

EDA (CS286.5b)

Day 19
Covering and Retiming

“Final”

- Like Assignment #1
 - longer
 - more breadth
 - focus since assignment #2
 - ...but ideas are cumulative
 - open notes, books, etc.
- Make available before 12:01am Mon. 3/13
- Due 11:59pm Wed. 3/15

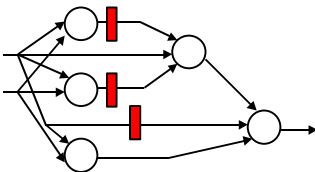
Previously

- Cover (map) LUTs for minimum delay
 - day 3
 - solve optimally
- Retiming for minimum clock period
 - day 18
 - solve optimally
- Simultaneous Cover and 1D placement
 - day 9
 - optimal area cover for trees

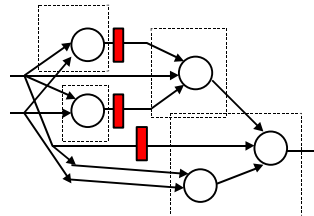
Today

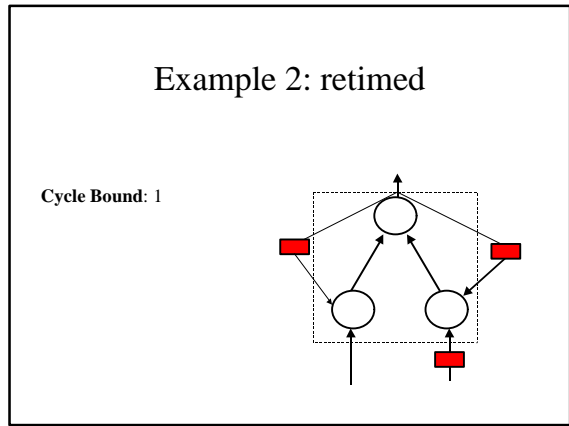
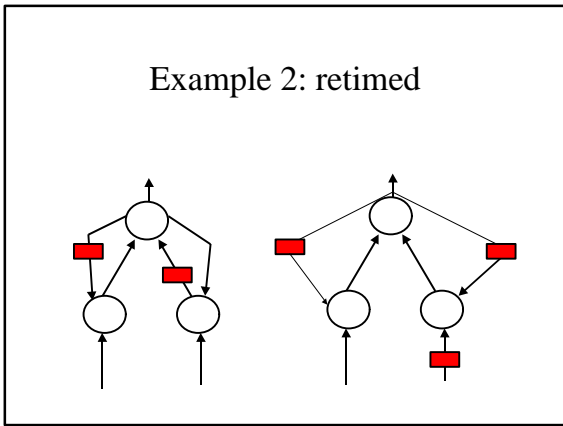
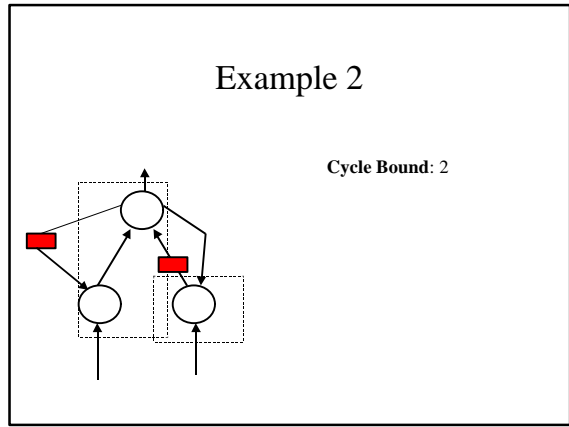
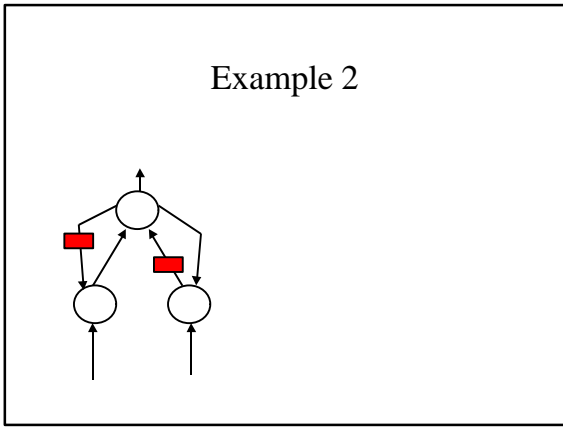
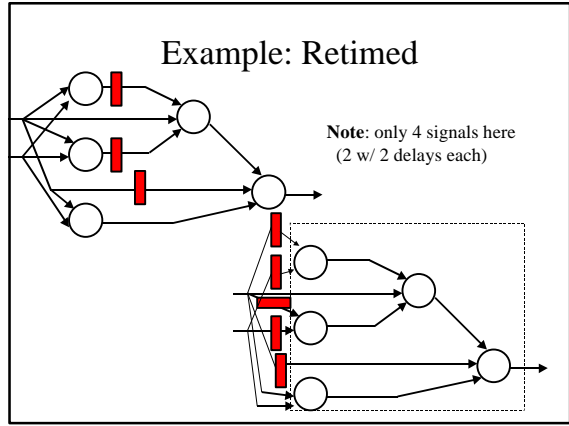
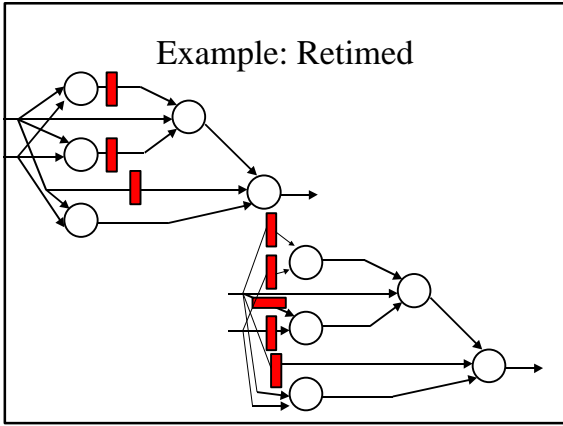
- Solving cover/retime separately not optimal
- Cover+retime
- Review

