
DO NOT SHARE
SLIDES AND CLASS MATERIALS
ON ONLINE SITES
Course Hero

Advanced Logic Design
Retiming

Mingoo Seok

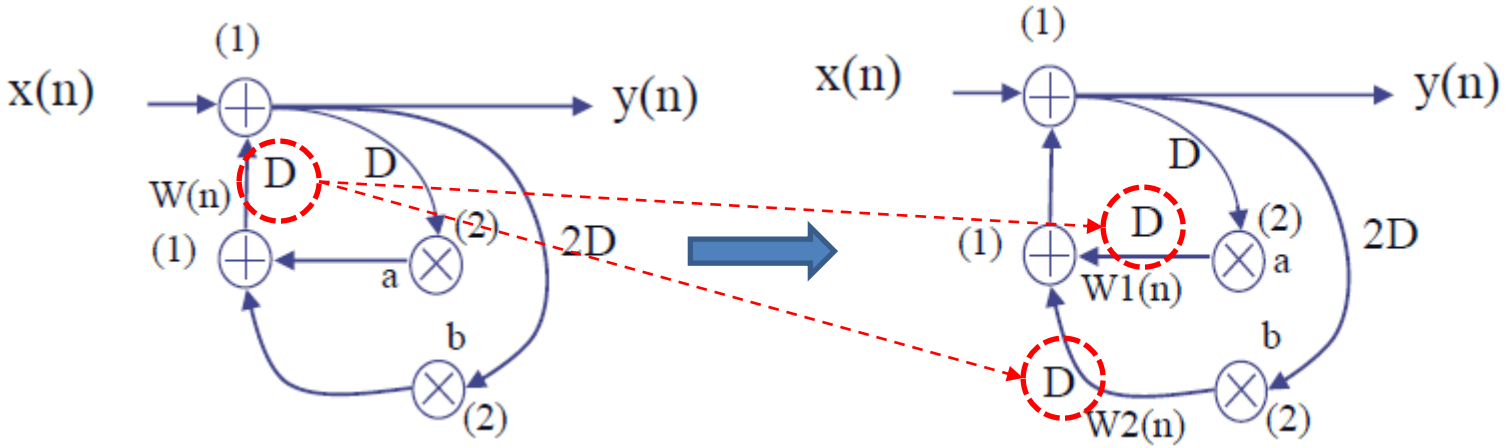
Columbia University

Reading: Parhi Sec. 4

Acknowledgement: Prof. Z. Zhang (UMich),
Prof. K. K. Parhi (UMinn)

Retiming

- Retiming: changing locations of delay elements without affecting the input/output characteristics
  - Reduce critical path and thus clock period
  - Reduce number of registers
  - Reduce power consumption by reducing switching: e.g., place registers at the inputs of nodes with large capacitances



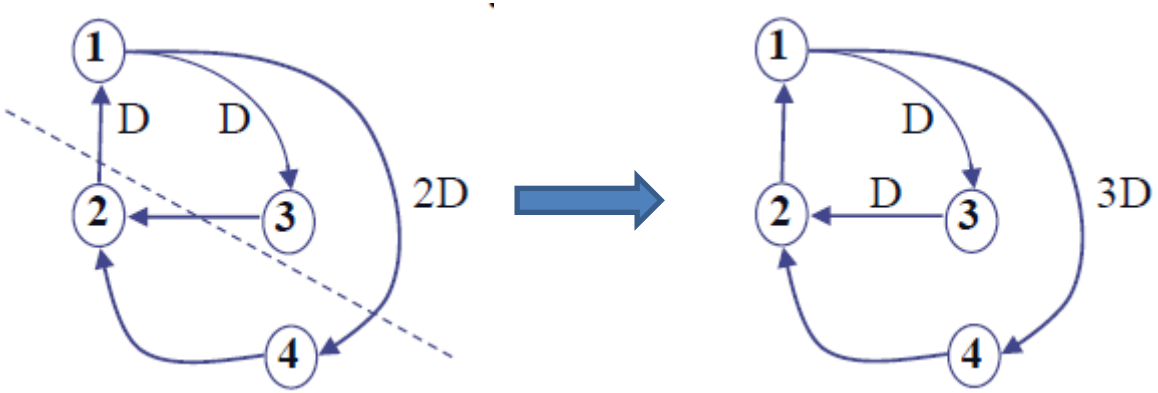
Cutset Retiming

- Recall
  - Cut: separate the vertices of a graph into two disjoint subsets: G1, G2
  - Cutset: the set of *edges* whose end points are in different subsets

