

RETIMING

© *Giovanni De Micheli*

Stanford University

Outline

© GDM

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

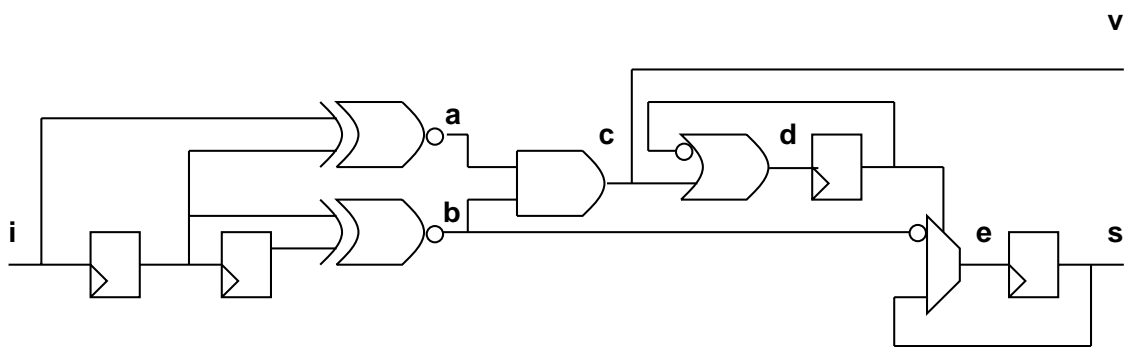
Synchronous Logic Network

© GDM

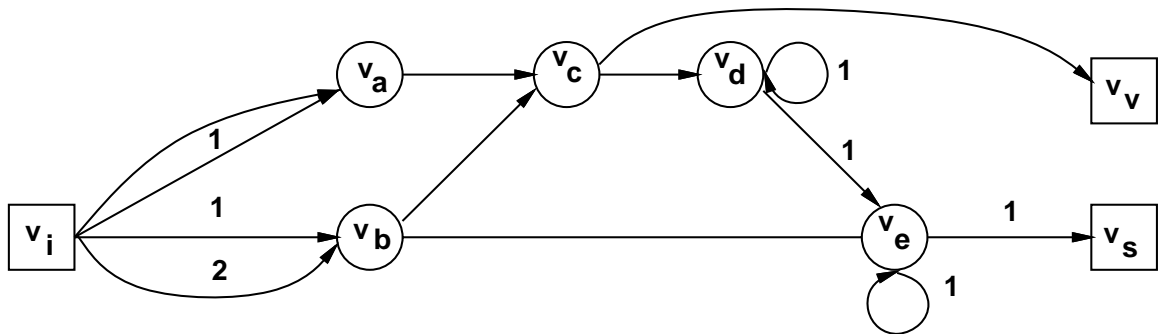
- Synchronous logic network:
 - Variables.
 - Boolean equations.
 - Synchronous delay annotation.
- Synchronous network graph:
 - Vertices \leftrightarrow equations \leftrightarrow I/O , gates.
 - Edges \leftrightarrow dependencies \leftrightarrow nets.
 - Weights \leftrightarrow synch. delays \leftrightarrow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

Synchronous Logic Network Example

© GDM



(a)



(b)

Example

© GDM

$$a^{(n)} = i^{(n)} \overline{\oplus} i^{(n-1)}$$

$$b^{(n)} = i^{(n-1)} \overline{\oplus} i^{(n-2)}$$

$$c^{(n)} = a^{(n)} b^{(n)}$$

$$d^{(n)} = c^{(n)} + d'^{(n-1)}$$

$$e^{(n)} = d^{(n)} e^{(n-1)} + d'^{(n)} b'^{(n)}$$

$$v^{(n)} = c^{(n)}$$

$$s^{(n)} = e^{(n-1)}$$

$$a = i \overline{\oplus} i \textcircled{1}$$

$$b = i \textcircled{1} \overline{\oplus} i \textcircled{2}$$

$$c = a b$$

$$d = c + d \textcircled{1}'$$

$$e = d e \textcircled{1} + d' b'$$

$$v = c$$

$$s = e \textcircled{1}$$

Synchronous Logic Circuit Modeling

© GDM

- 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

© GDM

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

Separate registers from combinational logic

© GDM

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

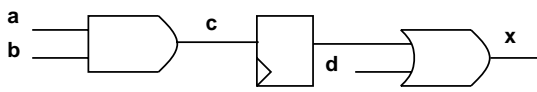
Retiming

© GDM

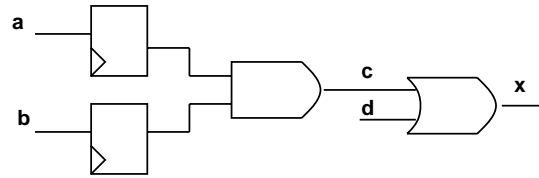
- Move register position.
- Do not modify combinational logic.
- Preserve network structure:
 - Modify weights.
 - Do not modify graph structure.

