CSE4210 Architecture & Hardware for DSP

Chapter 7

Retiming

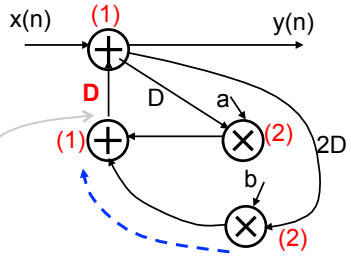
Instructor: Prof. Peter Lian
Department of Electrical Engineering & Computer Science
Lassonde School of Engineering
York University

Introduction

- Retiming is a transformation technique that is used to change the locations of delay elements in a circuit without changing its functionality.
- Could be considered as a generalization of the pipelining technique studies in Chapter 6
  - Pipelining is equivalent to introduce many delays at the input followed by retiming
- Applications
  - Reduce the number of registers in the circuit
  - Reduce the critical path of the system
  - Reduce the power consumption of the circuit
  - Logic synthesis

IIR Filter Example

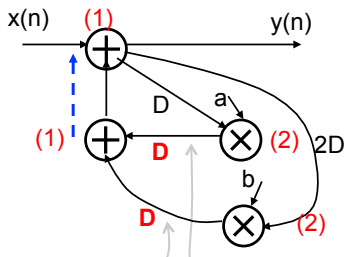
y(n)=ay(n-2) + by(n-3) +x (n)



Critical path = 3

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

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



Critical path = 2

w1(n)=ay(n-1)

w2(n)=by(n-2)

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

Retiming

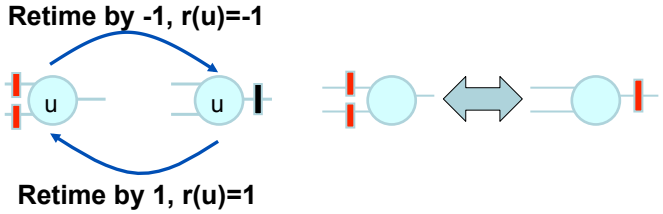
- Mapping G to G_r (retimed)
- U and V are nodes, e is an edge
- r(U) is a retiming value
- w(e) is the weight of edge e from node U → V in graph G
- $w_r(e)$ is the weight of edge e in from node U → V in retimed graph G_r

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

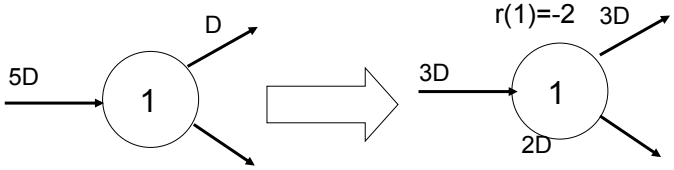
- A solution is feasible if all $w_r(e) \geq 0$

Retiming Value r(U)

- Movement of register from input to output or vice versa does not affect the functionality
- Retiming value of node U

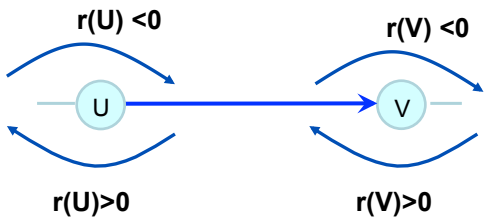


- Example

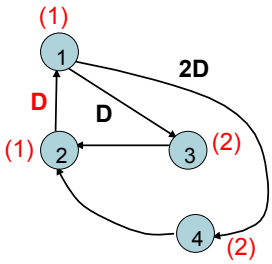


Feasible Solution

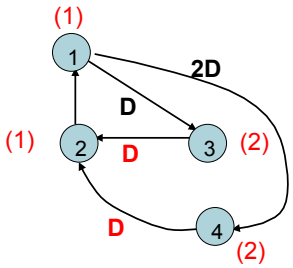
- A solution is feasible if all $w_r(e) \geq 0$
- $w_r(e)=w(e)+r(V)-r(U)$ for edge $e=(U,V)$
  - If $r(U)<0$, $r(V)>0$, $w_r(e)>0$
  - If $r(U)>0$, $r(V)<0$, $w_r(e)>0$ if $|r(U)+r(v)|<w(e)$
  - If $r(U)>0$, $r(V)>0$, $w_r(e)>0$ if $r(v)>r(U)$
  - If $r(U)<0$, $r(V)<0$, $w_r(e)>0$ if $|r(U)|>|r(V)|$



