Today

- Retiming
  - Cycle time (clock period)
  - Initial states
  - Register minimization

Task

- Move registers to:
  - Preserve semantics
  - Minimize path length between registers
    - Reduce cycle time
    - …while minimizing number of registers required

Example: Same Semantics

- Externally: no observable difference

Problem

- Given: clocked circuit
- Goal: minimize clock period without changing (observable) behavior
- I.e. minimize maximum delay between any pair of registers
- Freedom: move placement of internal registers
Other Goals

• Minimize number of registers in circuit
• Achieve target cycle time
• Minimize number of registers while achieving target cycle time

• ...start talking about minimizing cycle...

Legal Register Moves

• Retiming Lag/Lead

Critical Path Length

Critical Path: Length of longest node path of zero weight edges

Preclass 2 Example

Path Length (L) ?

Can we do better?

Canonical Graph Representation

Separate arc for each path

Weight edges by number of registers
(weight nodes by delay through node)

Retiming Lag/Lead

Retiming: Assign a lag to every vertex

weight(e') = weight(e) + lag(head(e)) - lag(tail(e))
Valid Retiming

- Retiming is valid as long as:
  - \( \forall e \text{ in graph} \):
    - weight(e') = weight(e) + lag(head(e))-lag(tail(e)) \( \geq 0 \)
  - Assuming original circuit was a valid synchronous circuit, this guarantees:
    - non-negative register weights on all edges
      - no travel backward in time :-)
    - all cycles have strictly positive register counts
    - propagation delay on each vertex is non-negative (assumed 1 for today)

Retiming Task

- Move registers = assign lags to nodes
  - lags define all locally legal moves
- Preserving non-negative edge weights
  - (previous slide)
  - guarantees collection of lags remains consistent globally

Retiming Transformation

- Properties invariant to retiming
  1. number of registers around a cycle
  2. delay along a cycle

- Cycle of length \( P \) must have
  - at least \( P/c \) registers on it to be retimeable to cycle \( c \)
  - Can be computed from invariant above

Optimal Retiming

- There is a retiming of
  - graph \( G \)
  - \( w/ \) clock cycle \( c \)
  - iff \( G-1/c \) has no cycles with negative edge weights

- \( G-\alpha = subtract \alpha \) from each edge weight

1/c Intuition

- Want to place a register every \( c \) delay units
- Each register adds one
- Each delay subtracts 1/c
- As long as remains more positives than negatives around all cycles
  - can move registers to accommodate
  - Captures the reg = \( P/c \) constraints
Illustrate with Pipeline Case

\[ G-1/c \]

Compute Retiming
- \( \text{Lag}(v) = \text{shortest path to I/O in } G-1/c \)
- Compute shortest paths in \( O(|V||E|) \)
  - Bellman-Ford
  - also use to detect negative weight cycles when \( c \) too small

Bellman Ford
- For \( l \leftarrow 0 \) to \( N \)
  - \( u_i \leftarrow \infty \) (except \( u_i = 0 \) for IO)
- For \( k \leftarrow 0 \) to \( N \)
  - for \( e_{ij} \in E \)
    - \( u_i \leftarrow \min(u_i, u_j + w(e_{ij})) \)
- For \( e_{ij} \in E \), //still update \( \rightarrow \) negative cycle
  - if \( u_i > u_j + w(e_{ij}) \)
    - cycles detected

Apply to Example

Try \( c = 1 \)

Draw \( G-1 \)
Negative cycles?
Try $c=2$

Draw $G-0.5$
Negative cycles?

Apply: Find Lags
Shortest paths?

Apply: Lags

Apply: Lags
• Take ceil

Phase Choice ($C=2$)

Apply: Move Registers
Compute new weights

weight($e'$) = weight($e$) + lag(head($e$))-lag(tail($e$))
Apply: Retimed Design

Apply: Lags (alternate)

• Take floor

Apply: Move Registers (floor)

weight(e') = weight(e) + lag(head(e)) - lag(tail(e))

Apply: Retimed Design (floor)

Compute new weights

Summary So Far

• Can move registers to minimize cycle time
• Formulate as a lag assignment to every node
• Optimally solve cycle time in O(|V||E|) time
  – Using a shortest path search

Questions?
Pipelining

- We can use this retiming to pipeline
- Assume we have enough (infinite supply) registers at edge of circuit
- Retime them into circuit

\[ C > 1 \rightarrow \text{Pipeline} \]

Add Registers

\[ n - \text{regs} \]

Draw G

<table>
<thead>
<tr>
<th>Setup ( G-1/c )</th>
<th>Minimum ( n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( G )</td>
<td>( G-1/1 )</td>
</tr>
</tbody>
</table>

Lags?

\[ n = 5 \]

Pipeline Retiming: Lag
Move Registers

Compute new weights (move registers)

Note

- Algorithm/examples shown
  - for special case of unit-delay nodes
- For general delay,
  - a bit more complicated
  - still polynomial

Initial State

What should initial value be?

In general, constraints \( \rightarrow \) satisfiable?
Initial State

- Cannot always get exactly the same initial state behavior on the retimed circuit
  - without additional care in the retiming transformation
  - sometimes have to modify structure of retiming to preserve initial behavior
- Only a problem for startup transient
  - if you're willing to clock to get into initial state, not a limitation

Minimize Registers

- Number of registers: $\sum w(e)$
- After retime: $\sum w(e) + \sum (F_l(v)-F_o(v))\text{lag}(v)$
- delta only in lags
- So want to minimize: $\sum (F_l(v)-F_o(v))\text{lag}(v)$
  - subject to earlier constraints
    - non-negative register weights, delays
    - positive cycle counts
- $F_l(v)-F_o(V)$ is a constant $c_v$
  - Minimize $\sum (c_v\text{lag}(v))$
  - $w(e_i) + \text{lag(\text{head}(e_i))} - \text{lag(\text{tail}(e_i))} > 0$

Minimize Registers $\rightarrow$ ILP

- So want to minimize: $\sum (F_l(v)-F_o(v))\text{lag}(v)$
  - subject to earlier constraints
    - non-negative register weights, delays
    - positive cycle counts

Minimize Registers: ILP → flow

- Can be formulated as flow problem
- Can add cycle time constraints to flow problem
- Time: $O(|V||E|\log(|V|)\log(|V|^2/|E|))$

**Summary**

- Can move registers to minimize cycle time
- Formulate as a lag assignment to every node
- Optimally solve cycle time in $O(|V||E|)$ time
- Also
  - Minimize registers
  - Watch out for initial values

**Big Ideas**

- Exploit freedom
- Formulate transformations (lag assignment)
- Express legality constraints
- Technique:
  - graph algorithms
  - network flow

**Admin**

- Reading for Wednesday online