Example

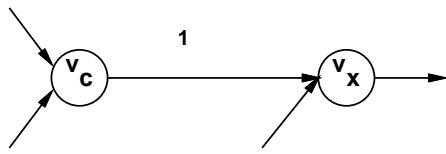
© GDM



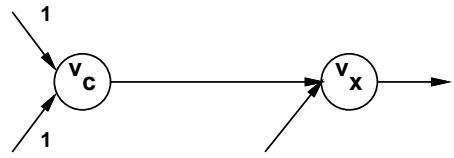
(a)



(c)



(b)



(d)

Retiming

© GDM

- 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

© GDM

- 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

© GDM

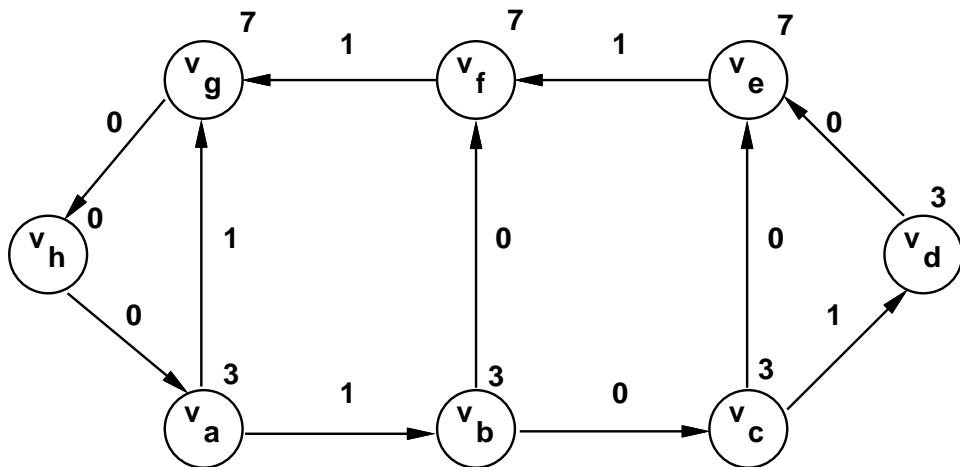
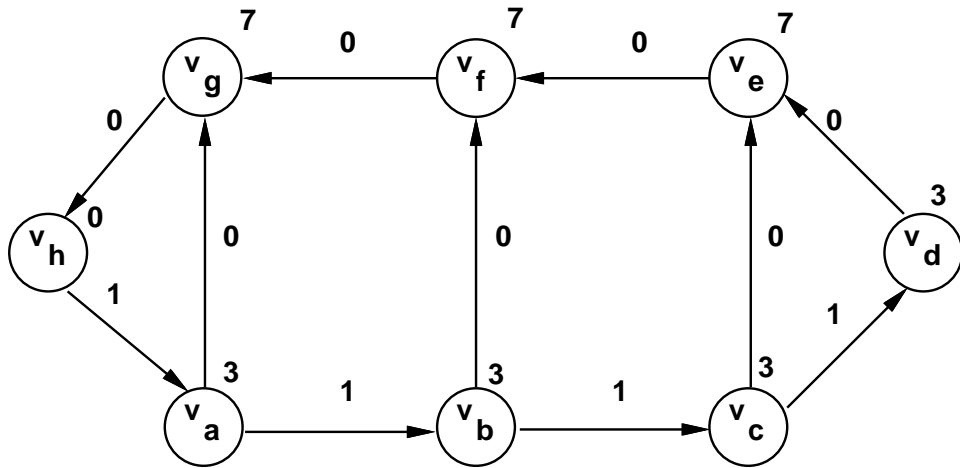
- Retiming of a vertex:
 - Integer.
 - Registers moved from output to input.

- Retiming of a network:
 - Vector of vertex retiming.

- A family of equivalent networks are specified by:
 - The original network.
 - A retiming vector.

