Chapter 3
Pipelining and parallel Processing

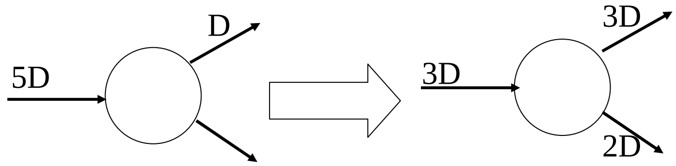
CSE4210 Winter 2012

Mokhtar Aboelaze

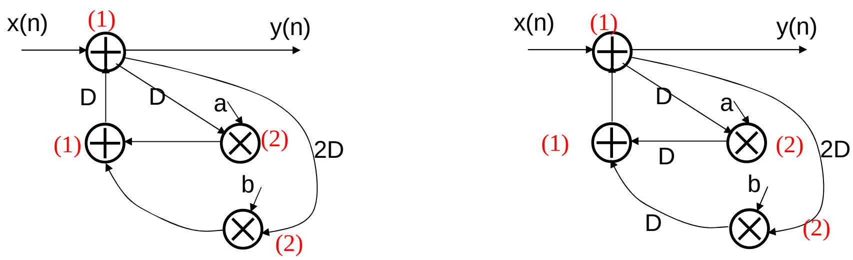
Introduction

- Retiming is a transformation technique that is used to change the locations of delay elements in a circuit without changing its functionality.
- Can be used to reduce the number of registers, or the clock cycle
- Could be considered as a generalization of the pipelining technique studies earlier

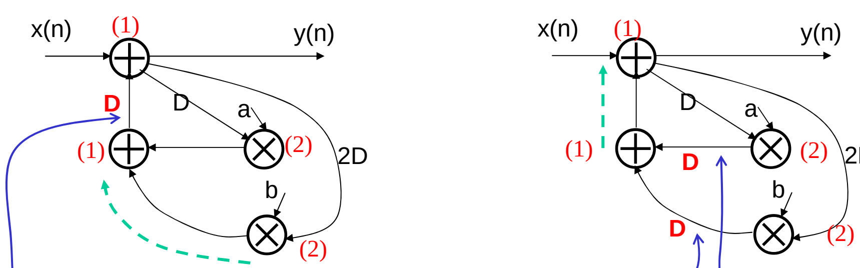
Introduction



y(n)=ay(n-2) + by(n-3) +x (n)



Introduction



y(n)=ay(n-2) + by(n-3) +x (n)

Critical path = 2

w(n)=ay(n-1)+by(n-2)

y(n)=w(n-1) +x (n)

Critical path = 3

w1(n)=ay(n-1)

w2(n)=by(n-2)

y(n)=w(n-1) +x (n)

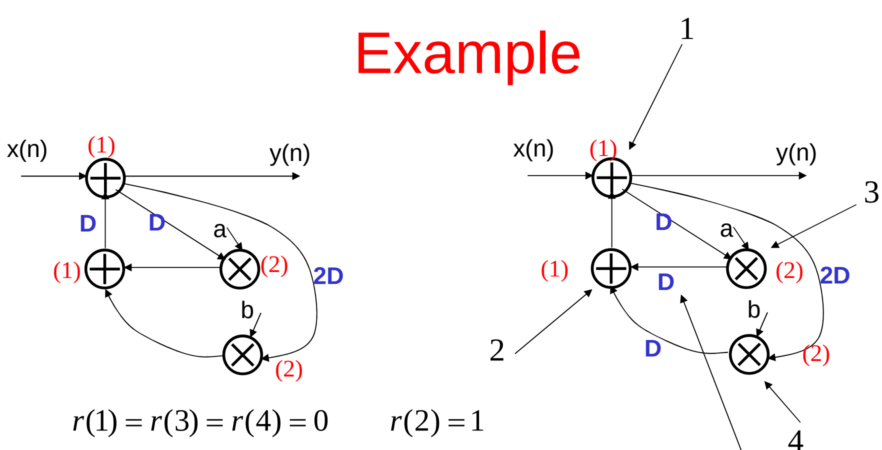
Definitions

- Mapping G to G_r
- U and V are nodes, e is an edge
- r(v) is a retiming value
- w(e) is the weight of edge e in graph G
- $w_r(e)$ is the weight of edge e in graph G_r