IIR Filter Example



Retimed



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

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

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

Retiming Properties

Property I

- The weight of a retimed Path is given by:

$$w_r(p) = w(p) + r(k) - r(0)$$

$$p = V_0 \xrightarrow{e_0} V_1 \xrightarrow{e_1} V_2 \cdots V_{k-1} \xrightarrow{e_{k-1}} V_k$$

$$w_r(p) = w_r(e_0) + w_r(e_1) + \cdots w_r(e_{k-1})$$

$$w_r(p) = w(e_0) + r(1) - r(0) + w(e_1) + r(2) - r(1) \cdots + w(e_{k-1}) + r(k) - r(k-1)$$

$$w_r(p) = w(e_0) + w(e_1) + \cdots w(e_{k-1}) + r(k) - r(0)$$

$$w_r(p) = w(p) + r(k) - r(0)$$

Properties II to IV

- Property II: Retiming does not change the number of delays in a cycle
  - This is a special case of Property I, where $V_k = V_0$
- Property III: Retiming does not alter the iteration bound in a DFG
- Property IV: Adding a constant *j* to the retiming value of each node does not change the mapping from *G* to G_r

Retiming Techniques

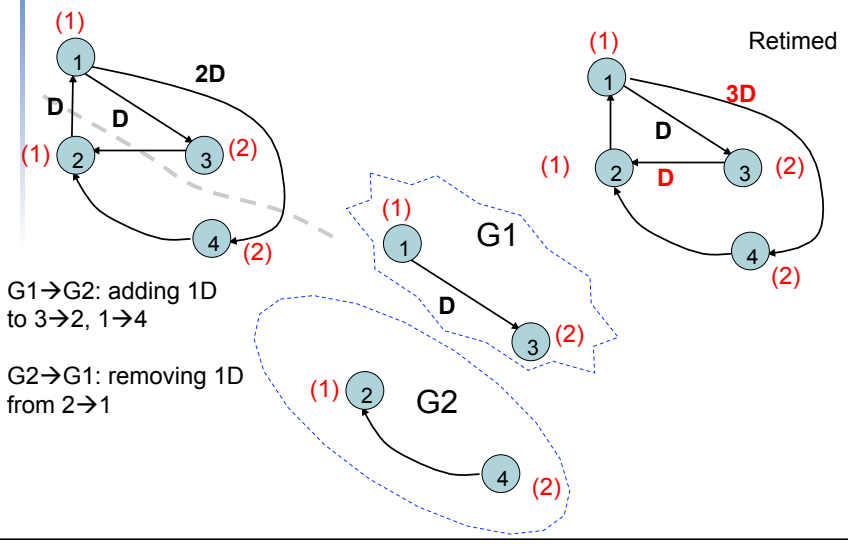
Cutset Retiming

- 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

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