Example

© GDM



Definitions and properties

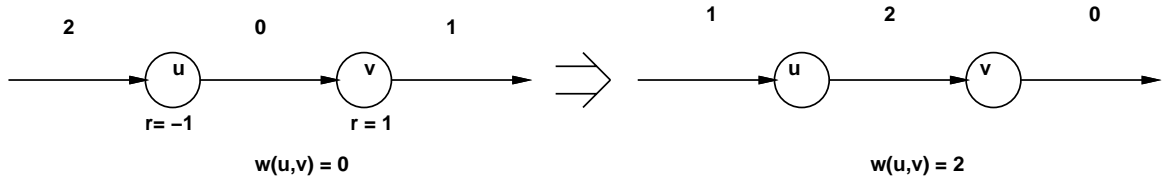
© GDM

- Definitions:
 - $w(v_i, v_j)$ – weight of edge (v_i, v_j) .
 - (v_i, \dots, v_j) – path from v_i to v_j .
 - $d(v_i, \dots, v_j)$ – 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, \dots, v_j) :
 - * $\tilde{w}(v_i, \dots, v_j) = w(v_i, \dots, v_j) + r_j - r_i$.
 - Cycle weights are invariant.

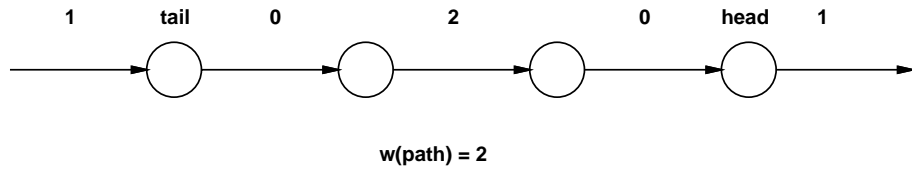
Example

© GDM

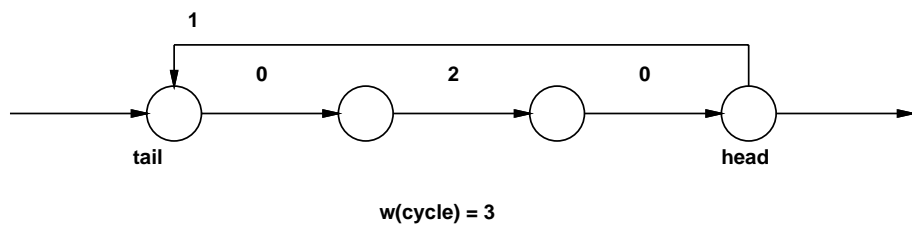
RETIMING of an EDGE



PATH



CYCLE



Legal retiming

© GDM

- 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, \dots, v_j) with $d(v_i, \dots, v_j) > \phi$ has at least one register:
$$\tilde{w}(v_i, \dots, v_j) = w(v_i, \dots, v_j) + r_j - r_i \geq 1$$

 $\forall i, j.$
- Fact:
 - Original graph has no cycles with weight ≤ 0
 \Rightarrow new graph has no cycles with weight ≤ 0

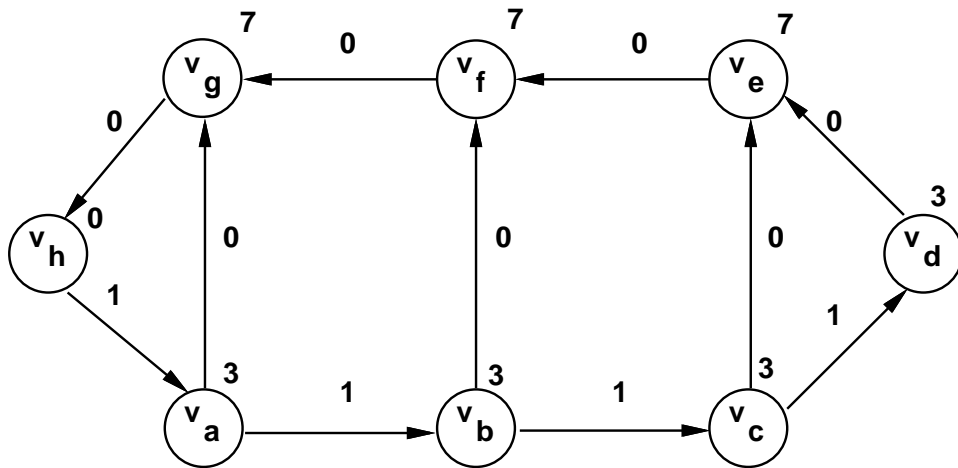
Refined analysis

© GDM

- Least register path:
 - $W(v_i, v_j) = \min w(v_i, \dots, v_j)$.
 - Over all paths between v_i and v_j .
- Critical delay:
 - $D(v_i, v_j) = \max d(v_i, \dots, v_j)$
 - Over all the paths from v_i to 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

© GDM



- 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

© GDM

- 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.

