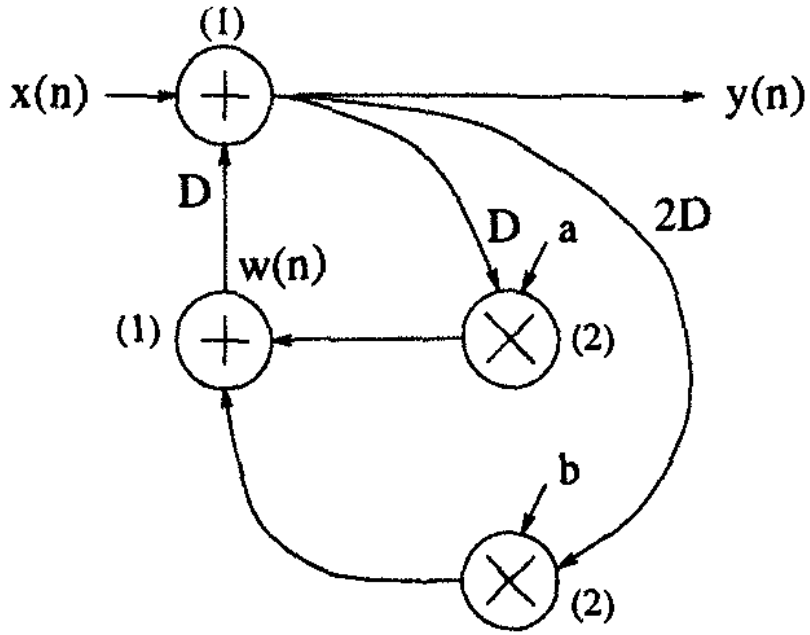
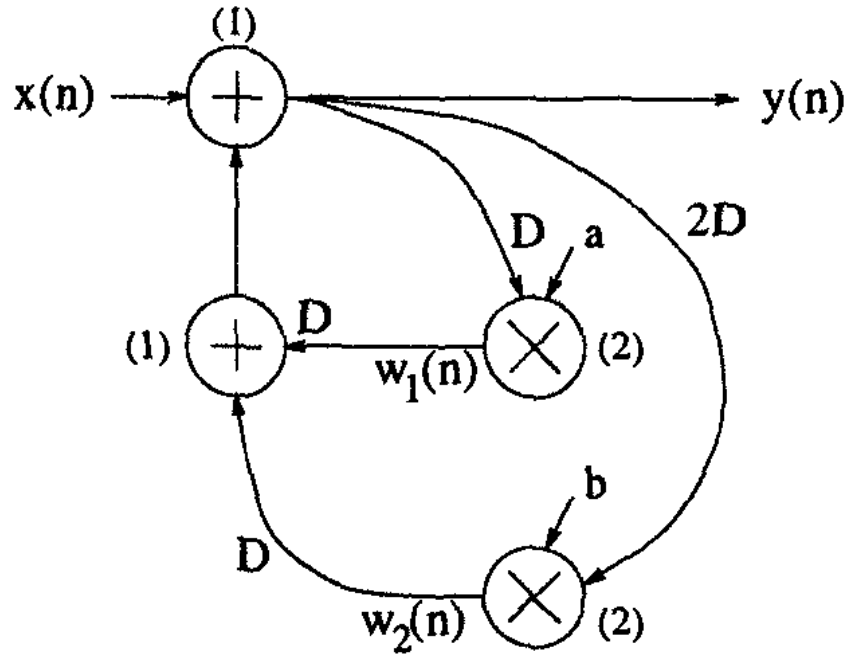
Retiming

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



(a)

$$
\begin{aligned}
w(n) &= ay(n-1) + by(n-2) \\
y(n) &= w(n-1) + x(n) \\
&= ay(n-2) + by(n-3) + x(n).
\end{aligned}
$$



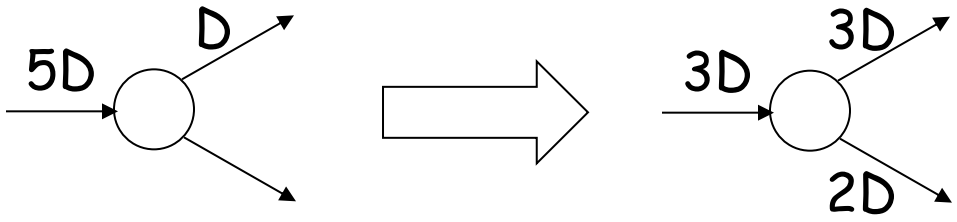
(b)

$$
\begin{aligned}
w_1(n) &= ay(n-1) \\
w_2(n) &= by(n-2) \\
y(n) &= w_1(n-1) + w_2(n-1) + x(n) \\
&= ay(n-2) + by(n-3) + x(n).
\end{aligned}
$$