- Cutset retiming – one of the retiming methods
  - Only affect edges in the cutset
  - **Add** *k* delays to each edge **from G1 to G2**
  - **Remove** *k* delays to each edge **from G2 to G1**
  - The weight (delay) of each edge of the retimed graph ≥ 0

Cutset Retiming

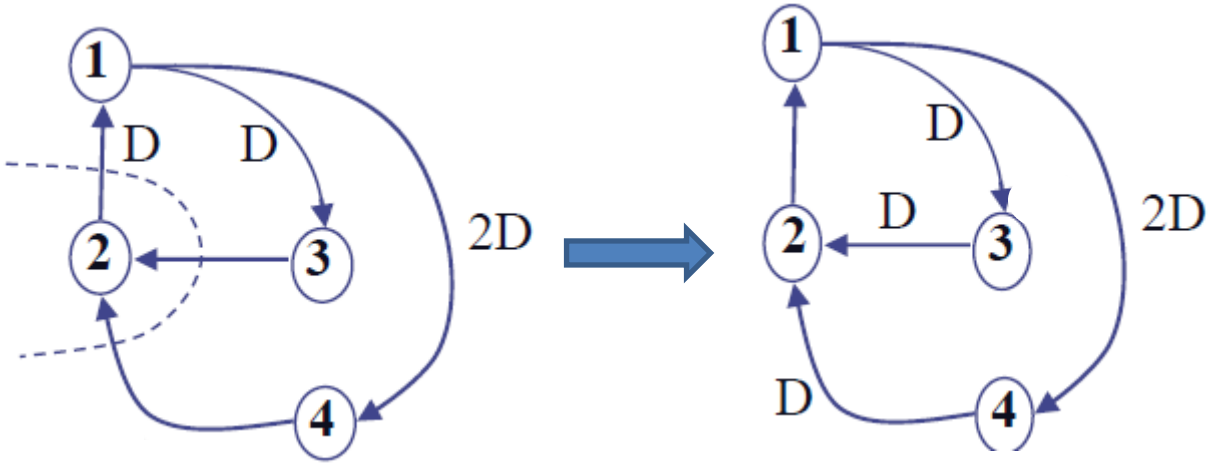
- Cutset retiming
  - Only affect edges in the cutset
  - Add *k* delays to each edge from G1 to G2
  - Remove *k* delays to each edge from G2 to G1
  - The weight (delay) of each edge of the retimed graph ≥ 0



  - *k is bounded by the original weight (delay) of cutset*
    - $w'(e_{1,2}) \geq 0 \Rightarrow w(e_{1,2}) + k \geq 0 \Rightarrow k \geq -w(e_{1,2}) \Rightarrow k \geq -\min\{w(e_{1,2})\}$
    - $w'(e_{2,1}) \geq 0 \Rightarrow w(e_{2,1}) - k \geq 0 \Rightarrow k \leq w(e_{2,1}) \Rightarrow k \leq \min\{w(e_{2,1})\}$

