

$$\sum_{k=0}^{n/2} (m'_{2k-1} + k_{2k-1})(m'_{2k} + k_{2k})x^k$$

ECE 699—Digital Signal Processing Hardware Implementations

Lecture 9

Retiming Transformations

4/8/09

1

Outline

- Retiming Introduction
- Preliminaries
 - Quantitative Description
 - Properties of Retiming
 - Solving systems of inequalities
- Special Cases
 - Cutset Retiming
 - Pipelining
- Uses of Retiming
 - Retiming for Clock Period Minimization
 - Retiming for Register Minimization

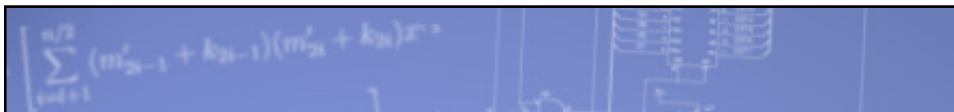
2

Reading

- Retiming
 - Parhi, VLSI Digital Signal Processing Systems
 - Chapter 3
 - Appendix A

3

$$\sum_{i=0}^{n/2} (m'_{2i-1} + k_{2i-1})(m'_{2i} + k_{2i})x^{i-1}$$

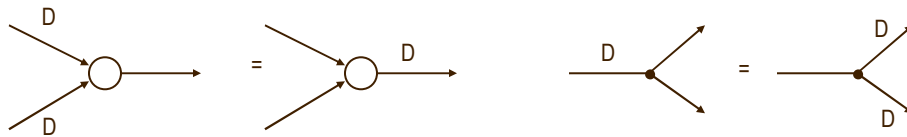


Retiming

4

Retiming Introduction

- Retiming moves around registers which already exist in the system
 - Retiming does not alter the latency in the system
 - Retiming does not change the input/output characteristics
 - Retiming DOES change the critical path of the system and/or the number of registers in the system
- Uses the primary rules:



5

Retiming Uses

- Retiming used:
 - 1) to decrease minimum clock period of a circuit (i.e. faster)
 - 2) to reduce number of registers of a circuit (i.e. smaller)
 - 3) for logic synthesis (not covered in class)
 - 4) for low power CMOS circuits

6

Quantitative Description of Retiming

- Retiming maps circuit G to a retimed circuit G_r
- Retiming solution characterized by a value $r(V)$ for each node V in graph
 - Let $w(e)$ denote weight of edge e of graph G , and $w_r(e)$ denote weight of edge e of graph G_r
 - Weight of edge e from $U \rightarrow V$ in the retimed graph is computed from weight of edge in original graph using
$$w_r(e) = w(e) + r(V) - r(U)$$
- Retiming solution is feasible if $w_r(e) \geq 0$ for all edges

7

Properties of Retiming

- Weight of a path from node 0 to node k is number of delays between those nodes $w(p) = \sum_{i=0}^{k-1} w(e_i)$
- Computation time of a path between node 0 to node k is the sum of computation times (adders, etc.) of each of the nodes $t(p) = \sum_{i=0}^k t(V_i)$
- Properties:
 - Retiming does not change number of delays in a cycle
 - Retiming does not alter iteration bound of DFG
 - Adding a constant value j to the retiming value of each node does not change the mapping from G to G_r

8

Solving Systems of Inequalities

- Shortest path algorithms (Appendix A of Parhi book)
 - Bellman-Ford
 - Floyd-Warshall
- Given a set of M inequalities and N variables, where each inequality has the form $r_i - r_j \leq k$ for integer values of k , can use one of shortest path algorithms to determine if solution exists and to find one solution
- Procedure:
 - 1) Draw the constraint graph
 - a) Draw the node i for each of the N variables $r_i, i=1, \dots, N$
 - b) Draw the node $N+1$
 - c) For each inequality $r_i - r_j \leq k$, draw the edge $j \rightarrow i$ for node j to node i with length k
 - d) For each node $i, i=1, 2, \dots, N$, draw the edge $N+1 \rightarrow i$ from the node $N+1$ to the node i with length 0
 - 2) Solve using a shortest path algorithm
 - a) the system of equalities has a solution if and only if the constraints graph contains no negative cycles
 - b) if a solution exists, one solution is where r_i is the minimum-length path from the node $N+1$ to the node i