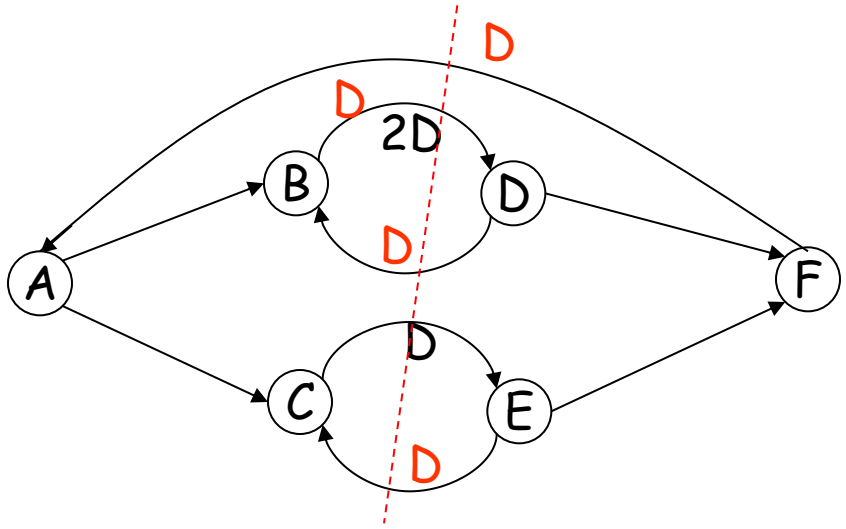
Retiming

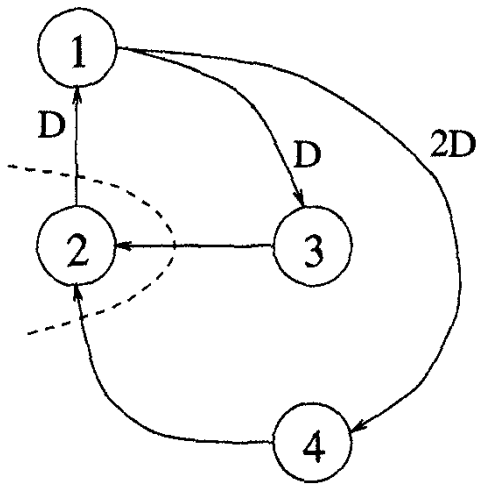
Moving around existing delays

- Does not alter the latency of the system
- Reduces the critical path of the system
- Node Retiming

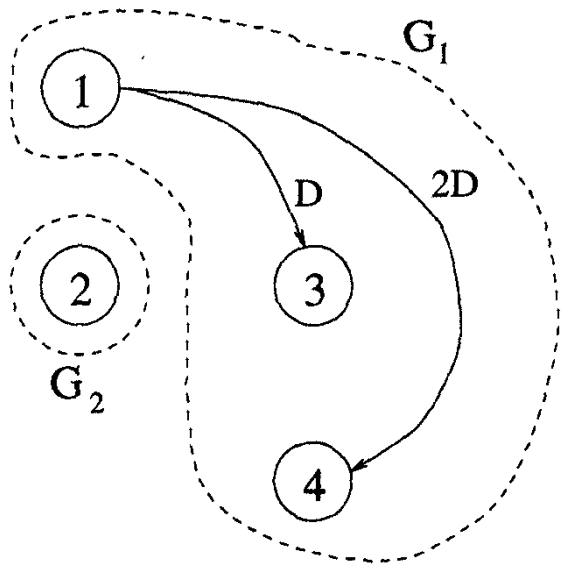


- Cutset Retiming

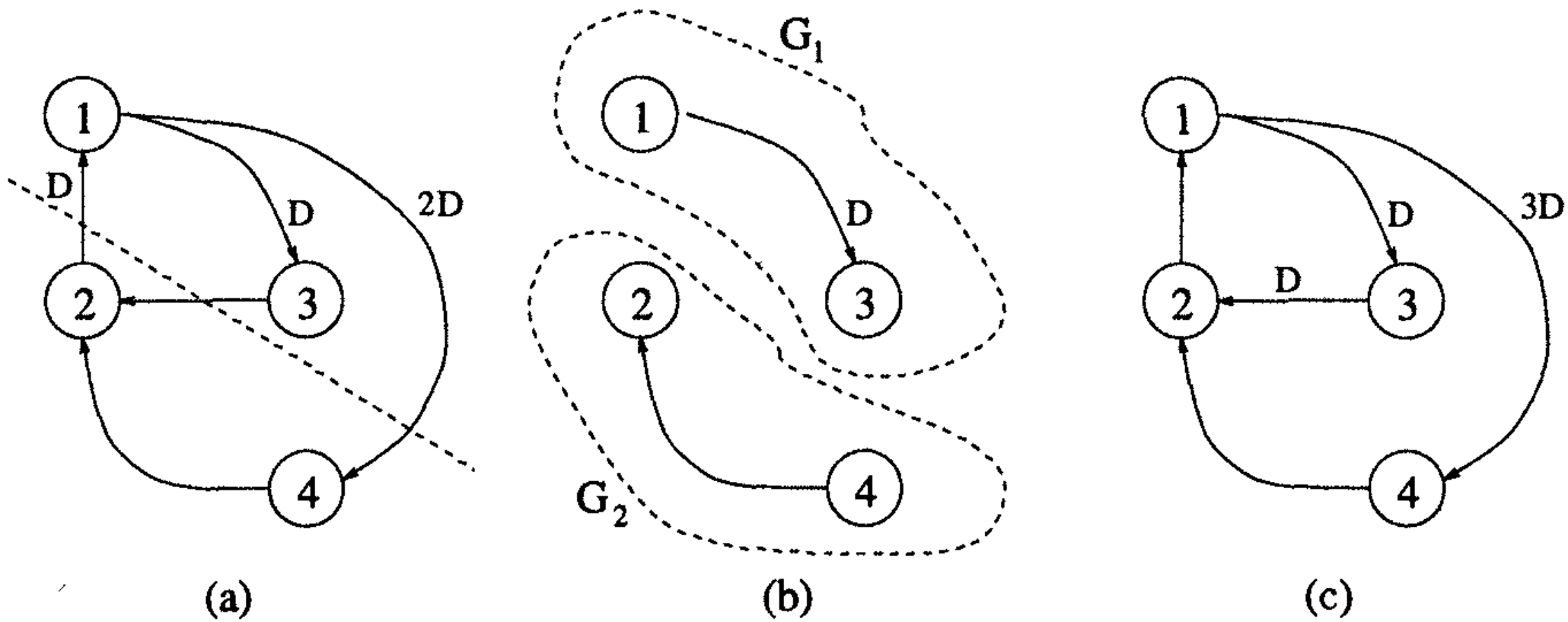




(a)