Special Case

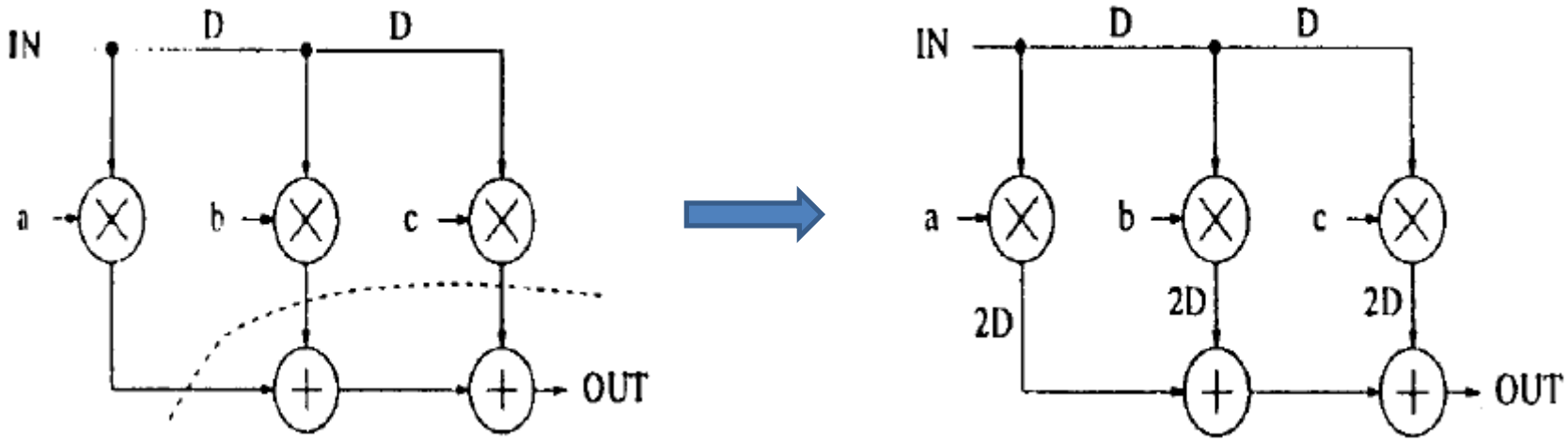
- Single node subgraph



  - Remove 1 delay from each outgoing edge and add 1 delay to each incoming edge

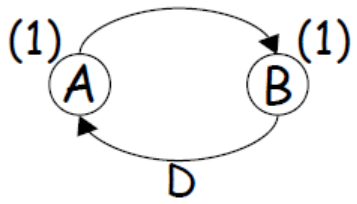
Special Case

- Pipelining

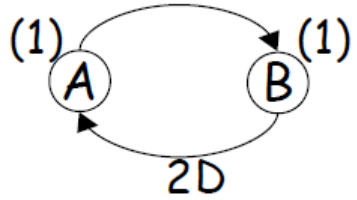


  - Edges only going one direction (feed-forward cutset), i.e., design has no loop
  - k is bounded in one direction only (i.e., large k can be used)

Slow Down

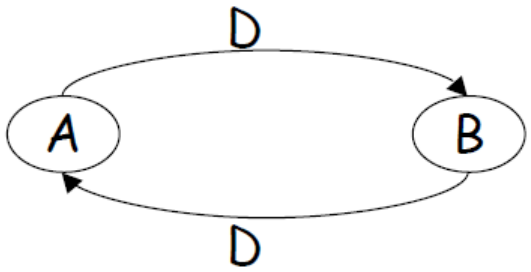


- 2-slow transformation
  - Replace each D by 2D
  - Input new samples every alternate cycle
  - Null operation used for odd clock cycle
  - Hardware utilized only 50% of the time



