Retiming

Adapted from: *Synthesis and Optimization of Digital Circuits*, G. De Micheli © Stanford

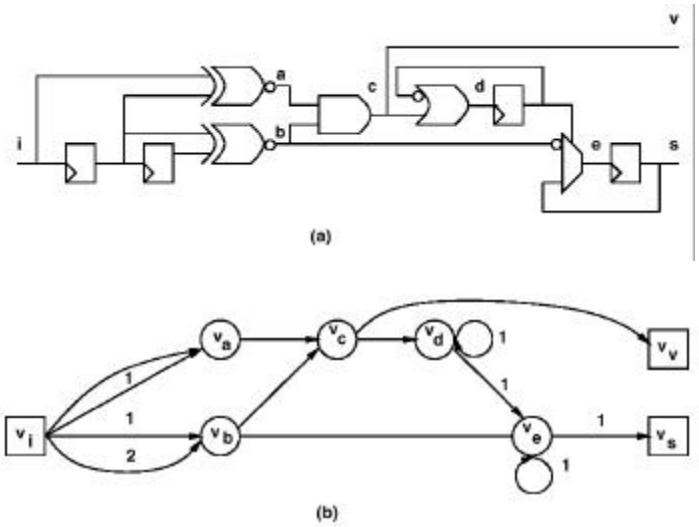
Outline

- Structural optimization methods.
- Retiming.
  - Modeling.
  - Retiming for minimum delay.
  - Retiming for minimum area.

Synchronous Logic Circuits

- Synchronous logic network:
  - Variables.
  - Boolean equations.
  - Synchronous delay annotation.
- Synchronous network graph:
  - Vertices ↔ equations ↔ I/O , gates.
  - Edges ↔ dependencies ↔ nets.
  - Weights ↔ synch. delays ↔ registers.

Synchronous Logic Network: Example



Example

$$
\begin{aligned}
a^{(n)} &= i^{(n)} \overline{\oplus}\, i^{(n-1)} \\
b^{(n)} &= i^{(n-1)} \overline{\oplus}\, i^{(n-2)} \\
c^{(n)} &= a^{(n)} b^{(n)} \\
d^{(n)} &= c^{(n)} + d'^{(n-1)} \\
e^{(n)} &= d^{(n)} e^{(n-1)} + d'^{(n)} b'^{(n)} \\
v^{(n)} &= c^{(n)} \\
s^{(n)} &= e^{(n-1)}
\end{aligned}
\qquad
\begin{aligned}
a &= i \overline{\oplus}\, i@1 \\
b &= i@1 \overline{\oplus}\, i@2 \\
c &= a\ b \\
d &= c + d@1' \\
e &= d\ e@1 + d'\ b' \\
v &= c \\
s &= e@1
\end{aligned}
$$

Synchronous Logic Circuit Modeling

- State-based model:
  - Transition diagrams or tables.
  - Explicit notion of state.
  - Implicit notion of area and delay.
- Structural model:
  - Synchronous logic network.
  - Implicit notion of state.
  - Explicit notion of area and delay.

Approaches to Synchronous Logic Optimization

- Optimize combinational logic only.
- Optimize register position only:
  - Retiming.
- Optimize overall circuit:
  - Peripheral retiming.
  - Synchronous transformations:
    - Algebraic.
    - Boolean.

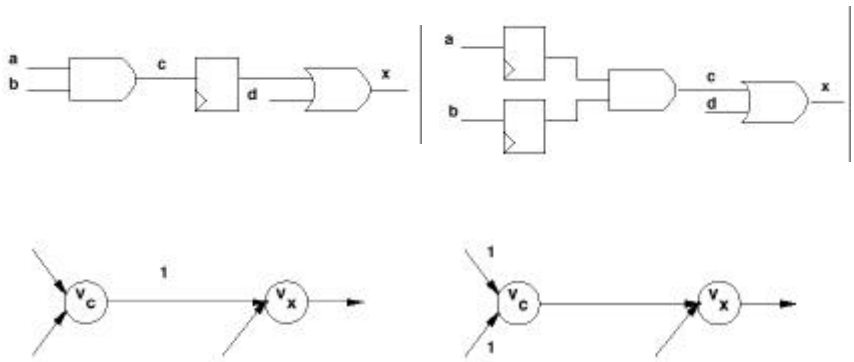
Separate Registers from Combinational Logic