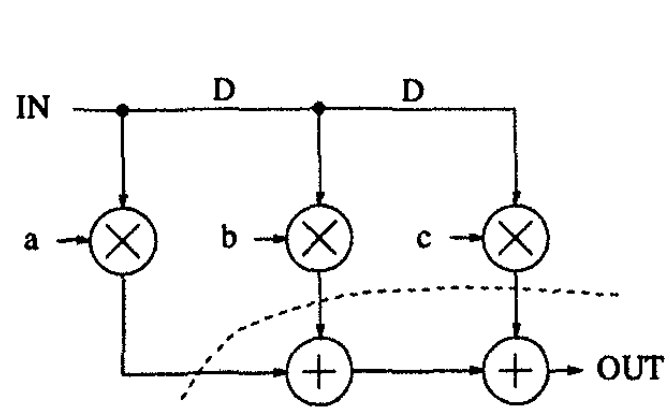
(b)


1
2
3
4
D
D
2D
(a)
G_1
G_2
1
2
3
4
D
(b)
1
2
3
4
D
D
3D
(c)

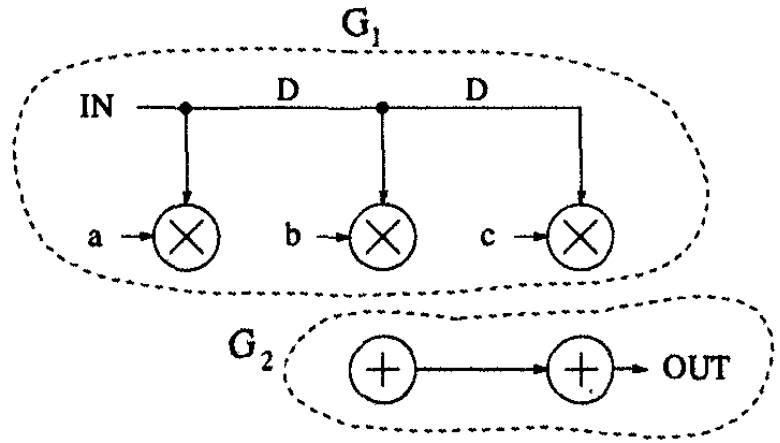
Retiming and pipelining

Generalization of pipelining

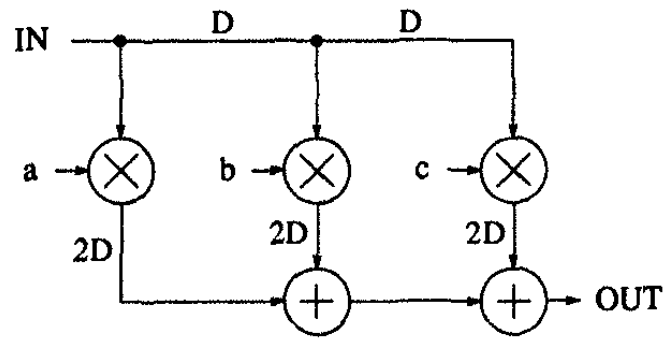
Pipelining is equivalent to introducing many delays at the input followed by retiming



(a)

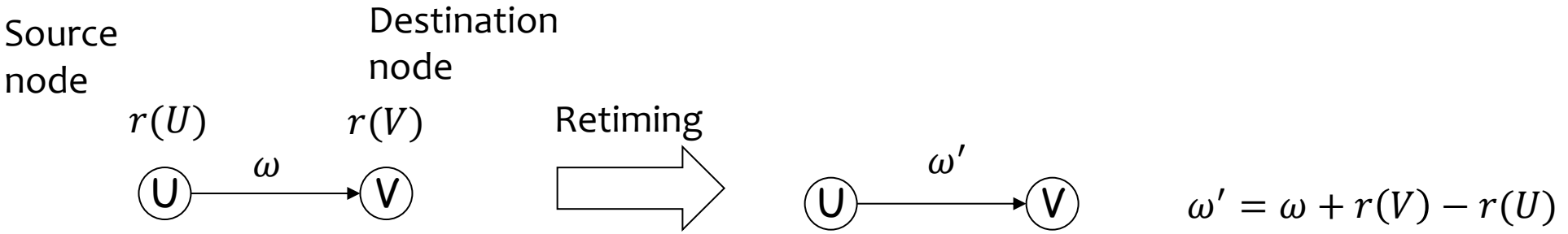


(b)



(c)

