ECE 667

Synthesis and Verification of Digital Systems

Retiming
Retiming

Outline:

• Problem
  – sequential synthesis
• Formulation
• Retiming algorithm
Optimizing Sequential Circuits by Retiming Gate-level Netlist

Netlist of gates and registers:

Various Goals:
- Reduce clock cycle time
- Reduce area
  - Reduce number of latches (registers)
Retiming

Problem

- Pure combinational optimization can be myopic since relations across register boundaries are disregarded

Solutions

- **Retiming**: Move register(s) so that
  - clock cycle decreases, or number of registers decreases and
  - input-output behavior is preserved

- **Peripheral retiming**: Combine retiming with combinational optimization techniques
  - move latches out of the way temporarily
  - optimize larger blocks of combinational logic
Circuit Representation

[Leiserson, Rose and Saxe (1983)]

Circuit representation: $G(V,E,d,w)$
- $V \leftrightarrow$ set of gates
- $E \leftrightarrow$ set of wires
- $d(v) =$ delay of gate/vertex $v$, $(d(v) \geq 0)$
- $w(e) =$ number of registers on edge $e$, $(w(e) \geq 0)$
Example: Correlator

\[ \delta(x, y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases} \]

Every cycle in the graph has at least one register, i.e., there are no combinational loops.
Preliminaries

For a path \( p \): 
\[
\begin{align*}
d(p) &= \sum_{i=0}^{k} d(v_i) \\
w(p) &= \sum_{i=0}^{k-1} w(e_i)
\end{align*}
\]
Clock cycle \( c \):
\[
c = \max_{p: w(p) = 0} \{ d(p) \}
\]
For the correlator circuit: \( c = 13 \)

Can we reduce it to 7? How?

For a path \( p \):
\[
v_0 \rightarrow v_1 \rightarrow \cdots v_{k-1} \rightarrow v_k
\]
includes endpoints}

Path with \( w(p) = 0 \)

Can we reduce it to 7? How?
Basic Operation

• Movement of registers
  – from input to output of a gate or vice versa

• Does not affect gate functionalities

• A mathematical definition: retardation
  – \( r: V \rightarrow Z \), an integer vertex labeling
  – \( w_r(e) = w(e) + r(v) - r(u) \) for edge \( e = (u,v) \)
In the example, $r(u) = -1$, $r(v) = -1$ results in

- For a path $p$: $s \rightarrow t$, $w_r(p) = w(p) + r(t) - r(s)$
- Retardation
  - $r: V \rightarrow Z$, an integer vertex labeling
  - $w_r(e) = w(e) + r(v) - r(u)$ for edge $e = (u, v)$
  - A retiming $r$ is legal if $w_r(e) \geq 0$, $\forall e \in E$ (prove it !)
Retiming for Minimum Clock Cycle

Problem Statement: (minimum cycle time)

Given $G(V, E, d, w)$, find a legal retiming $r$ so that

$$c = \max_{p: w_r(p) = 0} \{d(p)\}$$

is minimized.

Retiming: 2 important matrices

- Register weight matrix

$$W(u, v) = \min_{p} \{w(p) : u \xrightarrow{p} v\}$$

- Delay matrix

$$D(u, v) = \max_{p} \{d(p) : u \xrightarrow{p} v, w(p) = w(u, v)\}$$
Retiming for Minimum Clock Cycle

$W = \text{register path weight matrix}$
( minimum # latches on all paths between $u$ and $v$ )

$D = \text{path delay matrix}$
( maximum delay on all paths between $u$ and $v$ with min number of latches)

$c \leq \alpha \iff \forall p, \text{ if } d(p) > \alpha \text{ then } w(p) \geq 1$
Conditions for Legal Retiming

Assume that we are asked to check if a retiming exists for a clock cycle $\alpha$

**Legal retiming**: $w_i(e) \geq 0$ for all $e$. Hence

$$w_r(e) = w(e) + r(v) - r(u) \geq 0 \quad \text{or}$$

$$r(u) - r(v) \leq w(e)$$

For all paths $p: u \rightarrow v$ such that $d(p) > \alpha$, we require $w_r(p) \geq 1$

- Thus

$$1 \leq w_r(p) = \sum_{i=0}^{k-1} w_r(e_i)$$

$$= \sum_{i=0}^{k-1} [w(e_i) + r(v_{i+1}) - r(v_i)]$$

$$= w(p) + r(v_k) - r(v_0)$$

$$= w(p) + r(v) - r(u)$$

*Or take the least $w(p)$ (tightest constraint)*

$$r(u) - r(v) \leq W(u,v) - 1$$

**Note**: this is independent of the path from $u$ to $v$, so we just need to apply it to $u, v$ such that $D(u,v) > \alpha$
Solving the Constraints