Example 1

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

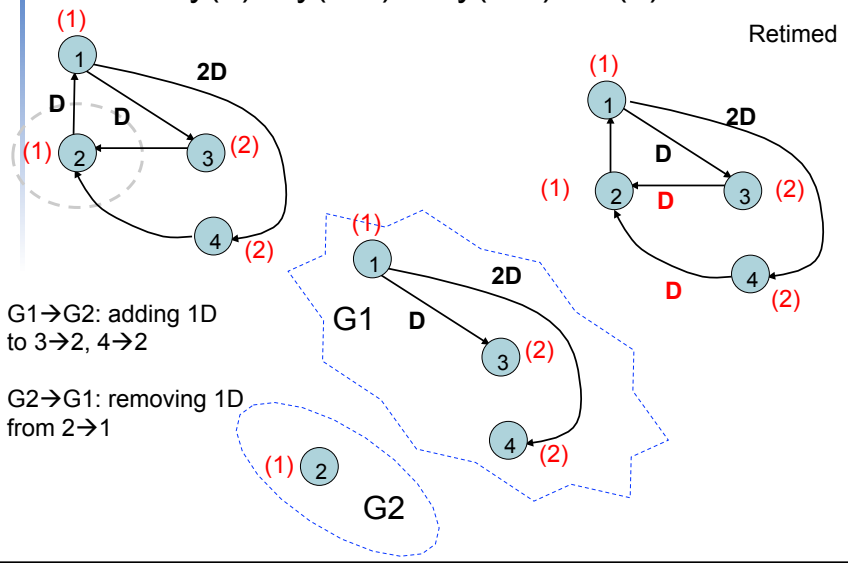


G1→G2: adding 1D to 3→2, 1→4

G2→G1: removing 1D from 2→1

Example 2

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

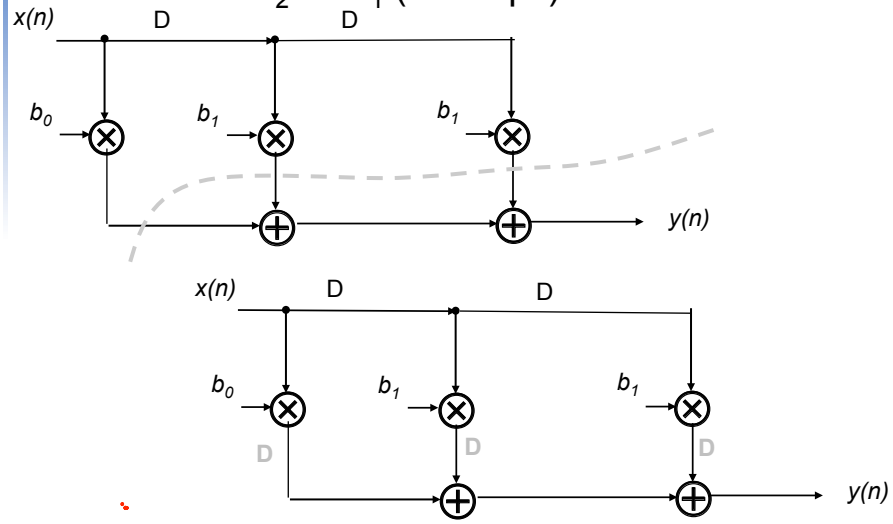


G1→G2: adding 1D to 3→2, 4→2

G2→G1: removing 1D from 2→1

Cutset Retiming: Pipelining

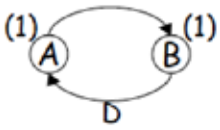
- Pipelining is a special case where there are no nodes from G_2 to G_1 (no loops).



K-Slow Transformation

- Replace each delay with N delays

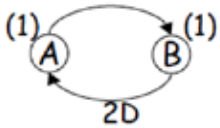
Example:



Clock	
0	A0→B0
1	A1→B1
2	A2→B2

T_{clk}=2ut
T_{iter}=2ut

After 2-slow transformation

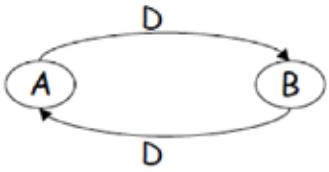


Clock	
0	A0→B0
1	
2	A2→B2
3	
4	A4→B4

T_{clk}=2ut
T_{iter}=2*2ut=4ut

K-Slow Transformation

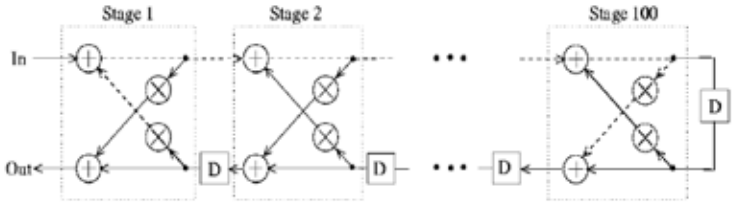
- K-slow transformation
  - Input new samples every alternative cycles
  - Null operations account for odd clock cycles.
  - Hardware utilized only 50% time
- Combining K-slow with retiming
  - Hardware utilization = 50%
  - Hardware can be fully utilized if two independent operations are available.



