ESE535: Electronic Design Automation

Day 23: April 13, 2011
Retiming

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

Path Length (L)?

Can we do better?

Legal Register Moves

• Retiming Lag/Lead

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 $G-\alpha$ 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
Compute Retiming

- \( \text{Lag}(v) = \text{shortest path to I/O in } G - \frac{1}{c} \)
- 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 \( u_0 = 0 \) for IO)
- For \( k \leftarrow 0 \) to \( N \)
  - for \( e_{i,j} \in E \)
    - \( u_i \leftarrow \min(u_i, u_j + w(e_{i,j})) \)
  - for \( e_{i,j} \in E \) //still update \( \rightarrow \) negative cycle
    - if \( u_i > u_j + w(e_{i,j}) \)
      - cycles detected

Apply to Example

Try \( c = 1 \)

Try \( c = 2 \)

Apply: Find Lags

Shortest paths?
Apply: Lags

Apply: Lags

- Take ceil

Apply: Move Registers

Apply: Retimed Design

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 → Pipeline

Add Registers

Pipeline Retiming: Lag

Move Registers
Pipelined Retimed

Note

• Algorithm/examples shown
  – for special case of unit-delay nodes

• For general delay,
  – a bit more complicated
  – still polynomial

Initial State

• What about 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

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: $\Sigma w(e)$
- After retime: $\Sigma w(e)+\Sigma (FI(v)-FO(v))lag(v)$
- delta only in lags
- So want to minimize: $\Sigma (FI(v)-FO(v))lag(v)$
  - subject to earlier constraints
    - non-negative register weights, delays
    - positive cycle counts

Minimize Registers → ILP

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

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

Admin

• Milestone Monday
• Will try to send some feedback from milestone 1 today or tomorrow
• Reading for Wednesday online

Big Ideas

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