Retiming Formulation

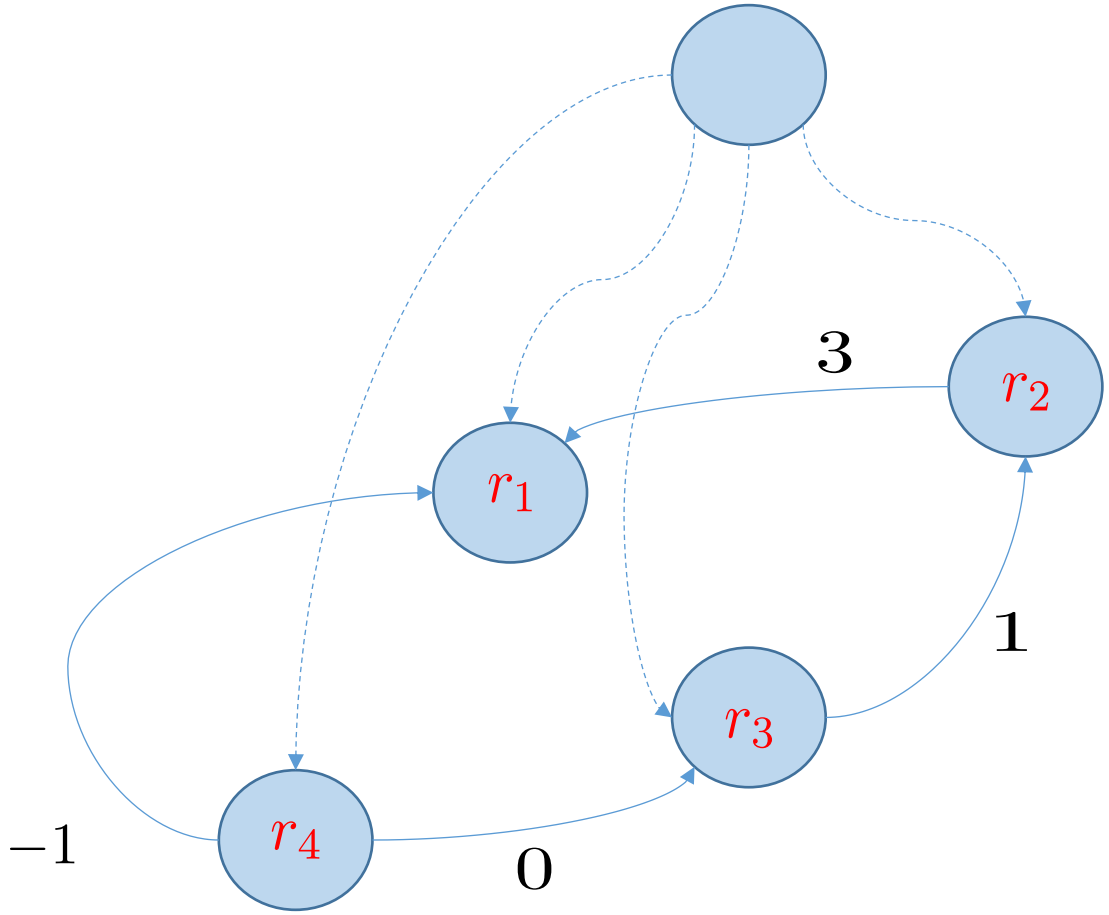


- Properties of retiming
  - The weight of the retimed path $p = V_0 \rightarrow V_1 \rightarrow \cdots \rightarrow V_k$ is given by $\omega_r(p) = \omega(p) + r(V_k) - r(V_0)$
  - Retiming does not change the number of delays in a cycle.
  - Retiming does not alter the iteration bound in a DFG
  - Adding the constant value j to the retiming value of each node does not alter the number of delays in the edges of the retimed graph.
- Retiming is done to meet the following
  - Clock period minimization
  - Register minimization
  - Reduce power consumption

Solving a system of inequalities

- Given M inequalities in N variables, where each inequality is of the form $r_i - r_j \leq k$, for integer values of k: **Use a shortest path algorithm:**
  - Draw a constraint graph
    - Draw a node i for each of the N variables r_i, $i = 1,2, \dots, N$.
    - Draw a node $N + 1$.
    - For each inequality $r_i - r_j \leq k$, draw the edge $j \rightarrow i$ of length k.
    - For each node $i, i = 1,2, \dots, N$, draw the edge $N + 1 \rightarrow i$ from the node $N + 1$ to node i with length 0.
  - Solve using a shortest path algorithm.
    - The system of inequalities ***has a solution***, iff the constraint graph contains ***no negative cycles.***
    - If a solution exists, one solution is where r_i is the minimum-length path from the node $N + 1$ to node i.

$$r_1 - r_2 \leq 3$$
$$r_3 - r_4 \leq 0$$
$$r_2 - r_3 \leq 1$$
$$r_1 - r_4 \leq -1$$