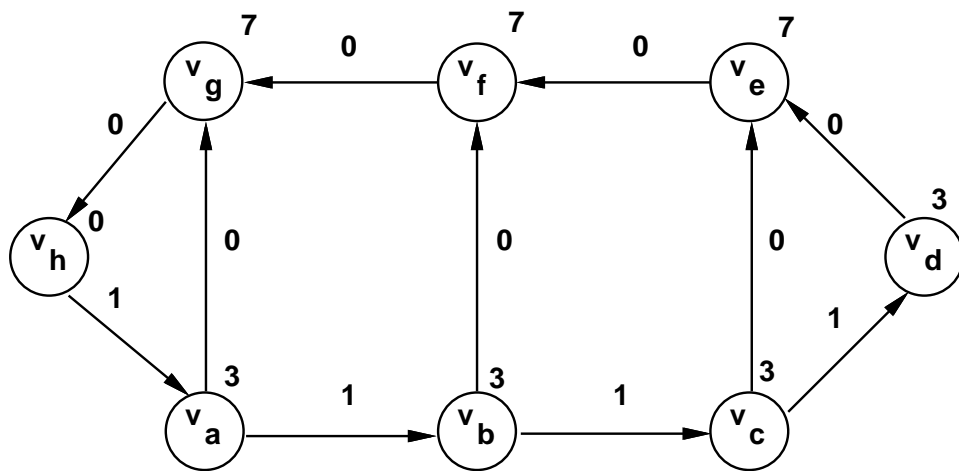
Minimum cycle-time retiming algorithm

© GDM

- Compute all-pair $W(v_i, v_j)$ and $D(v_i, v_j)$.
 - Warshall-Floyd algorithm ($O(V^3)$).
- Sort the elements of \mathbf{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

© GDM



- Constraints (first type):

- $r_a - r_b \leq 1$ or equivalently $r_b \geq r_a - 1$

- $r_c - r_b \leq 1$ or equivalently $r_c \geq r_b - 1$

- ...

