CS137: Electronic Design Automation

Day 11: October 21, 2005
Retiming

Today

• Retiming
  – Cycle time (clock period)
  – C-slow
  – Initial states
  – Register minimization
  – Necessary delays (time permitting)

Task

• Move registers to:
  – Preserve semantics
  – Minimize path length between registers
  – (make path length 1 for maximum throughput or reuse)
  – Maximize reuse rate
  – …while minimizing number of registers required

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...

Simple Example

Path Length (L) = 4
Can we do better?
Legal Register Moves

- Retiming Lag/Lead

Critical Path Length

- Critical Path: Length of longest path of zero weight nodes
- Compute in $O(|E|)$ time by levelizing network:
  - Topological sort, push path lengths forward until find register.

Valid Retiming

- Retiming is valid as long as:
  - $\forall e$ in graph
    - $\text{weight}(e') = \text{weight}(e) + \text{lag(head(e))} - \text{lag(tail(e))} \geq 0$
  - 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

Canonical Graph Representation

- Separate arc for each path
- Weight edges by number of registers
- (weight nodes by delay through node)
Retiming Transformation

- **N.B.:** unchanged by retiming
  - number of registers around a cycle
  - delay along a cycle

- Cycle of length \( P \) must have
  - at least \( P/c \) registers on it
  - to be retimable to cycle \( c \)

Optimal Retiming

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

- \( G-\alpha \equiv \) 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 \( \text{regs}=P/c \) constraints

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 \( k \leftarrow 0 \) 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 \( e_{ij} \in E \)
  - \( u_i > u_j + w(e_{ij}) \) /still update → negative cycle
    - cycles detected
Apply to Example

Try c=1

Apply: Find Lags
Negative weight cycles?
Shortest paths?

Apply: Lags

Apply: Move Registers
weight(e') = weight(e) + lag(head(e))-lag(tail(e))

Apply: Retimed
Apply: Retimed Design

Revise Example (fanout delay)

Revised: Graph

Revised: Graph

Revised: C=1?

Revised: C=2?
Take ceiling to convert to integer lags:

Revised: Apply Lag

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

Real Cycle
Real Cycle

Cycle C=1?

Cycle C=2?

Cycle: C-slow

2-slow Cycle ⇒ C=1

2-Slow Lags
C-Slow applicable?

• Available parallelism
  – solve C identical, independent problems
    • e.g. process packets (blocks) separately
    • e.g. independent regions in images
  • Commutative operators
    – e.g. max example

Max Example

• 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?

In general, constraints → 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: $\Sigma w(e)$
- After retiming: $\Sigma w(e) + \Sigma (FI(v) - FO(v)) \text{lag}(v)$
- Delta only in lags
- So want to minimize: $\Sigma (FI(v) - FO(v)) \text{lag}(v)$
  - subject to earlier constraints
    - non-negative register weights, delays
    - positive cycle counts

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

Accommodating HSRA Interconnect Delays

- Add buffers to LUT→LUT path to match interconnect register requirements
- Retime to $C=1$ as before
- Buffer chains force enough registers to cover interconnect delays
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
  – Compute multithreaded computations
  – Minimize registers
• Watch out for initial values
• Can accommodate mandatory delays

Admin

• Homework Due Today
• No class on Monday and Wednesday
• Class next Friday

Big Ideas

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