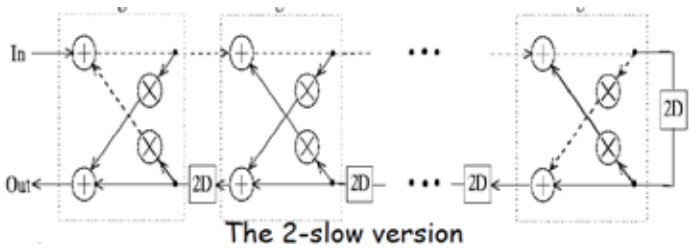
T_{clk}=1ut
T_{iter}=2*1ut=2ut

2-Slow Lattice Filter

- A 100-stage Lattice filter



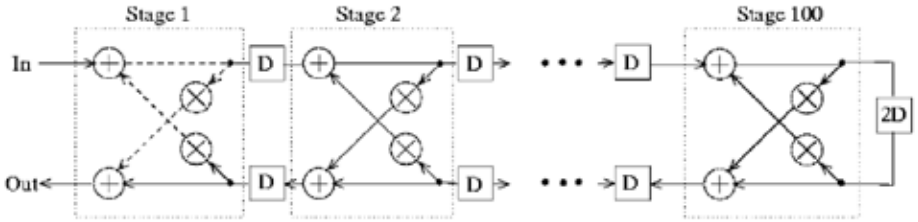
What is the critical path?



The 2-slow version

2-Slow Lattice Filter

- Retimed version



Critical path=2 multiplier + 2 adder

If T_{mul}=2 ut, T_{add}=1ut, then T_{clk}=6ut, T_{iter}=2*6=12ut

In original filter, T_{iter}=105

Solving System of Inequalities

Feasible Retiming Solution

- A solution is feasible if all $w_r(e) \geq 0$, i.e.

  $w_r(e)=w(e)+r(V)-r(U) \geq 0$

  → $r(U)-r(V) \leq w(e)$ for all edges

Example:

$$r_1-r_2 \leq 0$$

$$r_3-r_1 \leq 5$$

$$r_4-r_1 \leq 4$$

$$r_4-r_3 \leq -1$$

$$r_3-r_2 \leq 2$$

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 $r_i-r_j \leq k$, draw an edge from node j → i with weight k
  - For each node i=1,2,..N draw an edge N+1 → i with weight 0
- Solve
  - The system has a solution if the constraints graph has no negative cycle. Bellman Ford Algorithm
  - One solution is the minimum length from node N+1 to i

Activity 1

- Given the following inequalities, draw the constraint graph.

$$r_1 - r_2 \le 0$$

$$r_3 - r_1 \le 5$$

$$r_4 - r_1 \le 4$$

$$r_4 - r_3 \le -1$$

$$r_3 - r_2 \le 2$$

Retiming for Clock Period Minimization

Retiming for Clock Period Minimization

- The minimum 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

- Minimum 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 maximum computation time among all paths from U $\rightarrow$ V with weight W(U,V)

Steps in Minimize Clock Period

1. Let $M=t_{max}n$, where t_{max} is the maximum 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 $w'(e)=Mw(e)-t(U)$ ($e=U \rightarrow V$), where $t(U)$ is computation time of node U.
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 $D(U,V)=M \times W(U,V)-S_{UV}+t(V)$. $\lceil X \rceil$ denotes the smallest integer not less than X.
5. If $U=V$, $W(U,V)=0$, $D(U,V)=t(U)$

Steps in Minimize Clock Period

6. Use $W(U,V)$, $D(U,V)$ to find if there a retiming solution such that $\Phi(G) \leq$ c (cycle time).

   This is done by constructing 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 $D(U,V)>c$.

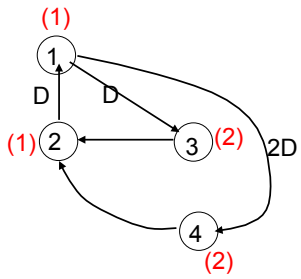
   - If there is a solution to the inequalities (constraints), then the solution is a feasible retiming solution that the circuit can be clocked with period c.