$$w_r(e)=w(e)+r(V)-r(U)$$

- A solution is feasible if all $w_r \geq 0$

Example



$$r(1)=r(3)=r(4)=0 \qquad r(2)=1$$

$$w_r(3 \xrightarrow{e} 2) = w(3 \xrightarrow{e} 2) + r(2) - r(3) = 0+1-0=1$$

$$w_r(4 \xrightarrow{e} 2) = w(4 \xrightarrow{e} 2) + r(2) - r(4) = 0+1-0=1$$

Definitions and Properties

- The weight of a retimed Path

$$p = V_1 \rightarrow V_2 \rightarrow V_3 \cdots V_k$$
$$p = w_r(V_1 \rightarrow V_2) + w_r(V_2 \rightarrow V_3) + \cdots w_r(V_{k-1} \rightarrow V_k)$$
$$p = w(V_1 \rightarrow V_2) + r(2) - r(1) + w(V_2 \rightarrow V_3) + r(3) - r(2) + \cdots w(V_{k-1} \rightarrow V_k) + r(k) - r(k-1)$$
$$p = w(V_1 \rightarrow V_2) + w(V_2 \rightarrow V_3) + \cdots w(V_{k-1} \rightarrow V_k) + r(k) - r(0)$$
$$p = w(p) + r(k) - r(0)$$

Definitions and Properties

- Retiming does not change the number of delays in a cycle
  - Put k=0 in the previous equation
- Retiming does not alter the iteration bound in a DFG
- Adding a constant *j* to the retiming value of each node does not change the mapping from *G* to G_r

Solving Systems of Inequalities

- From the constraints, we end up with a system of equations in the form

$$r_i - r_j \le k$$

- Any solution that satisfy the constraints is a viable solution.
- A procedure to solve the system

Solving Systems of Inequalities

- Draw a constraints graph
  - Draw the node i for each of the N variables 1 2 .. N
  - Draw the node N+1
  - For each inequality $r_i - r_j \le k$, draw an edge from node j $\rightarrow$ i with weight k
  - For each node i=1,2,..N draw an edge N+1 $\rightarrow$ i with weight 0
- Solve
  - The system has a solution iff the constraints graph has no negative cycle. Bellman Ford Algorithm
  - One solution is the min. length from node N+1 to i

Example

1

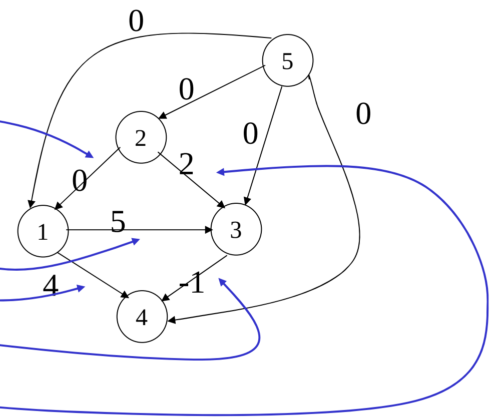
$r_1 - r_2 \le 0$

$r_3 - r_1 \le 5$

$r_4 - r_1 \le 4$

$r_4 - r_3 \le -1$

$r_3 - r_2 \le 2$



YORK UNIVERSITY CSE4210

Example

1

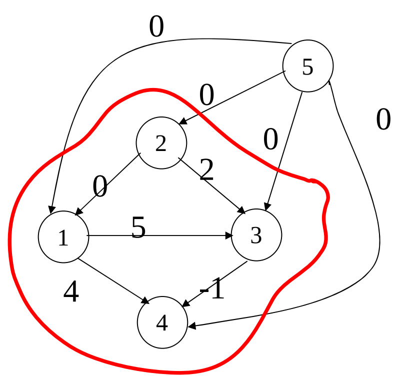
$r_1 - r_2 \le 0$

$r_3 - r_1 \le 5$

$r_4 - r_1 \le 4$

$r_4 - r_3 \le -1$

$r_3 - r_2 \le 2$



r[4] = -1 (5→3→4)