9

Cutset Retiming

- Two special cases of retiming exist:
 - Cutset retiming
 - Pipelining : pipelining can be considered as adding a number of registers in the front of the DFG **and then** doing retiming on these new registers
- Cutset retiming
 - Cutset = set of edges that can be removed from graph to create 2 disconnect subgraphs
 - Cutset retiming only affects the weights of the edges in the cutset.
 - If 2 disconnected subgraphs are G_1 and G_2 then cutset retiming consists of adding k delays to each edge from G_1 to G_2 and removing k delays from each edge from G_2 to G_1
 - Cutset retiming is a special case of retiming where each node in the graph G_1 has the retiming value j and each node in the subgraph G_2 has the retiming value $j+k$ (j is arbitrary)
 - Remember: Retiming solution is feasible only if $w_r(e) \geq 0$ for all edges

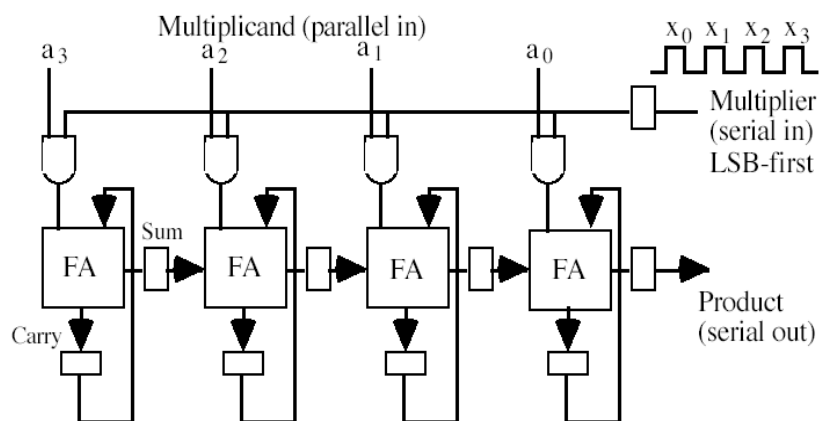
10

Cutset Retiming Example: Systolic Array Multiplier

- Systolic array: synchronous arrays of processing elements that are interconnected by only short, local wires thus allowing very high clock rates

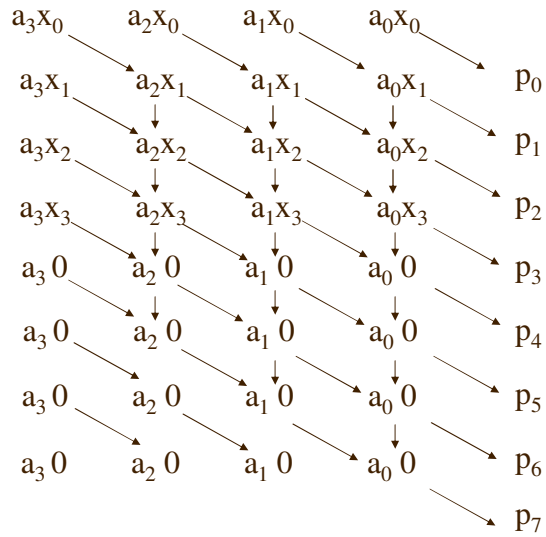
11

Semisystolic Bit-Serial Multiplier (1)