Clock	
0	A0→B0
1	
2	A1→B1
3	
4	A2→B2

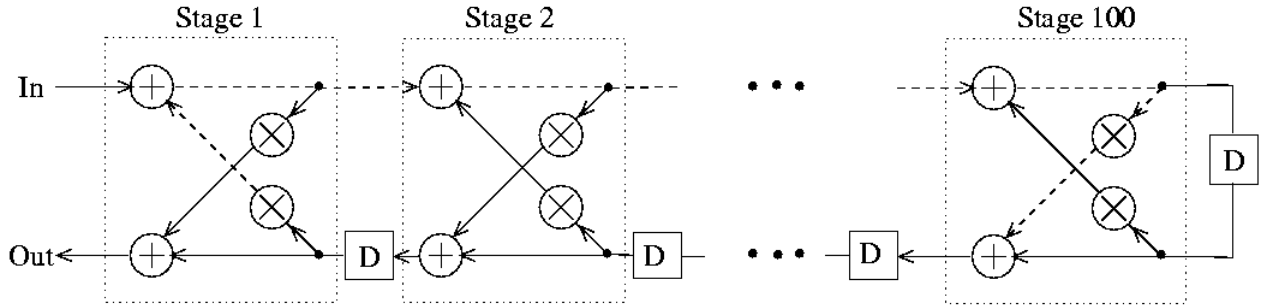
Retime



- Hardware utilization: 50%
- Critical path delay = 2 · 1 u.t. = 2 u.t.
- Can be fully utilized if two independent operations are available

Slow Down then Retiming

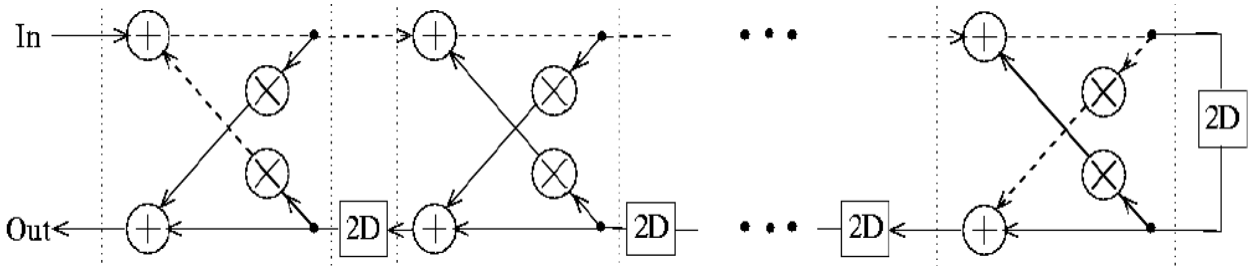
- 100-stage lattice filter



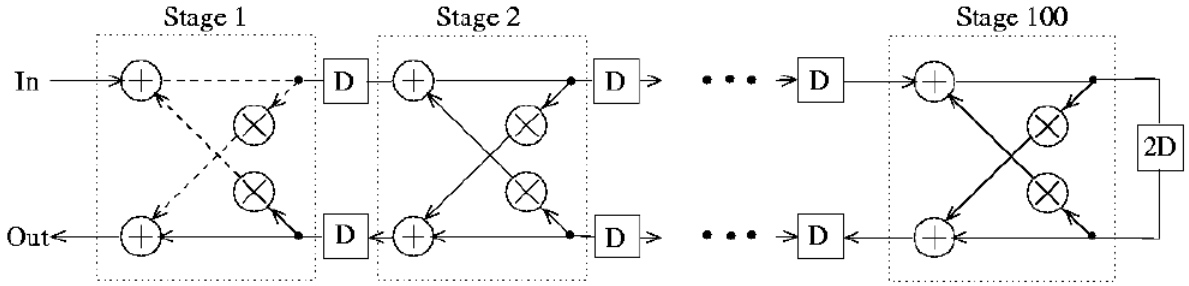
  - Critical path delay
  - Sample period

Slow Down then Retiming

- 2-Slow



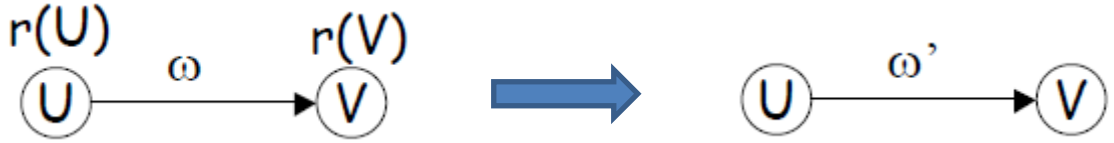
- Cutset retiming



  - New critical path delay:
  - New sample period:

Formulating Retiming

- Start with DFG
- Associate an integer value $r(V)$ with each node V
- Let $w(e)$ be the weight (or number of delays) of edge e
- Let $w'(e)$ be the weight of edge e in the retimed design