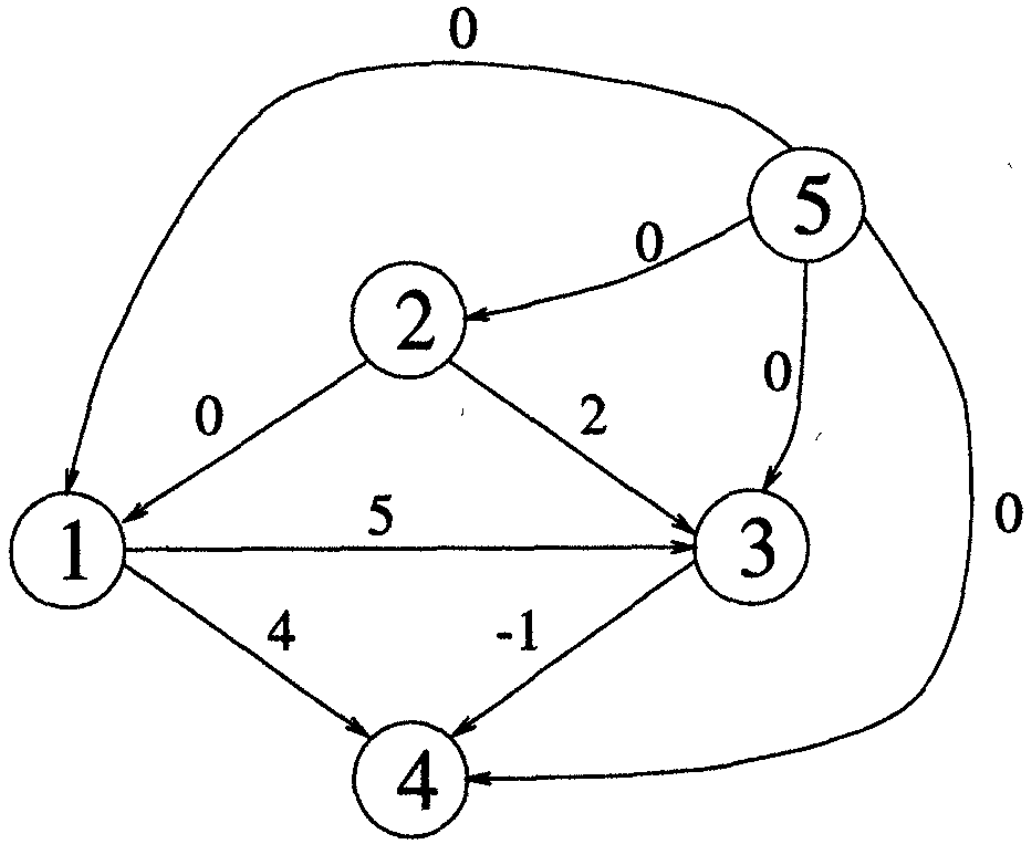
- Solution exists when nonnegative cycles
- Solve shortest path problem from root to every node

Solve systems of inequalities

Παράδειγμα

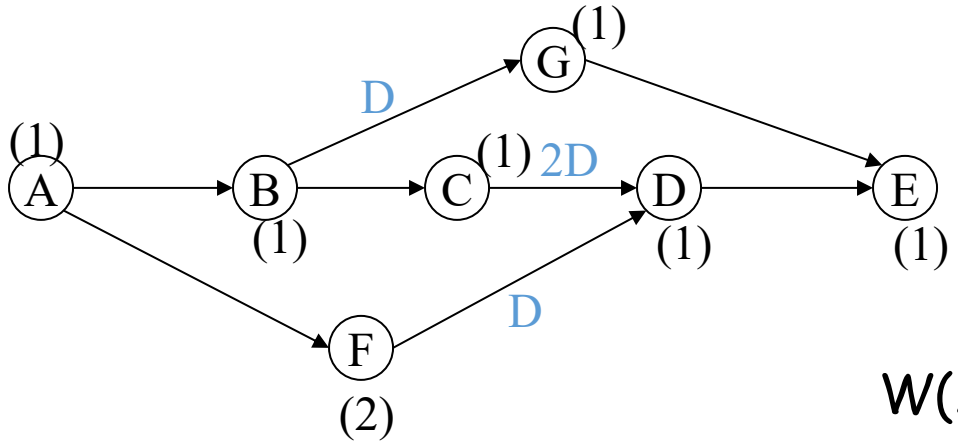
$$
\begin{aligned}
r_1 - r_2 &\leq 0 \\
r_3 - r_1 &\leq 5 \\
r_4 - r_1 &\leq 4 \\
r_4 - r_3 &\leq -1 \\
r_3 - r_2 &\leq 2.
\end{aligned}
$$

$$
\mathbf{R}^{(6)} = \begin{bmatrix}
\infty & \infty & 5 & 4 & \infty \\
0 & \infty & 2 & 1 & \infty \\
\infty & \infty & \infty & -1 & \infty \\
\infty & \infty & \infty & \infty & \infty \\
0 & 0 & 0 & -1 & \infty
\end{bmatrix}
$$



Retiming for Clock Period Minimization

- ***Feasibility constraint***: forces the number of delays on each edge of the retimed graph to be non negative
  - $\omega'(U,V) \geq 0$ causality of the system
  - $\omega'(U,V) \geq r(U) - r(V)$ (one inequality per edge)
- ***Critical Path constraint:*** enforces that the minimum feasible clock period Φ(G) is less than the target clock period c
  - Define two quantities $W(U,V)$ and $D(U,V)$ as:
    - $W(U,V) = \min\{w(p): U \to V\}$
    - $D(U,V) = \max\{t(p): U \to V \text{ and } w(p) = W(U,V)\}$
- Then it should hold that $r(U) - r(V) \leq W(U,V) - 1$, for all vertices U and V in the graph, for which $D(U,V) > c$.