Example

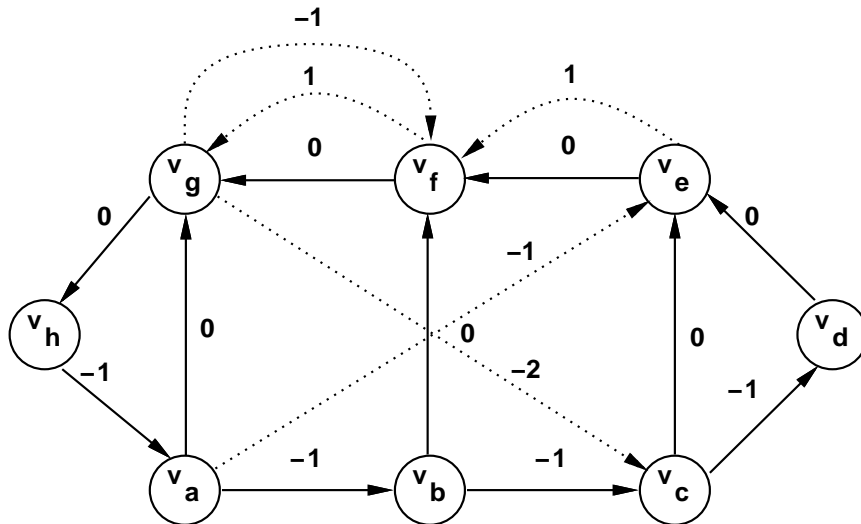
© GDM

- 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

$$\phi = 13$$

© GDM



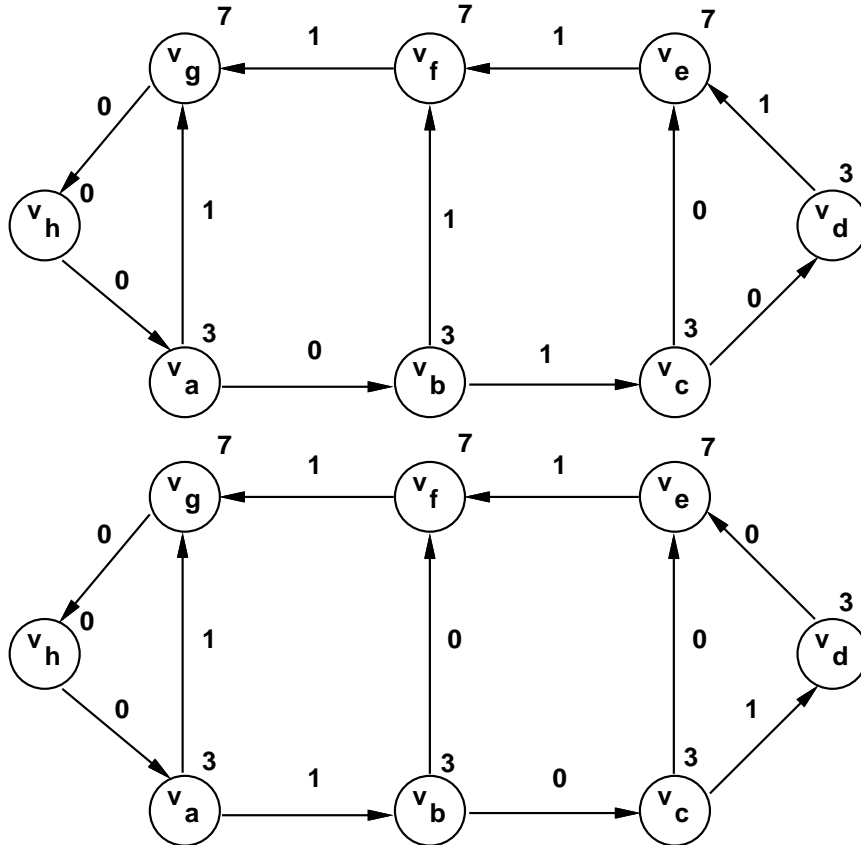
- Constraints (second type):

- $r_a - r_e \leq 2 - 1$ 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

$$\phi = 13$$

© GDM



- Solutions:

- $-[12232100]^T$ (LP from v_h).

- $-[11222100]^T$ (Equivalent solution).

