Sequential Logic Synthesis

TU Delft

Delft University of Technology

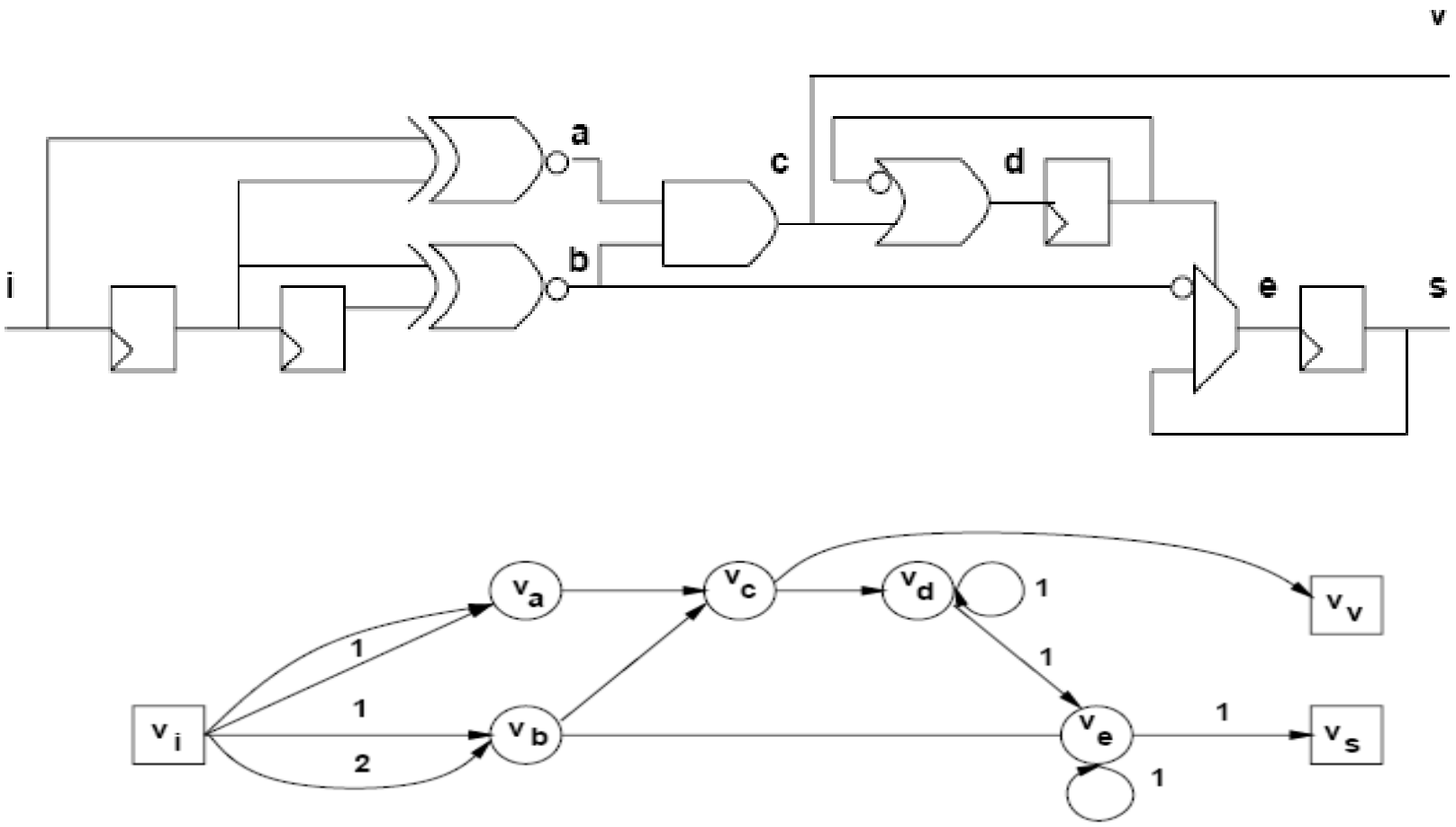
Module 2

- Objective
  - Structural representation of sequential circuits
  - Retiming
  -