- Optimize combinational logic by transformations:
  - Modify equations.
  - Modify graph structure.
- Connect registers back into the network:
  - Good heuristic.
  - Limited by the partitioning strategy.

Retiming

- Move register position.
- Do not modify combinational logic.
- Preserve network structure:
  - Modify weights.
  - Do not modify graph structure.

Example



Retiming

- Global optimization technique [Leiserson].
- Changes register positions:
  - affects area:
    - changes register count.
  - affects cycle-time:
    - changes path delays between register pairs.
- Solvable in polynomial time.

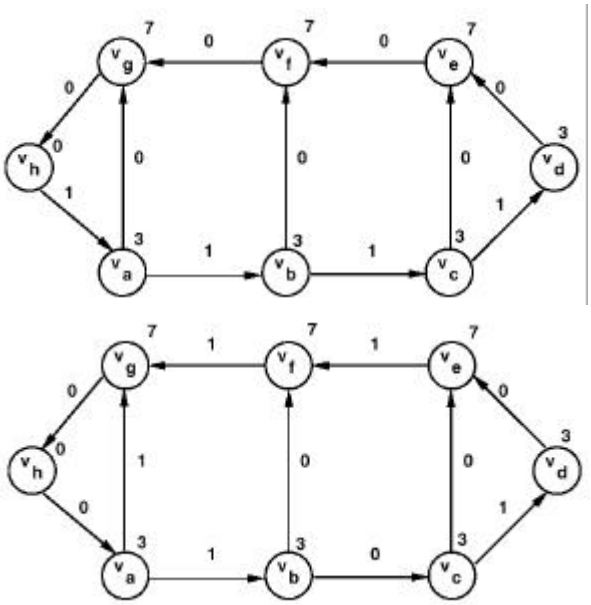
Assumptions

- Vertex delay is constant:
  - No fanout delay dependency.
- Graph topology is invariant:
  - No logic transformations.
- Synchronous implementation:
  - Cycles have positive weights.
  - Edges have non-negative weights.
- Consider topological paths:
  - No false path analysis.

Retiming

- Retiming of a vertex:
  - Integer.
  - Registers moved from output to input.
- Retiming of a network:
  - Vector of vertex retiming.
- A family of equivalent networks are specified by:
  - The original network.
  - A retiming vector.

Example



Definition and Properties

- Definitions:
  - $w(v_i, v_j)$: weight of edge (v_i, v_j).
  - $(v_i, \ldots, v_j)$: path from v_i to v_j.
  - $d(v_i, \ldots, v_j)$: path delay from v_i to v_j.
- Properties:
  - Retiming of an edge (v_i, v_j):

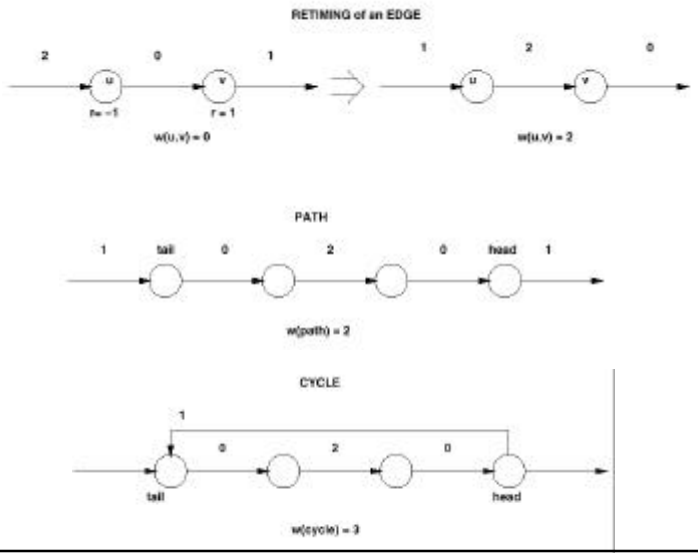
$$\tilde{w}_{ij} = w_{ij} + r_j - r_i.$$