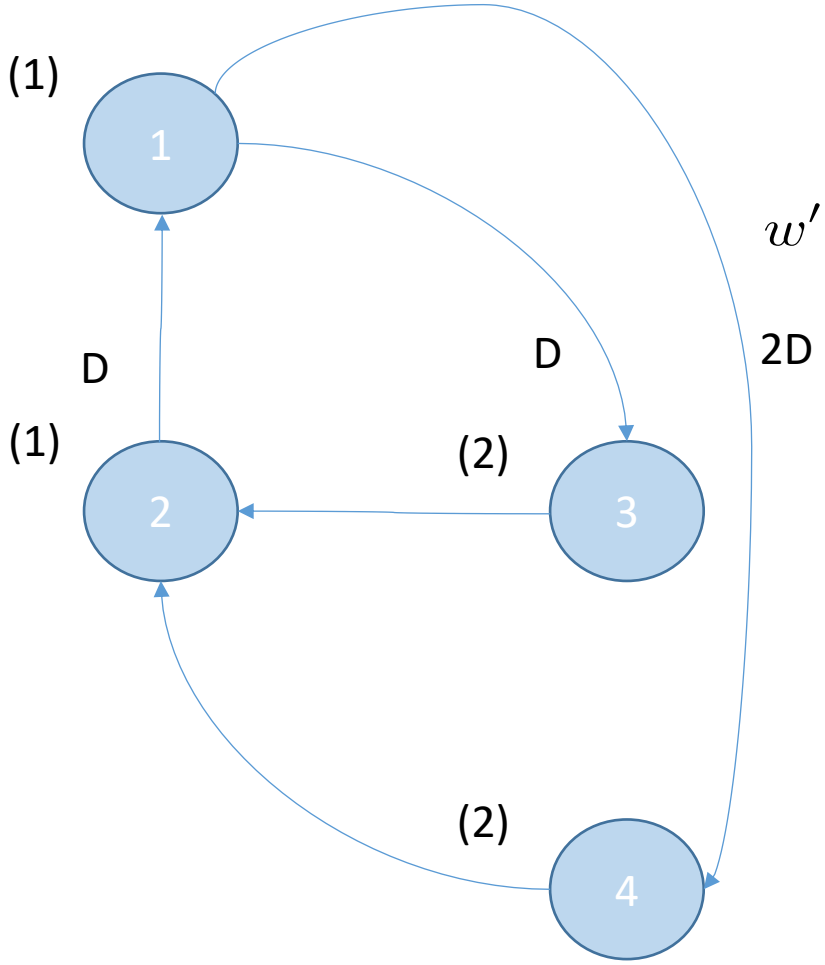


Namely: For those paths for which the computation time exceeds the target clock period c,
we demand that they are not critical paths (their retimed version contains at least 1 delay)

W(A,E) = 1 & D(A,E) = 5

Algorithm to compute $W(U,V)$ and $D(U,V)$

- Let $M = t_{\max} n$, where $t_{\max}$ is the maximum computation time of the nodes in G and n is the # of nodes in G.
- Form a new graph G' which is the same as G except the edge weights are replaced by $w'(e) = Mw(e) - t(U)$, for all edges $e: U \to V$.
- Solve for all-pairs shortest path problem on G' by using Floyd-Warshall algorithm. Let S'_{UV} be the shortest path from $U \to V$.
- If $U \neq V$, then
  - $W(U,V) = \left\lceil \frac{S'_{UV}}{M} \right\rceil$ and
  - $D(U,V) = MW(U,V) - S'_{UV} + t(V)$.
- If $U = V$, then $W(U,V) = 0$ and $D(U,V) = t(U)$.
- Using $W(U,V)$ and $D(U,V)$, the **feasibility** and **critical path** constraints are formulated to give certain inequalities.
- The inequalities are solved using constraint graphs, and, ***if a feasible solution is obtained***, then the circuit can be clocked with a period c.