Example

Step 1: find *M*

n=4, t_{max}=2

M=2*4=8

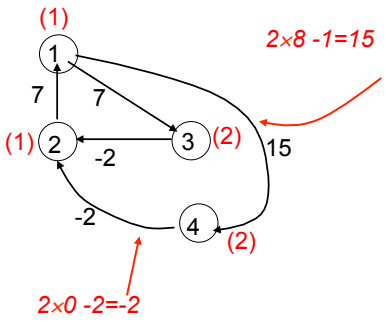


y(n)=ay(n-2) + by(n-3) +x (n)

Step 2: form new graph G'

e = U → V

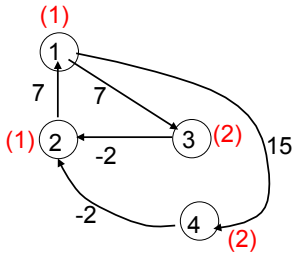
w'(e)=M × w(e)-t(U)



Example

Step 3: Solve all shortest path on G'

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



Example

Steps 4 and 5: Construct tables for W(U,V) and D(U,V)

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

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

$U=V$, $W(U,V)=0$

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

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

$U=V \quad T(U)$

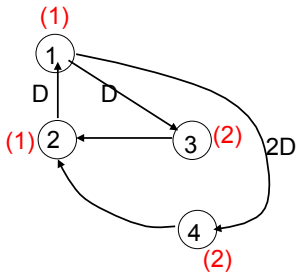
D(U,V)	1	2	3	4
1	1	4	3	3
2	2	1	4	4
3	4	3	2	6
4	4	3	6	2

Example

Step 6: Construct constraints, for c=3

- Feasibility constraints

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



Example

Critical path constraints

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

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

- Combine two sets of constraints, we have 12 inequalities.
  - Note that there is no overlap between these two sets of constraints

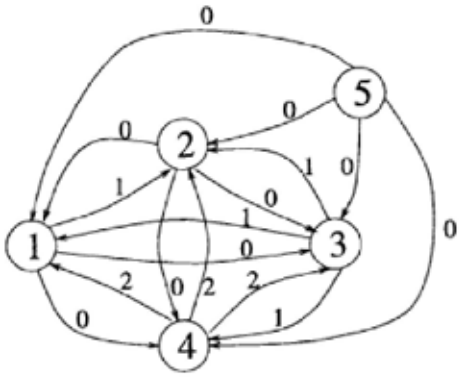
Feasibility constraint
r(1)-r(3) ≤ 1
r(1)-r(4) ≤ 2
r(2)-r(1) ≤ 1
r(3)-r(2) ≤ 0
r(4)-r(2) ≤ 0

Critical path constraint
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

Example

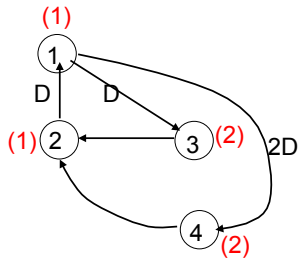
- Solve 12 inequalities
  - Construct constraint graph
  - Find weight matrix W of constraint graph, then solve using Bellman algorithm (Appendix A)

W=[0 1 0 0
0 0 0 0
1 1 0 1
2 2 2 0]



Example

- Solution from Bellman-Ford algorithm:
- r(1)=r(2)=r(3)=r(4)=0, no retiming needed. The graph already has a critical path =3



Redoing for c=2

Critical path constraints

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

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

Feasibility constraint
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

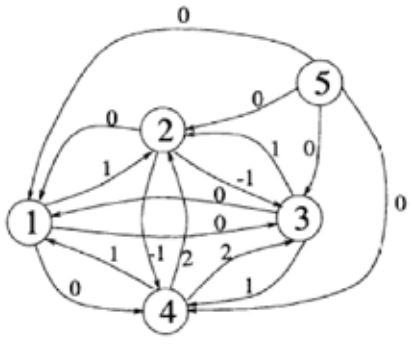
D(U,V)	1	2	3	4
1	1	4	3	3
2	2	1	4	4
3	4	3	2	6
4	4	3	6	2

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

For C=2

- Solve 12 inequalities
  - Construct constraint graph
  - Find weight matrix W of constraint graph, then solve using Bellman algorithm (Appendix A)

W=[0 1 0 0
0 0 -1 -1
0 0 0 1
1 2 2 0]



For c=2

- Bellman algorithm gives the following solution:

  r(2)=0, r(1)=r(3)=r(4)=-1