  - Retiming of a path $(v_i, \ldots, v_i)$:

$$\tilde{w}(v_i, \ldots, v_j) = w(v_i, \ldots, v_j) + r_j - r_i.$$

  - Cycle weights are invariant.

Example



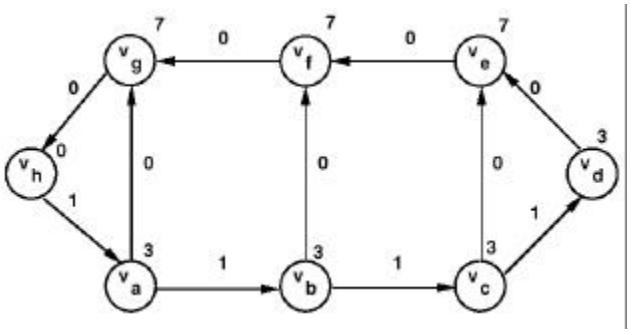
Legal Retiming

- Clock period ϕ.
- Retiming vector such that:
  - No edge weight is negative:
    $$\tilde{w}_{ij} = w_{ij} + r_j - r_i \geq 0 \quad \forall i, j.$$
  - Each path (v_i, …, v_j) with d(v_i, …, v_j) > ϕ has at least one register:
    $$\tilde{w}(v_i, \ldots, v_j) = w(v_i, \ldots, v_j) + r_j - r_i \geq 1 \ \forall i, j.$$
- Fact:
  - Original graph has no cycles with weight ≤ 0 ⇒ new graph has no cycles with weight ≤ 0.

Refined Analysis

- Least register path:
  - W(v_i, v_j) = min w(v_i, …, v_j).
  - Over all paths between v_i and v_j.
- Critical delay:
  - D (v_i, v_j) = max d(v_i, …, v_j)
  - Over all the paths from v_i and v_j with weight W(v_i, v_j).
- There exists a vertex pair v_i, v_j whose D(v_i, v_j) bounds the cycle-time.

Example



- Vertices: v_a,v_e.
- Paths: (v_a, v_b, v_c, v_e) and (v_a, v_b, v_c, v_d, v_e).
- W(v_a, v_e) = 2 and D(v_a, v_e) = 16.

Minimum Cycle-Time Retiming Problem

- Find minimum value of the clock period ϕ such that there exist a retiming vector where:

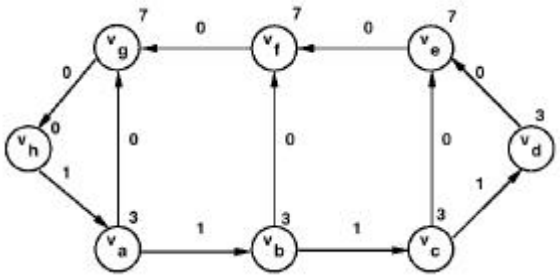
$$r_i - r_j \leq w_{ij} \quad \forall (v_i, v_j) \in E$$
$$r_i - r_j \leq W(v_i, v_j) - 1 \quad \forall v_i, v_j | D(v_i, v_j) > \phi.$$

- Solution:
  - Given a value of ϕ:
    - solve linear constraints.
    - methods:
      - Bellman-Ford or derivate.
      - MILP.
      - Relaxation.