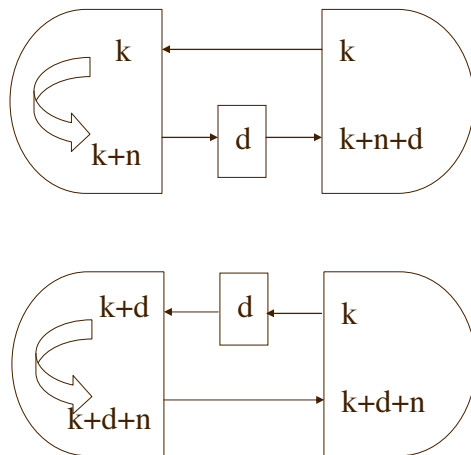
12

Semisystolic Bit-Serial Multiplier (2)



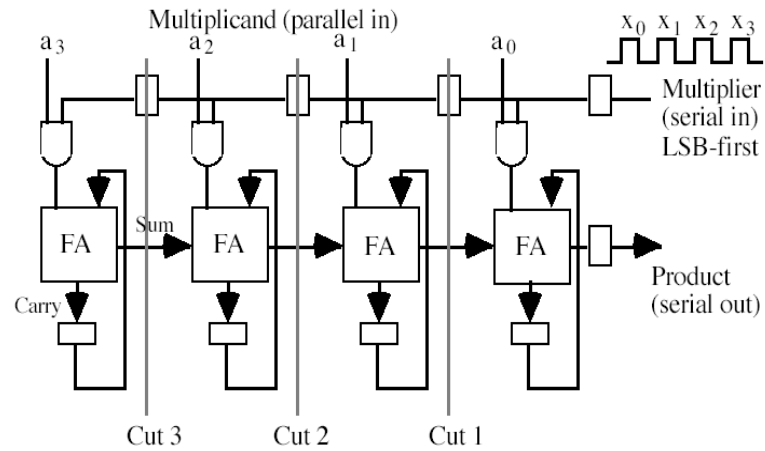
13

Cutset Retiming



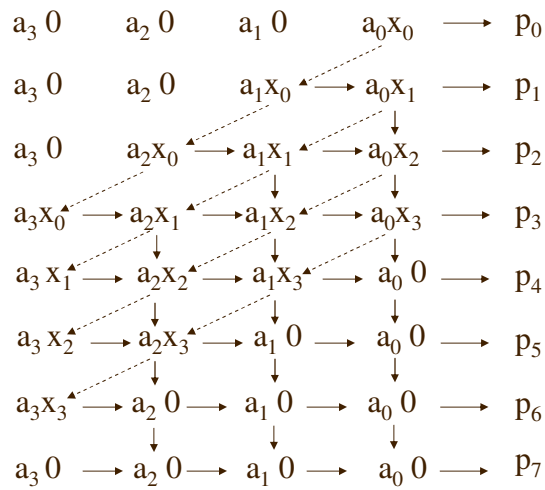
14

Retimed Semisystolic Bit-Serial Multiplier (1)



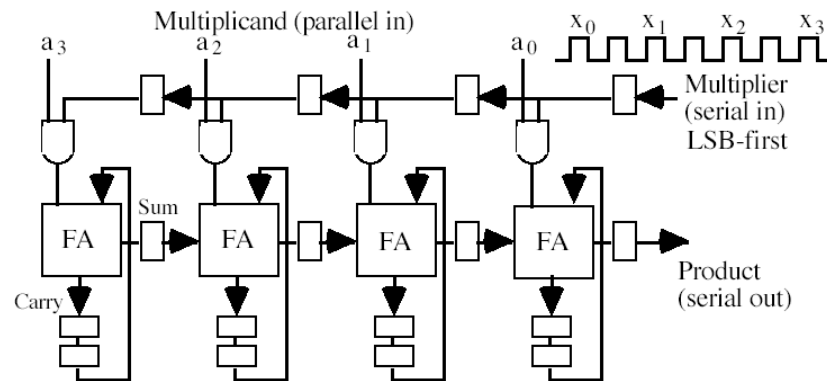
15

Retimed Semisystolic Bit-Serial Multiplier (1)



