time where W is the word length.
- Usually involves switching (multiplexers)

Bit Serial Adder & Edges with Switches

Bit serial adder architecture for worldlength 4:







edge with switch

Unfolding An Edge with Switch

Assumptions

- The word-length W is a multiple of the unfolding factor J, i.e., $W = W'J$
- All edges into and out of the switches have no delays

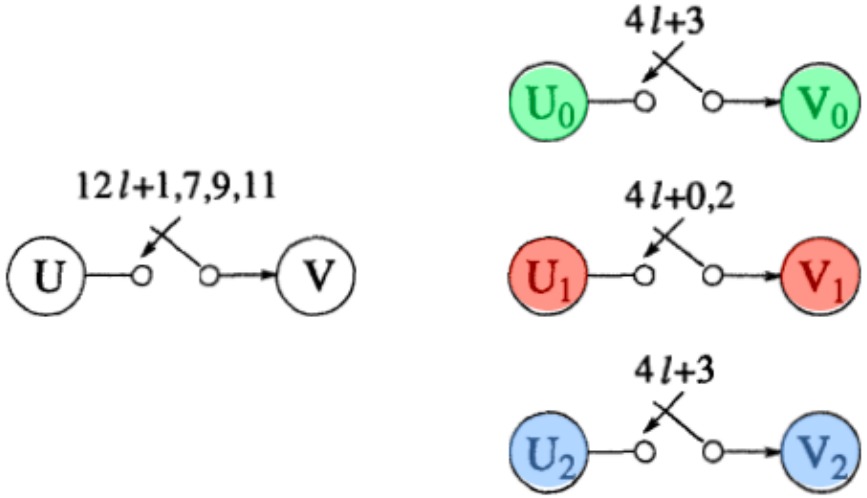
Unfolding an edge with switch

- Step1: Write the switching instance as

$$Wl + u = J\underbrace{(W'l + \lfloor \frac{u}{J} \rfloor)}_{\text{switch instance}} + \underbrace{(u\%J)}_{\text{edge}}$$

- Step2: 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 $(W'l + \lfloor \frac{u}{J} \rfloor)$.

Edges with Switches



$$12l+1 = 3(4l + \lfloor \frac{1}{3} \rfloor) + (1\%3) = 3(4l+0) + 1$$

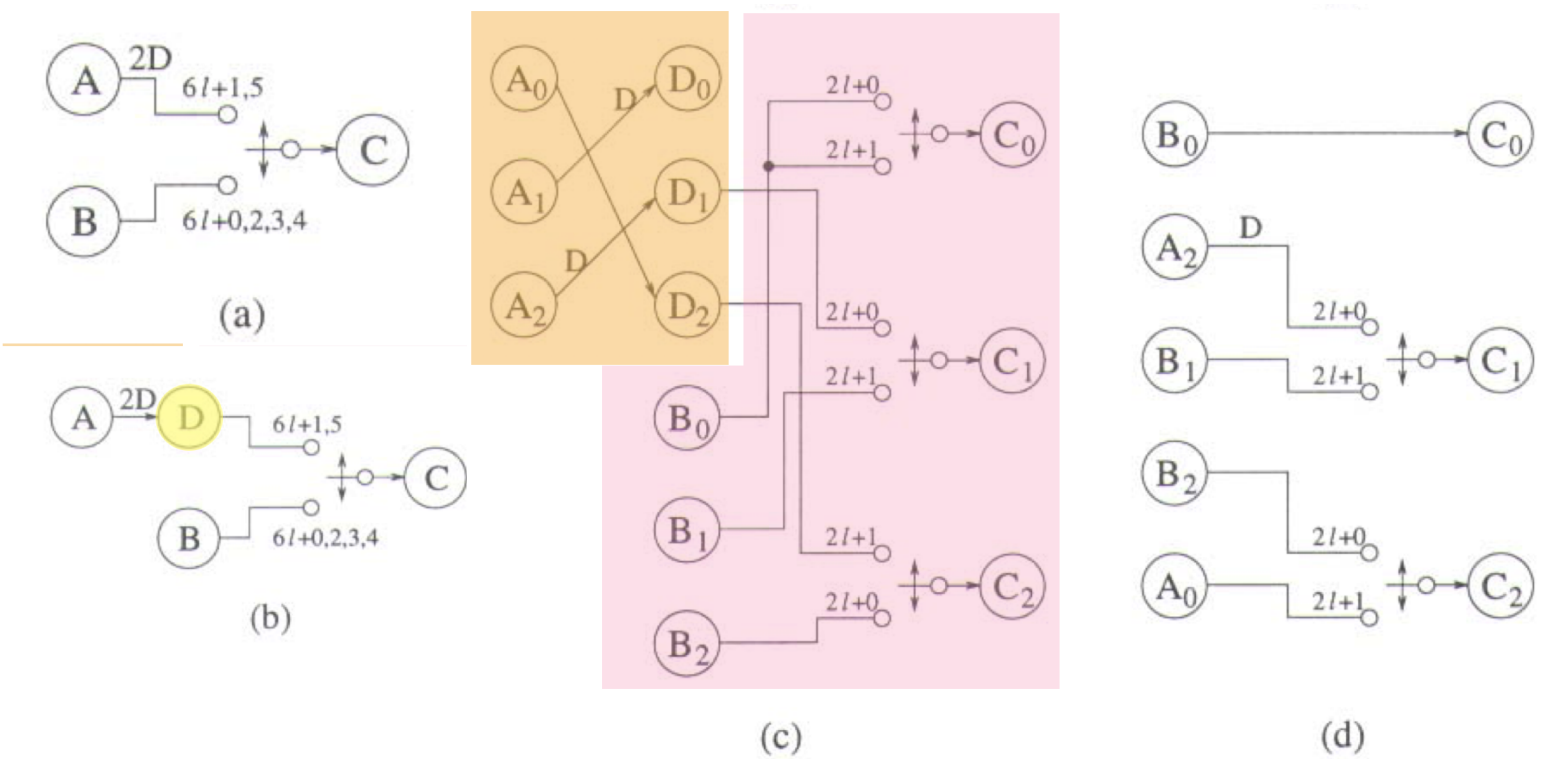
$$12l+7 = 3(4l + \lfloor \frac{7}{3} \rfloor) + (7\%3) = 3(4l+2) + 1$$

$$12l+9 = 3(4l + \lfloor \frac{9}{3} \rfloor) + (9\%3) = 3(4l+3) + 0$$

$$12l+11 = 3(4l + \lfloor \frac{11}{3} \rfloor) + (11\%3) = 3(4l+3) + 2$$

Edges with Switches

- If an edge contains a switch and a positive number of delays, a dummy node can be used to reduce this problem to the case where the edge contains zero delays.
- It may be noted that the number of delays is not preserved by unfolding of circuits containing switches.



Example:

Using these techniques, obtain 4-unfolded and 2-unfolded version of bit-serial addition. Page 137-140.