Example



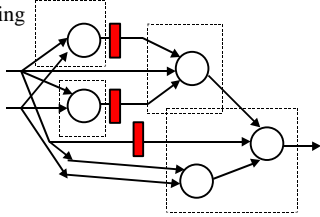
Example





Basic Observation

- Registers break up circuit, limiting coverage
 - fragmentation
 - prevent grouping

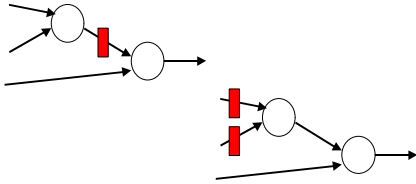


Phase Ordering Problem

- General problem we've seen before
 - e.g. placement
 - don't know where connected neighbors will be if unplaced...
 - don't know effect/results of other mapping step
- Here
 - don't know delay (what can be packed into LUT) if retime first
 - not retime first
 - fragmentation: forced breaks at bad places

Observation #1

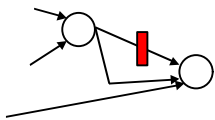
- Retiming flops to input of (fanout free) subgraph is trivial (and always doable)



Observation #1: Consequence

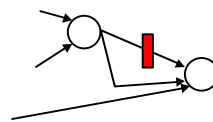
- Can cover *ignoring* flop placement
- Then retime LUTs to input

Fanout Problem?

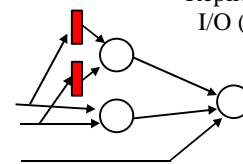


Can I use the same trick here?

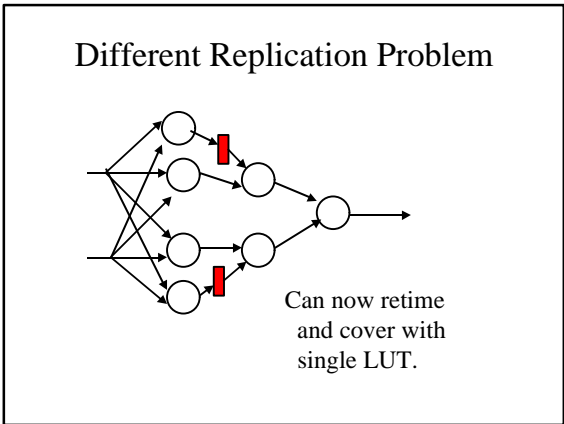
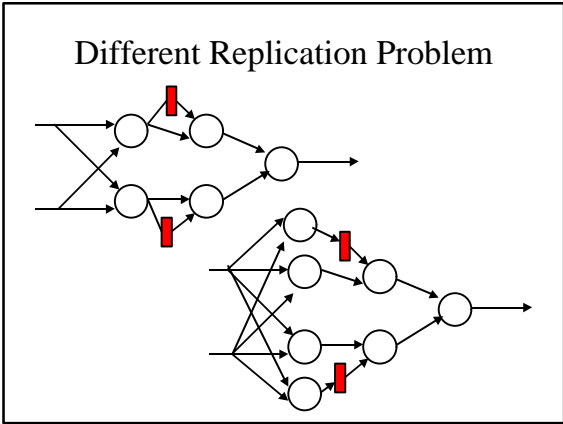
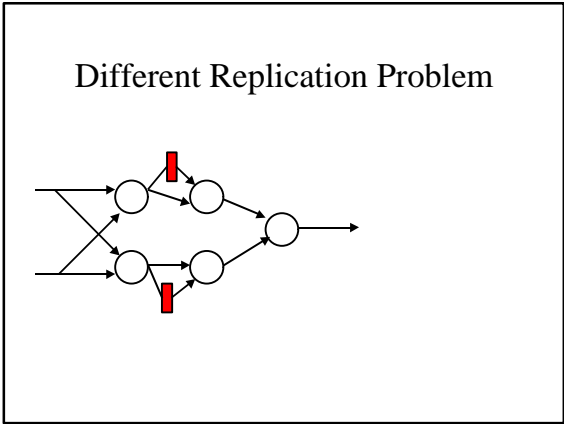
Fanout Problem?



