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...

Preclass 2 Example

Legal Register Moves

- Retiming Lag/Lead

Canonical Graph Representation

Critical Path Length

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

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:
  - ∀e in graph
    - weight(e') = weight(e) + lag(head(e))-lag(tail(e)) ≥ 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 retimable 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

  \[ G-\alpha = \text{subtract } \alpha \text{ from each edge weight} \]

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 regs=P/c constraints
Illustrate with Pipeline Case

Compute Retiming
- $\text{Lag}(v) =$ 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 $i \leftarrow 0$ to $N$
  - $u_i \leftarrow \infty$ (except $u_i = 0$ for IO)
- For $k \leftarrow 0$ to $N$
  - for $e_{ij} \in E$
    - $u_i \leftarrow \min(u_i, u_j + w(e_{ij}))$
  - if $u_i > u_j + w(e_{ij})$
    - Cycles detected

Try $c=1$

Apply to Example

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

Negative cycles?

Apply: Find Lags

Shortest paths?

Apply: Lags

• Take ceil

Apply: Move Registers

Compute new weights

\[
\text{weight}(e') = \text{weight}(e) + \text{lag(head}(e)) - \text{lag(tail}(e))
\]
Apply: Retimed Design

Apply: Lags (alternate)
• Take floor

Apply: Move Registers (floor)

Apply: Retimed Design (floor)

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?
Note

- Algorithm/examples shown
  - for special case of unit-delay nodes
- For general delay,
  - a bit more complicated
  - still polynomial
- May not achieve P/c lower bound due to indivisible blocks
  - Example: blocks of delay 2.1 and 1.9 w c=2
  - More general: 0.9, 1.3, 0.8, 1.1

Initial State

- What about initial state?

Initial State

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

Initial State

What should initial value be?

Initial State

What should initial value be?

Initial State

What should initial values be?

Initial State

What should initial values be?

Initial State

What should init be?
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: $\sum w(e) + \sum (FI(v) - FO(v))\text{lag}(v)$
- only in lags
- So want to minimize: $\sum (FI(v) - FO(v))\text{lag}(v)$
  - subject to earlier constraints
    - non-negative register weights, delays
    - positive cycle counts

Minimize Registers $\rightarrow$ ILP

- So want to minimize: $\sum (FI(v) - FO(v))\text{lag}(v)$
  - subject to earlier constraints
    - non-negative register weights, delays
    - positive cycle counts
- $FI(v) - FO(V)$ is a constant $c_v$
  - Minimize $\sum (c_v \cdot \text{lag}(v))$
  - $w(e_i) + \text{lag(head(e_i))} - \text{lag(tail(e_i))} > 0$

Minimize Registers: ILP $\rightarrow$ flow

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

Retiming and Covering

Time Permitting
Issue

- Cover (map) LUTs for minimum delay
  - solve optimally for delay \( \rightarrow \) flowmap
- Retiming for minimum clock period
  - solve optimally
- …but, solving cover/retime separately **not** optimal
- We can formulate joint optimization

Phase Ordering Problem

- General problem
  - don’t know effect of other mapping step
  - Have seen this many places
- Here
  - don’t know delay if retime first
  - don’t know what can be packed into LUT
  - If we do not retime first
  - fragmentation: forced breaks at bad places

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
- Projects due Wednesday
- Need all work in by end-of-finals
  - May 12th