$$M = nt_{\max}$$

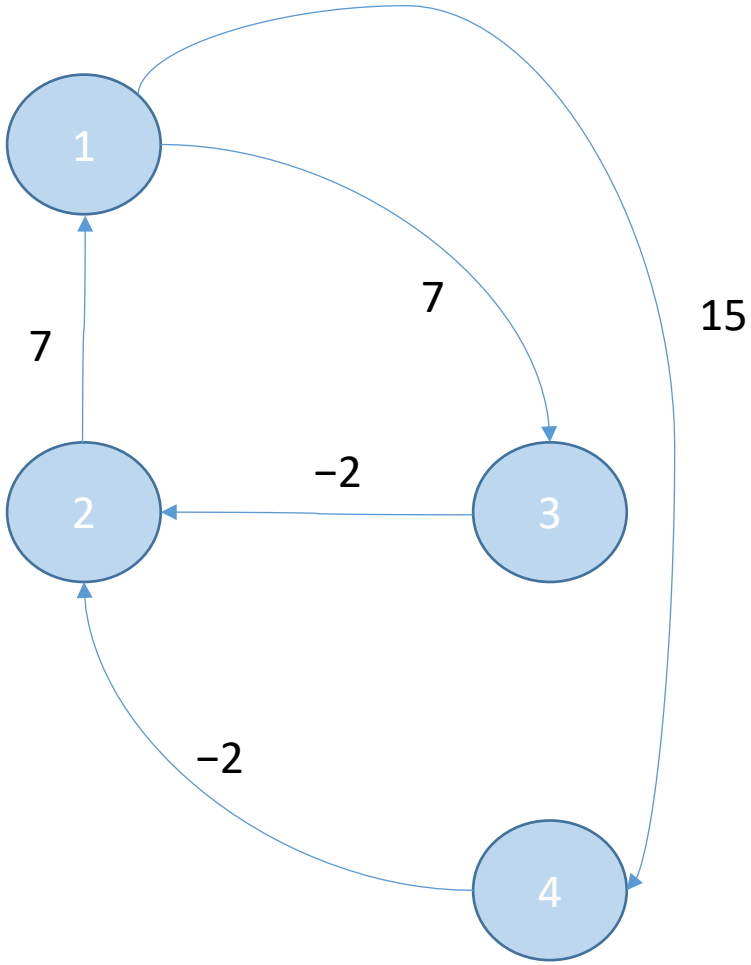
$$e : U \to V$$

$$w'(e) = Mw(e) - t(U)$$

$$n = 4$$

$$t_{\max} = 2$$

$$M = 8$$



Cutset retiming

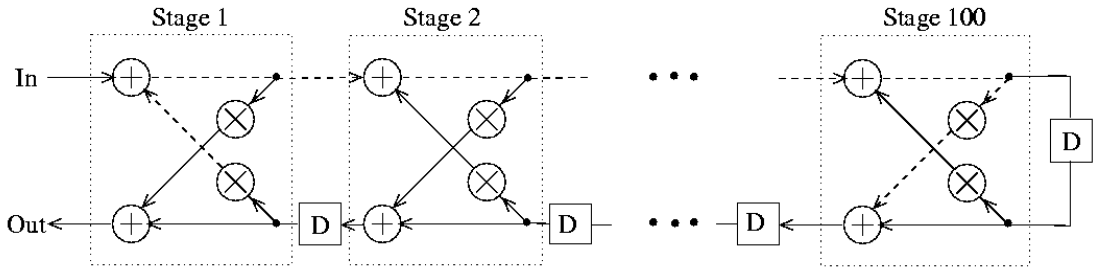
- Special case of retiming: A graphical method that simplifies complex retiming
- Only affects the weights of the cutset edges
- Add k delays to graph G_1 – remove k delays from other graph G_2
- Each node in G_1 has a retiming value j and each node in G_2 has $j + k$
- Pipelining is a special case of cutset retiming (and of retiming):
  - No edges in cutset that go from G_2 to G_1 (feed forward cutsets)
- **Cutset retiming** is often used in combination with slow down:
  - each delay is replaced with N delays to create an N-slow version of the DFG.
    Note that $N - 1$ null operations must be interleaved after each useful signal sample to preserve the functionality.

A 100-stage **Lattice Filter** (Fig. 4.7)

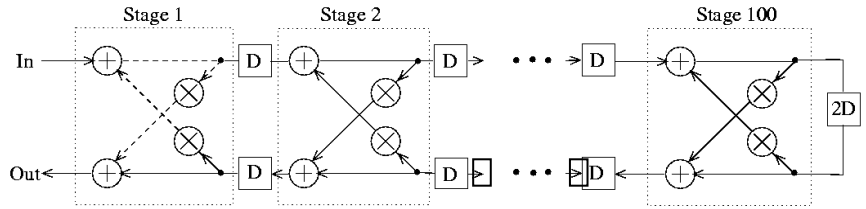
Critical path: 2 mults and 101 adds.

Assume T_m = 2 ut and T_a = 1 ut

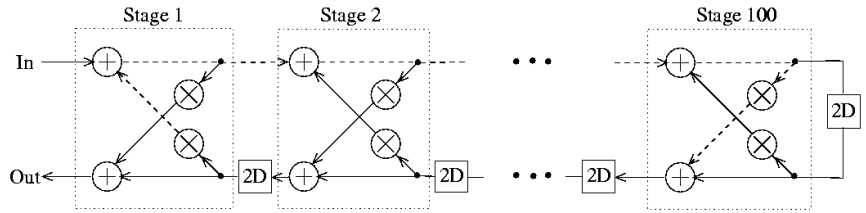
➔ $T_{sample} = T_{clk} = 105$ ut



A 2-slow version of the circuit (T_{clk} = 105 u.t.) (Input new samples every alternate cycle)

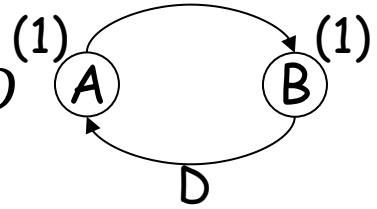


- **A (cutset) retimed version** of the 2-slow circuit (T_{clk} = 6 ut)
- **Critical path**: 2 multiplications and 2 additions
  ➔ T_{sample} = 2X6 = 12 ut