r[?] = 0

Cutset Retiming

- Cutset: A set of edges if removed, the graph G is partitioned into 2 graphs G_1,G_2 .
- Cutset retiming is done by adding k delays to all the edges in the cutset from G_1 to G_2, and removing k delays from the edges from G_2 to G_1

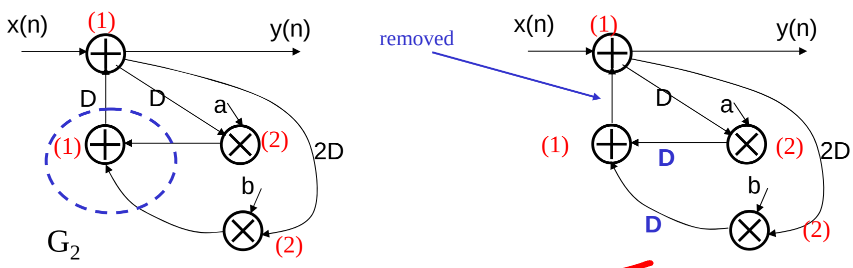
$$-\min_{G_1 \xrightarrow{e} G_2} \{w(e)\} \le k \le \min_{G_2 \xrightarrow{e} G_1} \{w(e)\}$$

Cutset Retiming

- Pipelining is a special case where there are no nodes from G_2 to G_1 (no loops).
- Cutset is combined with *slow-down*, where first an N-slow-down version of the graph is created by changing every *D* to *ND*, then retiming is used.
- With the N-slow-down version, the input is slowed down to (N-1 null operation or 0 samples must be interleaved with input data

Example

y(n)=ay(n-2) + by(n-3) +x (n)

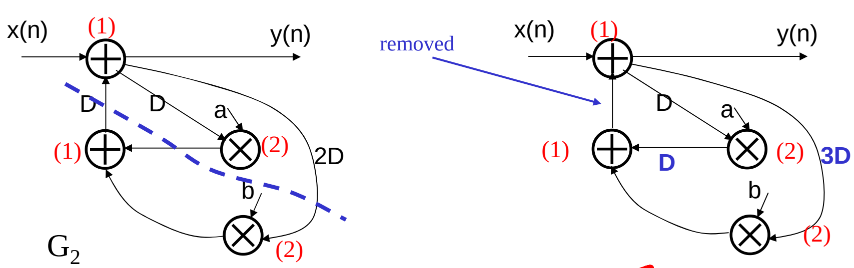


$T_M + T_A$ Clock Cycle ?

$2T_A$

Example

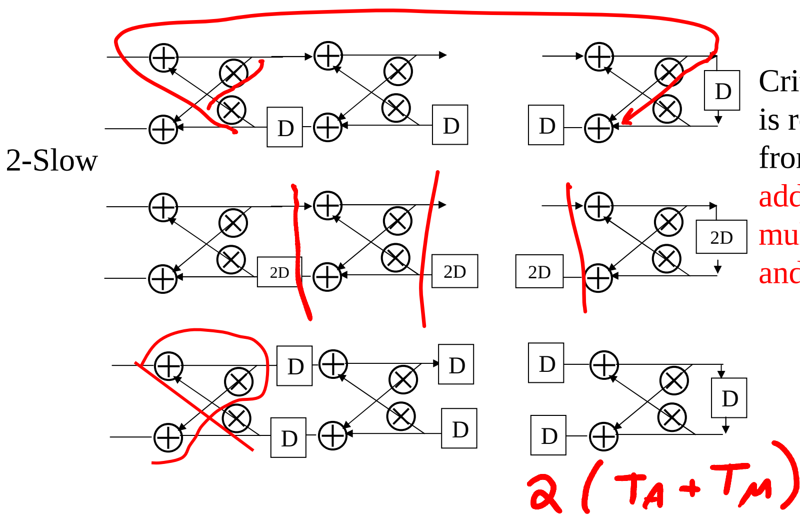
y(n)=ay(n-2) + by(n-3) +x (n)



$T_M + T_A$ Clock Cycle ?