Cannot retime without replicating.



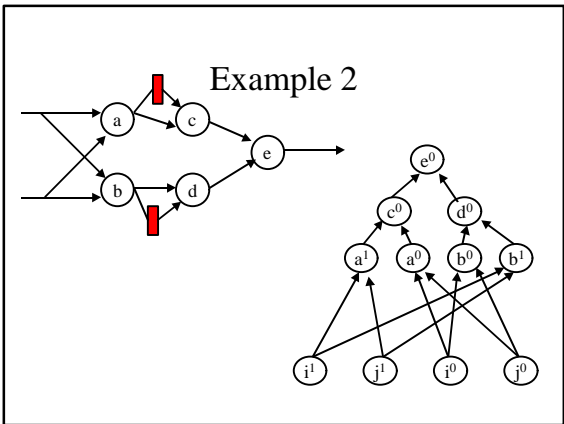
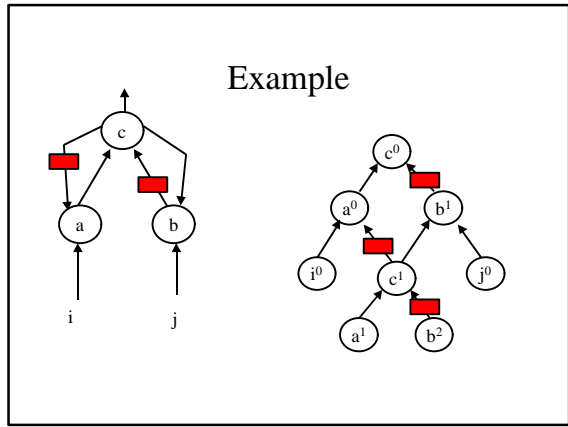
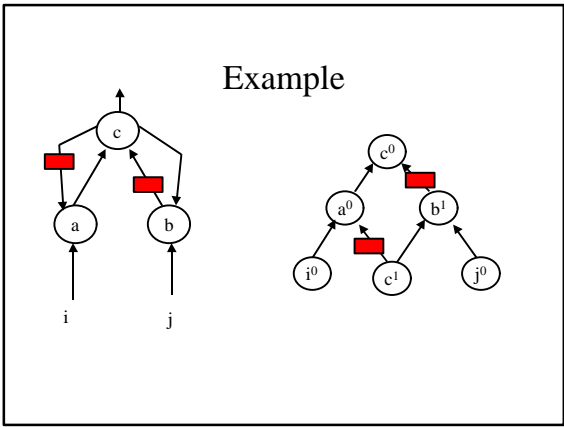
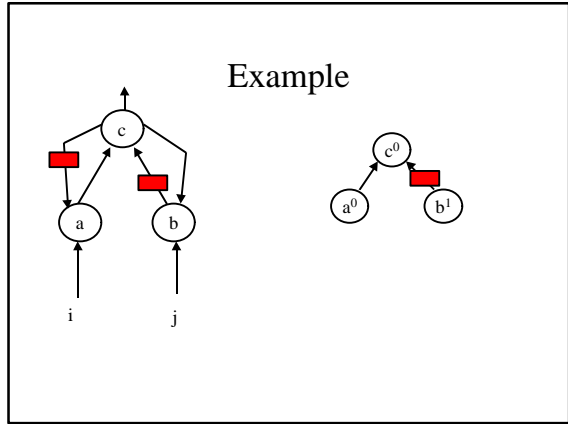
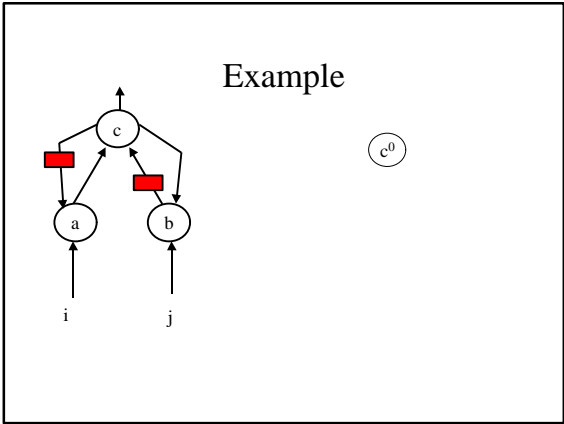
Replicating increase I/O (so cut size).