Relaxation-based algorithm Rationale

© GDM

- 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

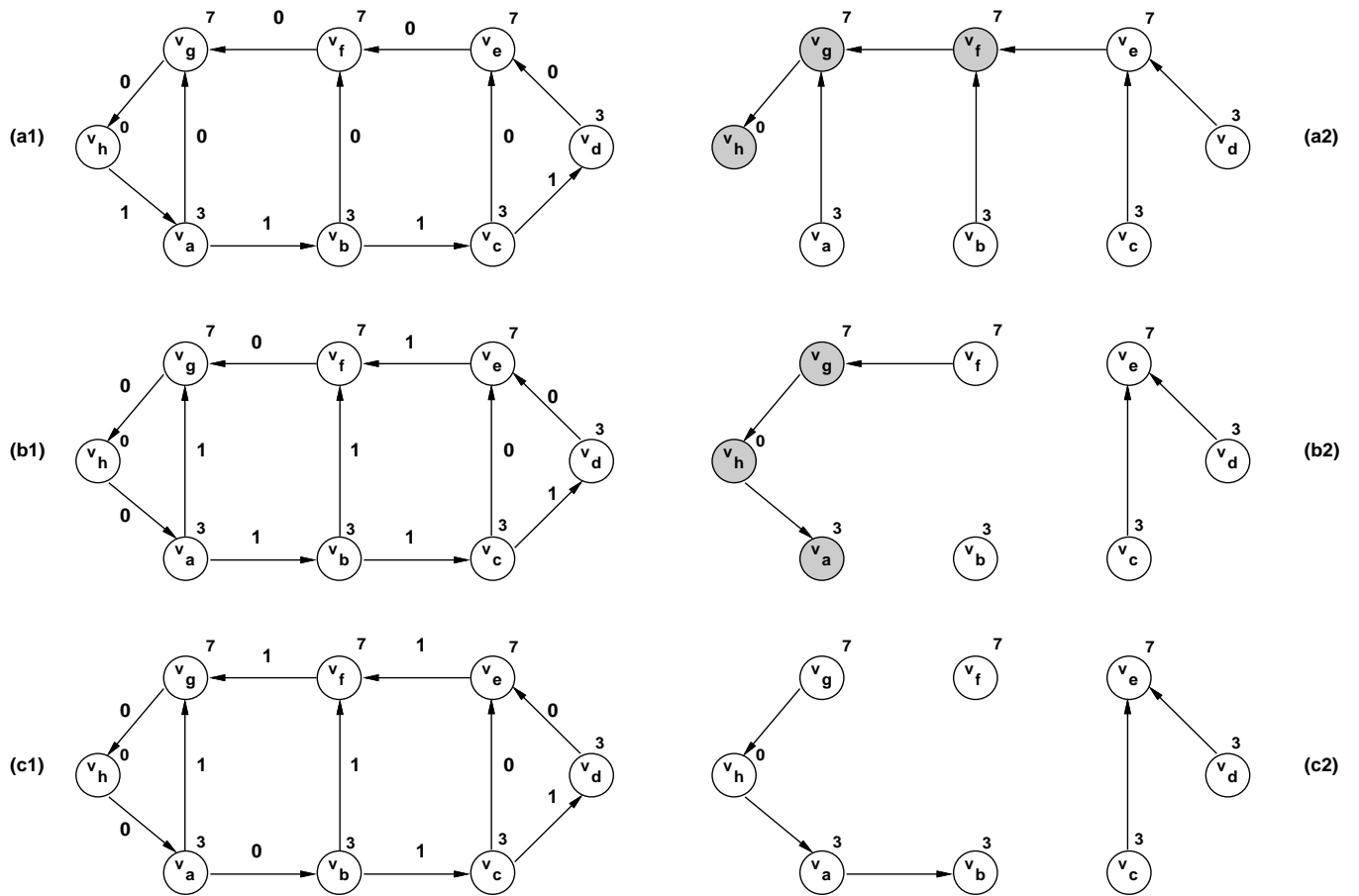
© GDM

- Define vertex *data ready* time:
 - Total delay from register boundary.
- Iterative approach:
 - Find vertices with *data ready* time $> \phi$.
 - Retime these vertices by 1.
- Properties:
 - Finds legal retiming in at most $|V|$ iterations, if one exists.

Example

$$\phi = 13$$

© GDM



Example

© GDM

- 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 !

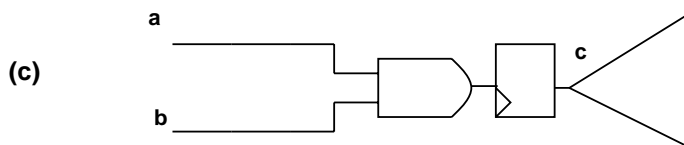
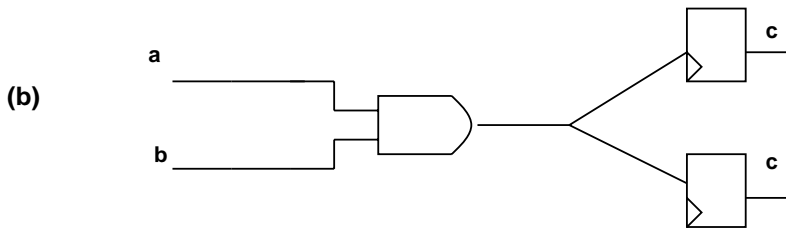
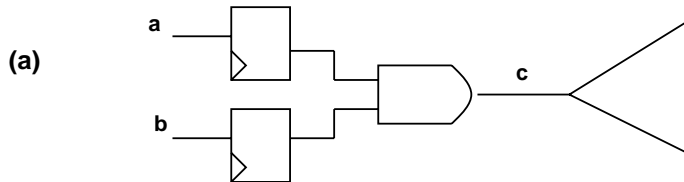
Minimum area retiming problem

© GDM

- 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

© GDM



Minimum area retiming simple model

© GDM

- Register variation at vertex v :
 - $r_v \cdot (\text{indegree}(v) - \text{outdegree}(v))$.
- Total area variation:
 - $\sum r_v \cdot (\text{indegree}(v) - \text{outdegree}(v))$.
- Area minimization problem:
 - min
 - * $\sum_{v \in V} r_v \cdot (\text{indegree}(v) - \text{outdegree}(v))$.
 - 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

© GDM

- min

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

- s.t.

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

- $r_i - r_j \leq W(v_i, v_j) - 1 \quad \forall v_i, v_j \mid D(v_i, v_j) > \phi.$

Minimum area retiming algorithm

© GDM

- 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

© GDM

- 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.