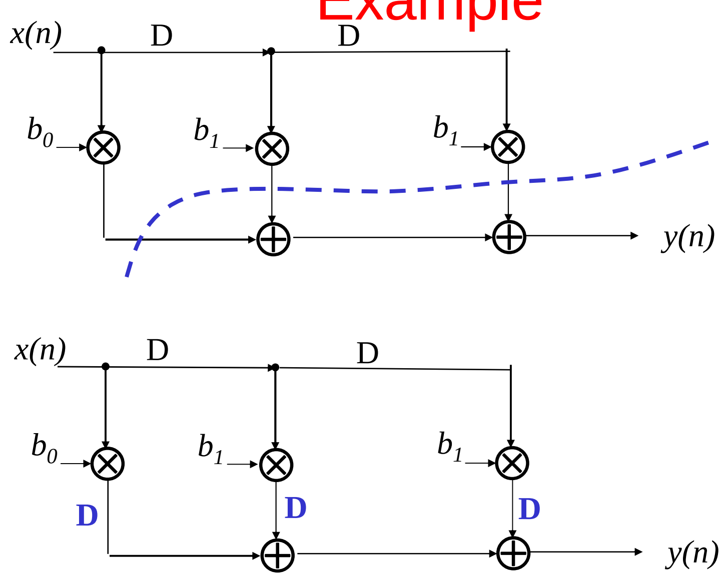
$2T_A$

100 stage lattice filter



Critical path is reduced from 101 addition + 2 mul to 2 add and 2 mul.

Example



Retiming for Clock Period Minimization

- The min. clock time is the computation time of the critical path.
- Critical path is the path with the longest computation time and **no delay**.
- Retiming could be used to minimize clock period.

Minimize Clock Period

- Min. feasible clock period of a graph G is $\Phi(G)=\max\{t(p): w(p)=0\}$
- W(U,V) is the minimum number of registers on any path from U $\rightarrow$ V
- D(U,V) is the max. computation time among all paths from U $\rightarrow$ V with weight W(U,V)

Minimize Clock Period

1. Let $M=t_{max}n$, where t_{max} is the max. computation time of any node in G, n =number of nodes in G
2. Form a new graph G' which is the same as G except the edge weights are replaced by $w'(e)=Mw(e)-t(U)\ (e=U \rightarrow V)$
3. Solve for all-pairs shortest path on G' (S_{UV})
4. If $U \neq V$, then $W(U,V)=\lceil S_{UV}/M \rceil$ and $D(U,V)=M \times W(U,V)-S_{UV}+t(V)$
5. IF $U=V$, $W(U,V)=0$, $D(U,V)=t(U)$

Minimize Clock Period

- Then, we use $W(U,V),D(U,V)$ to find if there a retiming solution such that $\Phi(G) \leq c$
- Construct the following set of constraints
- **Feasibility constraints**

  $r(U)-r(V) \leq w(e)$ for every edge in G
- **Critical path constraint**

  $r(U)-r(V) \leq W(U,V)-1$ for all nodes U,V in G such that $D(U,V)>c$ (cycle time).
- Solve to get r(.) (retiming values)