Minimum Cycle-Time Retiming Algorithm

- Compute all-pair $W(v_i, v_j)$ and $D(v_i, v_j)$.
  - Warshall-Floyd algorithm ($O(V^3)$).
- Sort the elements of D in decreasing order.
- Binary search for a ϕ in $D(v_i, v_j)$ such that:
  - There exist a legal retiming.
  - Bellman-Ford algorithm ($O(V^3)$).
- Remarks:
  - Result is a global optimum.
  - Overall complexity is $O(V^3 \log V)$.

Example

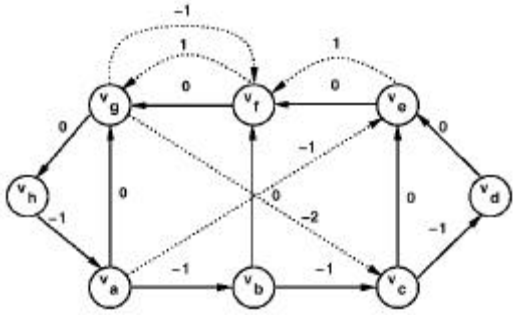


- Constraints (first type):
  - $r_a - r_b \leq 1$ or equivalently $r_b \geq r_a - 1$
  - $r_b - r_c \leq 1$ or equivalently $r_c \geq r_b - 1$
  - …

Example

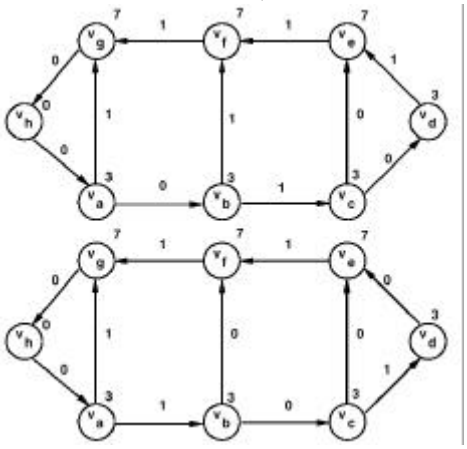
- Sort elements of D:
  - (33, 30, 27, 26, 24, 23, 21, 20, 19, 17, 16, 14, 13, 12, 10, 9, 7, 6, 3).
- Select: $\phi = 19$:
  - PASS.
- Select: $\phi = 13$:
  - PASS.
- Select: $\phi < 13$:
  - FAIL.

Example: $\phi = 13$



- Constraints (second type):
  - $r_a - r_e \leq 2 - 1$ or equivalently $r_e \geq r_a - 1$
  - $r_e - r_f \leq 0 - 1$ or equivalently $r_f \geq r_e + 1$
  - $r_f - r_g \leq 0 - 1$ or equivalently $r_g \geq r_f + 1$
  - $r_g - r_f \leq 2 - 1$ or equivalently $r_f \geq r_g - 1$
  - $r_g - r_c \leq 3 - 1$ or equivalently $r_c \geq r_g - 2$

Example: $\phi = 13$



- Solutions:
  - $-[12232100]^T$ (LP from vh)
  - $-[11222100]^T$ (Equivalent solution)