- ### Replication
- Once add registers
 - can't just grab max flow and get replication
 - (compare flowmap)
 - Or, can't just ignore flop placement when have reconvergent fanout through flop

- ### Replication
- Key idea:
 - represent timing paths in graph
 - differentiating based on number of registers in path
 - **new graph**: all paths from node to output have same number of flip-flops
 - label nodes u^d where d is flip-flops to output

- ### Deal with Replication
- *Expanded Graph*:
 - start with target output node
 - for each input u to current expanded graph
 - grab it's input edge $(x @ u)$ with weight $(w(e))$
 - add node $x^{(d+w(e))}$ to graph (if necessary)
 - add edge $x^{(d+w(e))} @ u^d$ with weight $(w(e))$
 - continue breadth first until have enough
 - enough for flow cut
 - at most $k * n$ node depth required



Expanded Graph

- Expanded graph does not have fanout of different flip-flop depths from the *same* node.
- Can now cover ignoring flip-flops and trivially retime.

Labeling

- Key idea #1:
 - compute distances/delay like flowmap
 - dynamic programming
- Key idea #2:
 - count distance from register (like $G-1/c$ graph)

Labeling: Edge Weights

- To target clock period c
 - use graph $G-1/c$
 - paper:
 - assign weight $-c*w(e)+1$
 - (same thing scaled by c and negated)

Labeling: Edge Weight Idea

- same idea:
 - will need register ever c LUT delays
 - credit with registers as encounter
 - charge a fraction $(1/c)$ every LUT delay
 - know net distance at each point
 - if negative (delays $> c*$ registers)
 - cannot distribute to achieve c
 - otherwise
 - labeling tells where to distribute

Labeling: Flow cut

- Label node as before (flowmap)
 - $L(v)=\min\{l(u)+w(e)\mid u@v\}$
 - trivially can be $L(v)-1/c ==$ new LUT
 - note min vs. max and $-1/c$ vs. $+1$ due to rescaling to match retiming formulation and $G-1/c$ graph
 - in this formulation, a combinational circuit of depth 4 would have $L(v)=-4/c$
 - if can put this and all $L(v)$'s in one LUT
 - this can be $L(v)$
 - construct and compute flow cut to test

LUT Map and Retime

- Start with outputs
- Cover with LUT based on cut
 - move flip-flops to inputs of LUT
- Recursively cover inputs
- Use label to retime
 - $r(v)=\lceil l(v) \rceil + 1/c$

Target Clock Period c

- As before (retiming)
 - binary search to find optimal c

Variations

- Relaxation/Iteration
 - original computed labels iteratively
- Flow cover
 - Cong+Wu/ICCAD96 showed can use flowmap-style min-cut
- Find all k-cuts first
 - Pan+Liu/FPGA'98

Summary

- Can optimally solve
 - LUT map for delay
 - retiming for minimum clock period
- But, solving separately does not give optimal solution to problem
- Account for registers on paths
- Label based on register placement and (flow) cover ignoring registers
- Labeling gives delay, covering, retiming

Today's Big Ideas

- Exploit freedom
- Cost of decomposition
 - benefit of composite solution
- Technique:
 - dynamic programming
 - network flow

This Class: Decomposition

- Scheduling
- Logic Optimization
 - sequential, two-level, multi-level
- Covering/gate-mapping
- Retiming
- Partitioning
- Placement
- Routing

This Class: Techniques

- Dynamic Programming
- Linear Programming (LP, ILP)
- Graph Algorithms
- Greedy Algorithms
- Randomization
- Search
- Heuristics
- Approximation Algorithms
- Iterative/Relaxation

Spring Quarter

- Primarily Project
 - select problem
 - formulate
 - survey attacks, similar problems, brainstorm
 - implement
 - benchmark/analyze
- Few lectures on additional topics
 - most geared to projects
 - time to discuss, present project