- The weight $w'(e)$ can be obtained by $w'(e) = w(e) + r(V) - r(U)$
  - Why?
- A retiming solution is feasible if $w'(e) \geq 0$ holds for all edges

Properties of Retiming

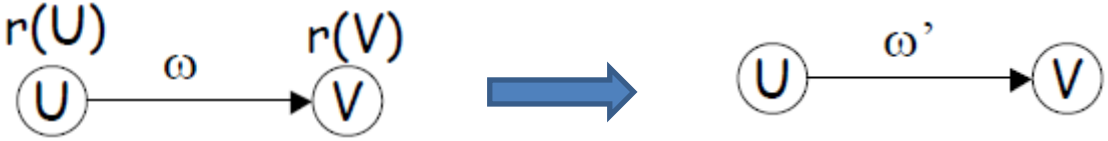
- Weight of a path: sum of delay elements along the path
- Computation time of a path: sum of computation time along the path

1. The weight of the retimed path $p = V_0 \rightarrow V_1 \rightarrow \ldots \rightarrow V_k$ is given by $w'(p) = w(p) + r(V_k) - r(V_0)$
2. Retiming does not change the number of delays in a loop
3. Retiming does not change the iteration bound in a DFG

- Retiming is formulated to meet the following goals
  - Minimize clock period → our focus
  - Minimize register count → Parhi book has a sub-chapter on this

Feasibility Constraint

- Recall formulating retiming
  - Associate an integer value $r(V)$ with each node V
  - $w(e)$: weight (or number of delays) of edge e
  - $w'(e)$: weight of edge e in the retimed design

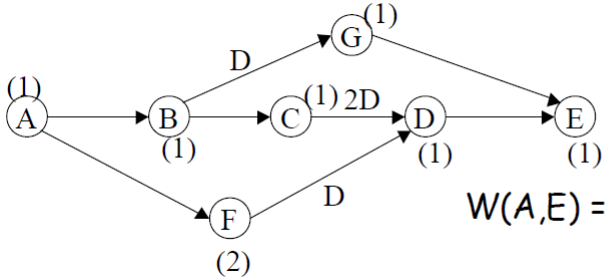


  - The weight $w'(e)$ can be obtained by $w'(e) = w(e) + r(V) - r(U)$
  - A retiming solution is feasible if $w'(e) \geq 0$ or $r(U) - r(V) \leq w(e)$

  - The feasibility constraint forces the number of delays on each edge in the retimed graph to be ***nonnegative***

Critical Path Constraint

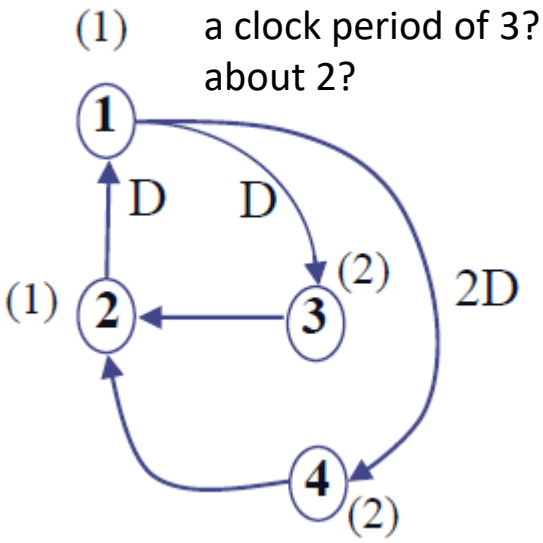
- Define:
  - $W(U,V)$ = min{$w(p)$: $U \rightarrow V$} (i.e., minimum delays from U to V)
  - $D(U,V)$ = max{$t(p)$: $U \rightarrow V$ and $w(p) = W(U,V)$} (i.e., maximum computation time of the path from U to V with the minimum delays)
  - c: target clock period



- **Q:** Does a retiming solution exist for a target clock period c?
  - If a path contains no delay, then the path computation time ≤ c
  - Contrapositive: if the path computation time > c, the path contains 1 or more delays, i.e., if $D(U,V) > c$ then $W(U,V) + r(V) - r(U) \geq 1$
  - The above defines the critical path constraint
  - Feasibility constraints: $r(U)-r(V) \leq w(e)$ for every edge U→V of G
  - Need to satisfy both (i) *the feasibility constraint* and (ii) *the critical path constraint* for a valid retiming solution

Example

Can we retime it to meet a clock period of 3? How about 2?