- All constraints in difference-of-2-variable form
- Related to longest/shortest path problem

**Correlator:** \( \alpha = 7 \)

**Legal:**
\[
\begin{align*}
    r(v_0) - r(v_1) &\leq 2 \\
    r(v_1) - r(v_2) &\leq 0 \\
    r(v_1) - r(v_3) &\leq 0 \\
    r(v_2) - r(v_3) &\leq 0 \\
    r(v_3) - r(v_0) &\leq 0
\end{align*}
\]

**Timing \( D>7 \):**
\[
\begin{align*}
    r(v_0) - r(v_3) &\leq 1 \\
    r(v_1) - r(v_0) &\leq -1 \\
    r(v_1) - r(v_3) &\leq -1 \\
    r(v_2) - r(v_0) &\leq -1 \\
    r(v_2) - r(v_3) &\leq 1 \\
    r(v_3) - r(v_0) &\leq 1 \\
    r(v_3) - r(v_2) &\leq 1
\end{align*}
\]
Solving the Constraints

- Do shortest path on constraint graph: \( O(|V|^3) \).
- A solution exists if and only if there exists no negative weighted cycle.

Legal:
\[
\begin{align*}
\text{Timing } D>7: \\
r(u) - r(v) &\leq W(u,v) - 1 \\
\end{align*}
\]

A solution:
\[
\begin{align*}
r(v_0) &= r(v_3) = 0, \\
r(v_1) &= r(v_2) = -1
\end{align*}
\]
Retiming

To find the minimum cycle time, do a binary search among the entries of the D matrix \( O(V^3 \log V) \)

Retimed correlator:

Clock cycle = 3 + 3 + 7 = 13

Clock cycle = 7
Retiming: two more Algorithms

1. Relaxation based:
   - Repeatedly find critical path;
   - retime vertex at end of path by +1 \( (O(|V|E\log|V|)) \)

2. Also, Mixed Integer Linear Program formulation
Relaxation Algorithm - Rationale

- Look for paths with excessive delay.
- Make them shorter by pulling closer the terminal register.
  - Some other paths may become too long.
  - Those paths whose tail has been moved.
- Use an iterative approach.
Relaxation Algorithm

- Define vertex *data ready* time:
  - Total delay from register boundary.

- Iterative approach:
  - Find vertices with *data ready* time > φ.
  - Retime these vertices by 1.

- Properties:
  - Finds legal retiming in at most |V| iterations, if one exists.
Relaxation Algorithm – step 1

Retime for $\phi = 13$

- Data-ready times:
  
  \[ t_a = 3; \ t_b = 3; \ t_c = 3; \ t_d = 3; \ t_e = 10; \]
  \[ t_f = 17; \ t_g = 24; \ t_h = 24. \]

- Retime: \{t_f, t_g, t_h\} by 1.
Relaxation Algorithm – step 2

Retime for $\phi = 13$

- Data-ready times:
  - $t_a = 17; t_b = 3; t_c = 3; t_d = 3; t_e = 10; t_f = 7; t_g = 14; t_h = 14.$

- Retime:
  - $\{t_a, t_g, t_h\}$ by 1.
Relaxation Algorithm – step 3

Retimed for $\phi = 13$

- Data-ready times:
  - $t_a = 10; t_b = 13; t_c = 3; t_d = 3; t_e = 10; t_f = 7; t_g = 7; t_h = 7,$
  - TIMING FEASIBLE NETWORK!
Relaxation Algorithm – summary
(Retiming for $\phi = 13$)
Retiming for Minimum Area
(Minimum # Latches)

**Goal:** minimize the number of registers used

\[
\begin{align*}
\min N_r &= \sum_{e \in E} w_r(e) \\
&= \sum_{e:u \to v} (w(e) + r(v) - r(u)) \\
&= \sum_{e \in E} w(e) + \sum_{e:u \to v} (r(v) - r(u)) \\
&= N + \sum_{u \to v} (r(v) - r(u)) \\
&= N + \sum_{v \in V} [r(v)(\# \text{fanin}(v) - \# \text{fanout}(v))]
\end{align*}
\]

where \( a_v \) is a constant for each node \( v \).
Minimum Registers - Formulation

Minimize:

\[ \sum_{v \in V} a_v r(v) \]

Subject to: \( w_r(e) = w(e) + r(v) - r(u) \geq 0 \)

- Reducible to a flow problem