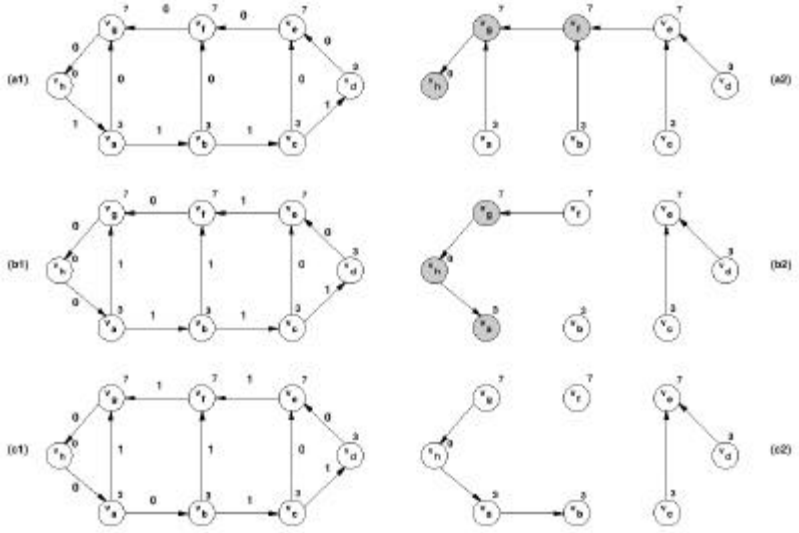
Relaxation-Based Algorithm: Rationale

- 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-Based 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.

Example: ϕ = 13



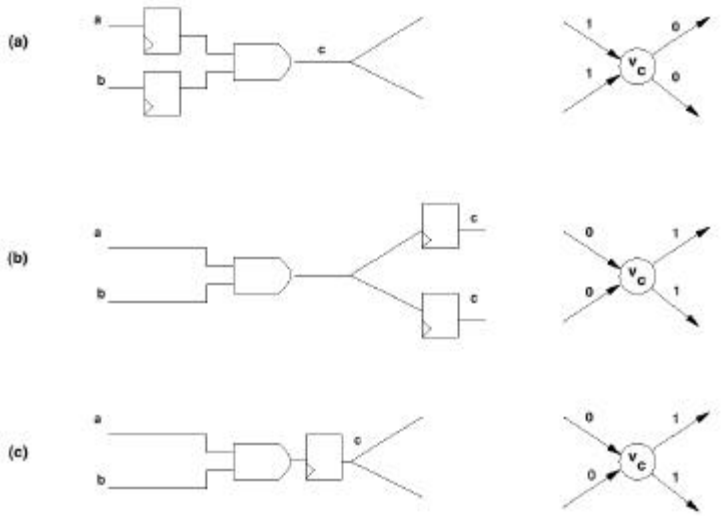
Example

- 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.
- 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.
- 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 !

Minimum Area Retiming Problem

- Find a retiming vector that minimizes the number of registers.
- Simple area modeling:
  - Every pos.-weighted edge $\rightarrow$ register.
  - Total register area cost equals total of weights.
- Register sharing model:
  - Every set of positively-weighted edges with common tail $\rightarrow$ shift-register.
  - Register area cost equals maximum of weights on outgoing edges.

Example



Minimum Area Retiming: Simple Model

- Register variation at vertex v:
  $r_v \cdot (indegree(v) - outdegree(v)).$
- Total area variation:
  $\sum r_v \cdot (indegree(v) - outdegree(v)).$
- Area minimization problem:

  min
  * $\sum_{v \in V} r_v \cdot (indegree(v) - outdegree(v)).$

  s.t.
  * $r_i - r_j \leq w(v_i, v_j)$ for every (v_i, v_j).

Minimum Area Retiming Under Cycle Time Constraint ϕ

min

$$- \sum_{v \in V} r_v \cdot (indegree(v) - outdegree(v)).$$

s.t.

- $r_i - r_j \leq w(v_i, v_j)$ for every (v_i, v_j).
- $r_i - r_j \leq W(v_i, v_j) - 1 \quad \forall v_i, v_j | D(v_i, v_j) > \phi.$

Minimum Area Retiming Algorithm

- Linear program.
- Transform into matching problem:
  - Edmonds-Karp algorithm.
  - Polynomial time.
- Remark:
  - Register sharing model can be transformed into the simple model.
  - Same solution algorithms.

Summary of Retiming

- Sequential optimization technique for:
  - Cycle time or register area.
- Applicable to:
  - Synchronous logic models.
  - Architectural data-path models:
    - Resources with delays.
- Exact algorithm in polynomial time.
- Problems:
  - Delay modeling.
  - Network granularity.