- Compute $W(U,V)$ and $D(U,V)$

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)	1	2	3	4
1	1	4	3	3
2	2	1	4	4
3	4	3	2	6
4	4	3	6	2

- Feasibility constraint and critical path constraint

$$r(1)-r(3)\le 1$$
$$r(1)-r(4)\le 2$$
$$r(2)-r(1)\le 1$$
$$r(3)-r(2)\le 0$$
$$r(4)-r(2)\le 0$$

One equation per a pair of *connected* nodes

$$r(1)-r(2)\le 0$$
$$r(1)-r(3)\le 0$$
$$r(1)-r(4)\le 1$$
$$r(2)-r(3)\le 1$$
$$r(2)-r(4)\le 2$$
$$r(3)-r(1)\le 0$$
$$r(3)-r(2)\le -1$$
$$r(3)-r(4)\le 2$$
$$r(4)-r(1)\le 0$$
$$r(4)-r(2)\le -1$$
$$r(4)-r(3)\le 1$$

- Solve the inequalities:

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

Check all pairs with computation time greater than target clock period

Modern CAD Tool Support

- Modern logic-synthesis CAD tools have retiming capability
- Example: Synopsis Design-Compiler
  - Related commands: optimize_registers, pipeline_design, balance_registers
- Explore this feature in your design project

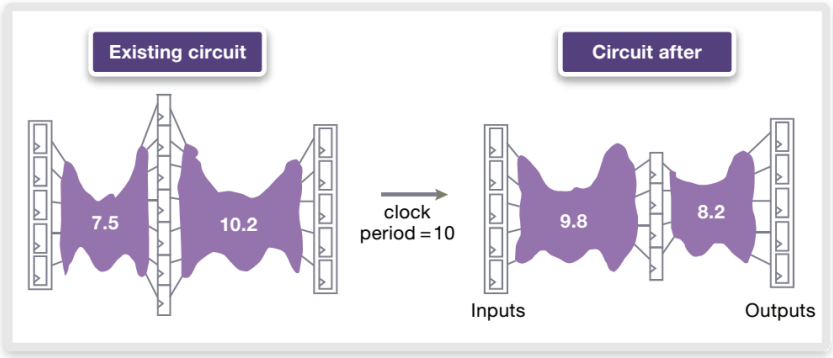


Figure 6: Retiming designs with registers

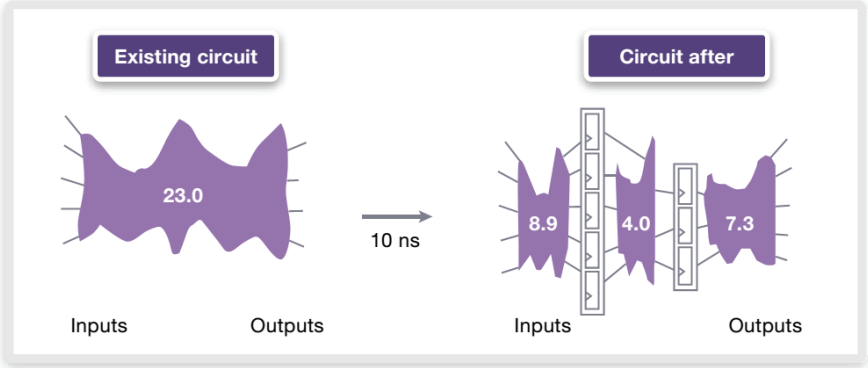


Figure 7: Retiming on con

[www.synapsis. com]