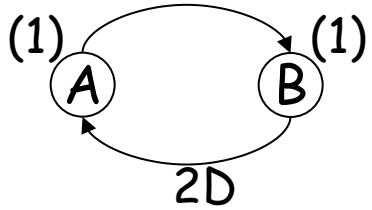


k-slow transformation

- Replace each D by kD



- After 2-slow transformation



Clock	
0	A0→B0
1	A1→B1
2	A2→B2

$T_{\text{iter}} = 2$ ut

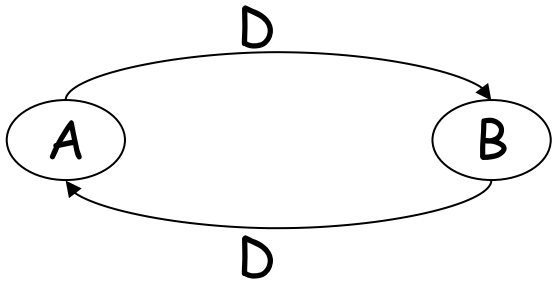
Clock	
0	A0→B0
1	
2	A1→B1
3	
4	A2→B2

$T_{\text{clk}} = 2$ ut

$T_{\text{iter}} = 2 \times 2 = 4$ ut

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

Retiming a 2-slow graph

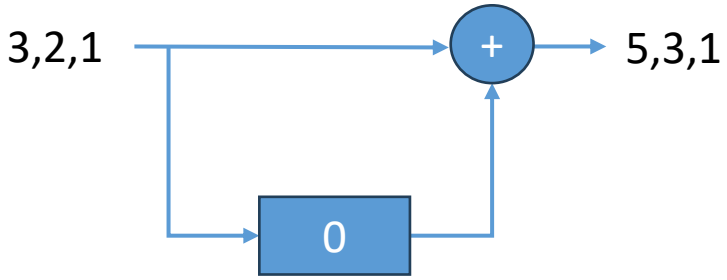


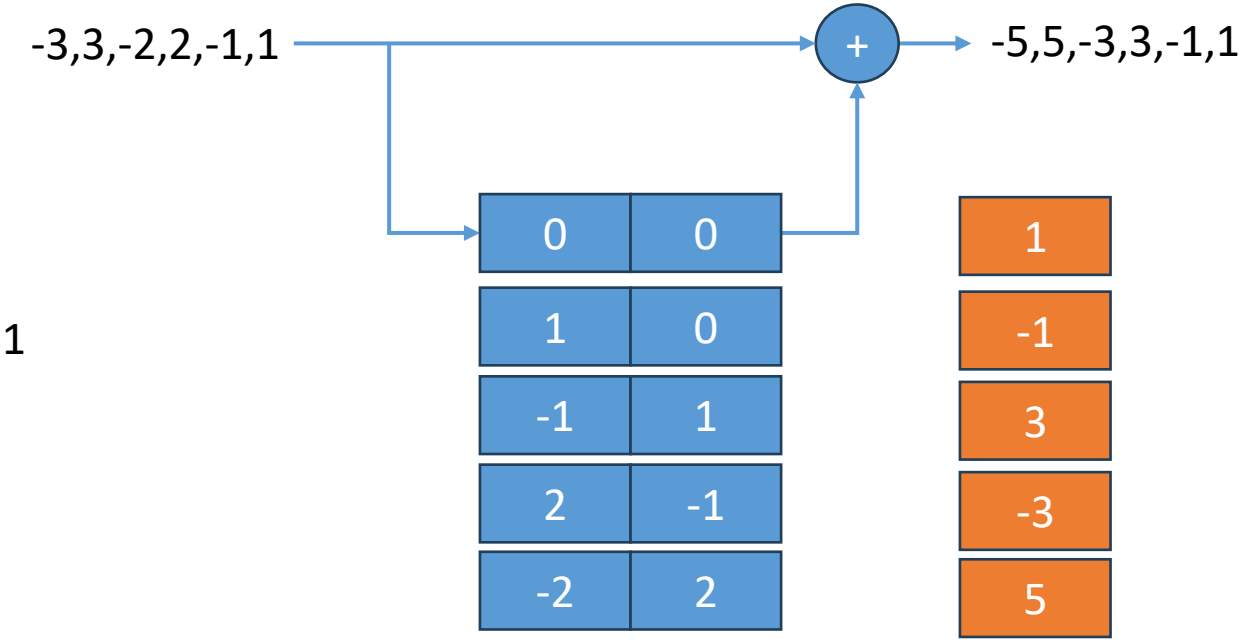
- $T_{\text{clk}} = 1$ ut
- $T_{\text{iter}} = 2 \times 1 = 2$ ut

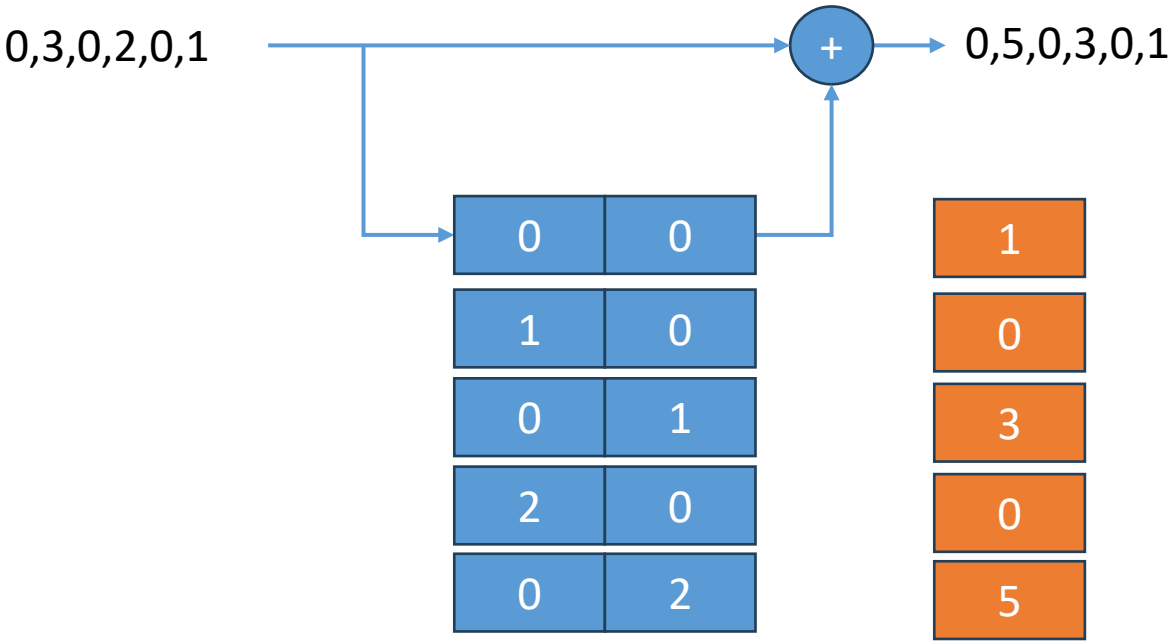
- Hardware Utilization = 50 %

- Hardware can be fully utilized if two independent operations are available.

Μετασχηματισμός N-slow







Other Applications of Retiming

- Retiming for Register Minimization (Section 4.4.3)
- Retiming for Folding (Chapter 6)
- Retiming for Power Reduction (Chap. 17)
- Retiming for Logic Synthesis (Beyond Scope of This Class)