16

Systolic Bit-Serial Multiplier



17

Pipelining

- Pipelining is a special case of cutset retiming where
 - Edges go from G1 to G2
 - No edges go from G2 to G1 (i.e. no loops, feedforward only)
 - In this case can add as many registers on the cutset as desired

18

Cutset Retiming and Slow-Down

- Cutset retiming often used in combination with slow-down
- Replace each delay in the DFG with N delays to create an N-slow version
 - This requires N-1 null operations to be interleaved

19

Retiming for Clock Period Minimization

- In previous lectures, we have learned to calculate the iteration bound of a DFG
 - Iteration bound determines the minimum clock period of a recursive DFG
- Retiming for clock period minimization is the tool used to cause a recursive DFG to have a clock period to equal the iteration bound

20

Retiming for Clock Period Minimization cont'd

- Minimum feasible clock period is computation time of the critical path, which is the path with the longest computation time among all paths with no delays. Minimum clock period is $\Phi(G)$

$$\Phi(G) = \max\{t(p) : w(p) = 0\}$$

- Want to find a retiming solution $\Phi(G_{r_0}) \leq \Phi(G_r)$ for any other retiming solution r . In other words, we want to find the retiming solution with minimum clock period

- Nomenclature:

- $W(U, V)$ = minimum numbers of registers on any path from node U to V

$$W(U, V) = \min\{w(p) : U \xrightarrow{p} V\}$$

- $D(U, V)$ = maximum computation time among all paths from U to V with weight $W(U, V)$

$$D(U, V) = \max\{t(p) : U \xrightarrow{p} V \text{ and } w(p) = W(U, V)\}$$

21

Algorithm for Retiming for Clock Period Minimization

- Algorithm for retiming for clock period minimization
- First construct $W(U, V)$ and $D(U, V)$
 - 1) Let $M = t_{\max} \cdot n$ where t_{\max} is the maximum computation time of the nodes in G and n is the number of nodes in G .
 - 2) Form a new graph G' which is the same as G except the edge weights are replaced by $w'(e) = Mw(e) - t(U)$ for all edges e for $U \rightarrow V$
 - 3) Solve the all-pairs shortest path problem on G' (using Floyd-Warshall, for example). Let S'_{UV} be the shortest path from U to V .
 - 4) If $U \neq V$, then $W(U, V) = \text{ceil}(S'_{UV}/M)$ and $D(U, V) = MW(U, V) - S'_{UV} + t(V)$. If $U=V$, then $W(U, V) = 0$ and $D(U, V) = t(U)$. $\text{Ceil}()$ is the ceiling function.
- Use $W(U, V)$ and $D(U, V)$ to determine if there is a retiming solution that can achieve a desired clock period c .
 - **Usually set this desired clock period equal to the iteration bound of the circuit.**

22

Algorithm for Retiming for Clock Period Minimization cont'd

- Given a desired clock period c , there is a feasible retiming solution r such that $\Phi(G_r) \leq c$ if the following constraints hold
 - CONSTRAINT 1: (feasibility) $r(U) - r(V) \leq w(e)$ for every $U \rightarrow V$ along edge e of G
 - This enforces the numbers of delays on each edge in the retimed graph to be nonnegative
 - CONSTRAINT 2: (critical path) $r(U) - r(V) \leq W(U,V) - 1$ for all vertices U, V , in G such that $D(U,V) > c$
 - This enforces $\Phi(G_r) \leq c$
- Thus, to find a solution
 - 1) pick a value of c (usually equal to iteration bound)
 - 2) Create a series of inequalities based on the feasibility constraint.
 - 3) Create a series of inequalities based on the critical path constraint.
 - 4) Combine these (using most restrictive if overlap exists) and create a constraint graph.
 - 5) Find feasibility using shortest-path algorithm (i.e. Floyd-Warshall) and find retiming values

23

Retiming for Register Minimization

- Derived in class

24