$$w_r(p)=w(p)+r(b)-r(a)$$

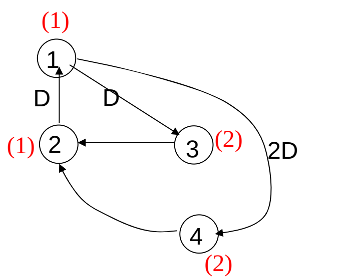
Example

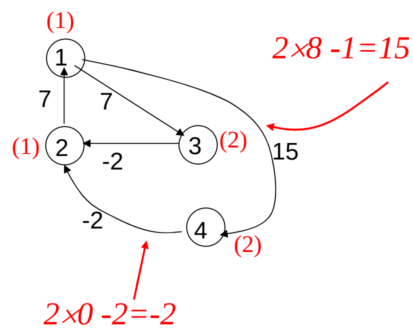
Step 1

M=2*4=8

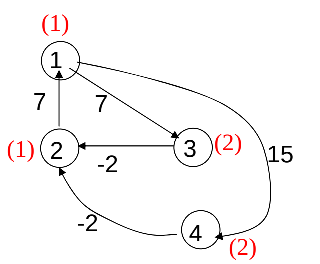
e = U → V

$w'(e) = M \times w(e) - t(U)$





y(n)=ay(n-2) + by(n-3) +x (n)



$U \neq V$, then $W(U,V) = \lceil SUV/M \rceil$

W(U,V)	1	2	3	4
1	0	1	1	2
2	1	0	2	3
3	1	0	0	3
4	1	0	2	0

S_{UV}	1	2	3	4
1	12	5	7	15
2	7	12	14	22
3	5	-2	12	20
4	5	-2	12	20

$D(U,V) = M \times W(U,V) - S_{UV} + t(V)$

$U = V \quad T(U)$

D(U,V)	1	2	3	4
1	1	4	3	3
2	2	1	4	4
3	4	3	2	6
4	4	3	6	2

Example, cont.

Feasibility constraints

C=3

r(1)-r(3) ≤ 1
r(1)-r(4) ≤ 2
r(2)-r(1) ≤ 1
r(3)-r(2) ≤ 0
r(4)-r(2) ≤ 0

All r(.) = 0, means the graph already has a cycle of 3

r(U)-r(V) ≤ W(U,V)-1* for all nodes *U,V* in *G* such that *D(U,V)>3

Critical path constraints

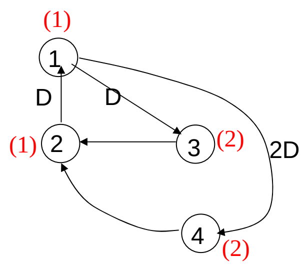
r(1)-r(2) ≤ 0
r(2)-r(3) ≤ 1
r(2)-r(4) ≤ 2
r(3)-r(1) ≤ 0
r(3)-r(4) ≤ 2
r(4)-r(1) ≤ 0
r(4)-r(3) ≤ 1

D(U,V)	1	2	3	4
1	1	4	3	3
2	2	1	4	4
3	4	3	2	6
4	4	3	6	2

W(U,V)	1	2	3	4
1	0	1	1	2
2	1	0	2	3
3	1	0	0	3
4	1	0	2	0

Example

- Solving the above inequalities, results in a solution
- r(1)=r(2)=r(3)=r(4)=0
- The graph already has a critical path =3



Redoing for c=2

r(U)-r(V) ≤W(U,V)-1* for all nodes *U,V* in *G* such that *D(U,V)>2

Critical path constraints

r(1)-r(2) ≤ 0
r(2)-r(3) ≤ 1
r(2)-r(4) ≤ 2
r(3)-r(1) ≤ 0
r(3)-r(4) ≤ 2
r(4)-r(1) ≤ 0
r(4)-r(3) ≤ 1
r(1)-r(3) ≤ 0
r(1)-r(4) ≤ 1
r(3)-r(2) ≤ -1
r(4)-r(2) ≤ -1

D(U,V)	1	2	3	4
1	1	4	3	3
2	2	1	4	4
3	4	3	2	6
4	4	3	6	2

W(U,V)	1	2	3	4
1	0	1	1	2
2	1	0	2	3
3	1	0	0	3
4	1	0	2	0

Example

- Redo, for c=2
- r(2)=0, r(1)=r(3)=r(4)=-1