Structural model for sequential circuits

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

Example

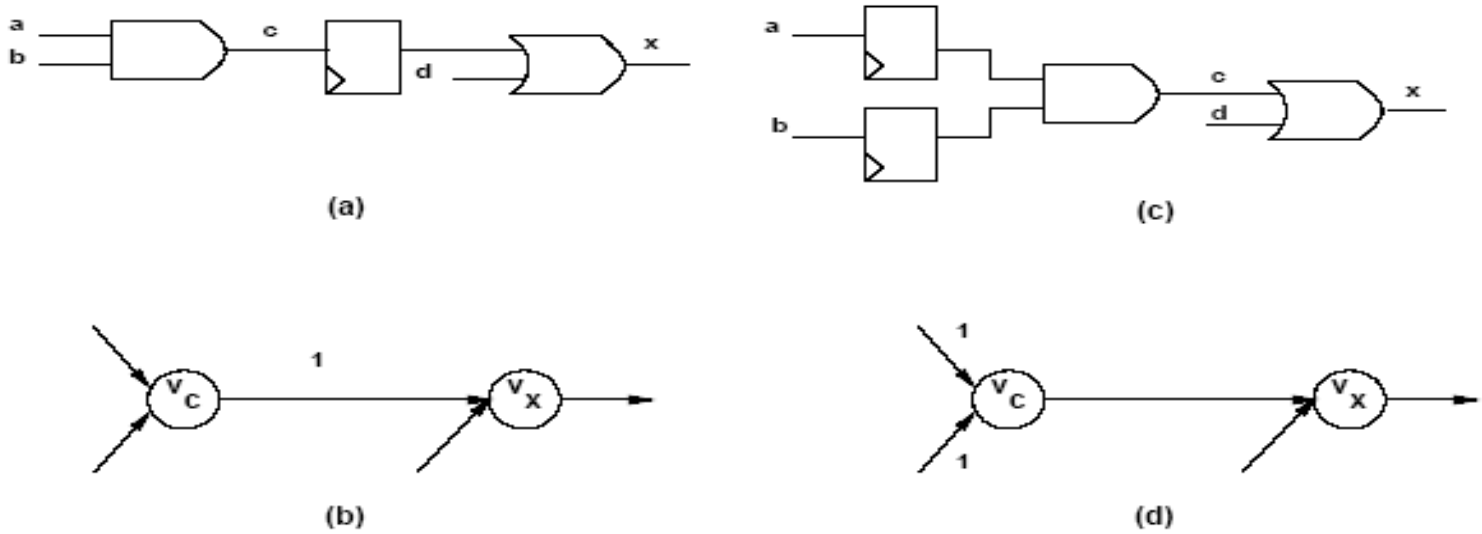


TU Delft

Approaches to sequential synthesis

- Optimize combinational logic only
  - Freeze circuit at register boundary
  - Modify equation and network graph topology
- Retiming
  - Move register positions. Change weights on graph
  - Preserve network topology
- Synchronous transformations
  - Blend combinational transformations and retiming
  - Powerful, but complex to use

Example of local retiming



Retiming

- Global optimization technique
- Change register positions
  - Affects area:
    - Retiming changes register count
  - Affects cycle-time
    - Changes path delays between register pairs
- Retiming algorithms have polynomial-time complexity

Retiming assumptions

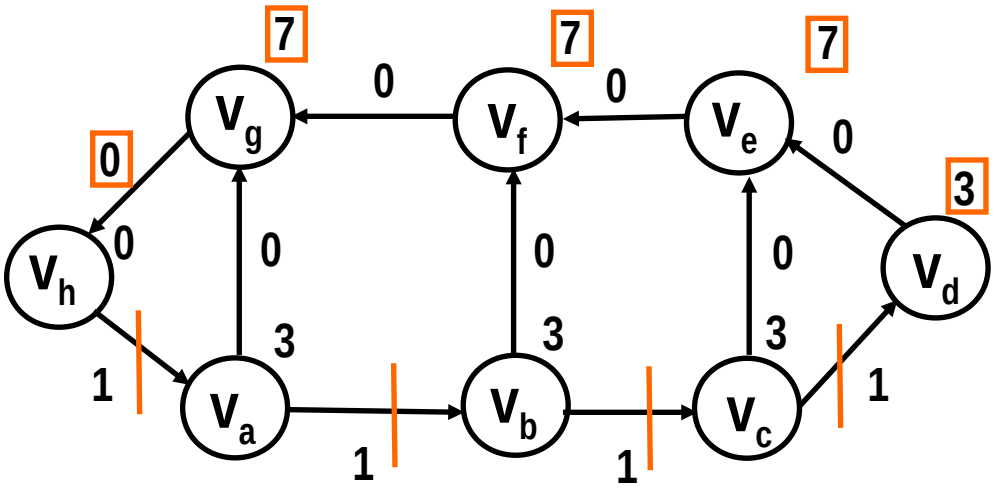
- Delay is constant at each vertex
  - No fanout delay dependency
- Graph topology is invariant
  - No logic transformations
- Synchronous implementation
  - Cycles have positive weights
    - Each feedback loop has to be broken by at least one register
  - Edges have non-negative weights
    - Physical registers cannot anticipate time
- Consider topological paths
  - No false path analysis

Retiming

- Retiming of a vertex v
  - Integer r_v
  - Registers moved from output to input – r_v positive
  - Registers moved from input to output – r_v negative
-

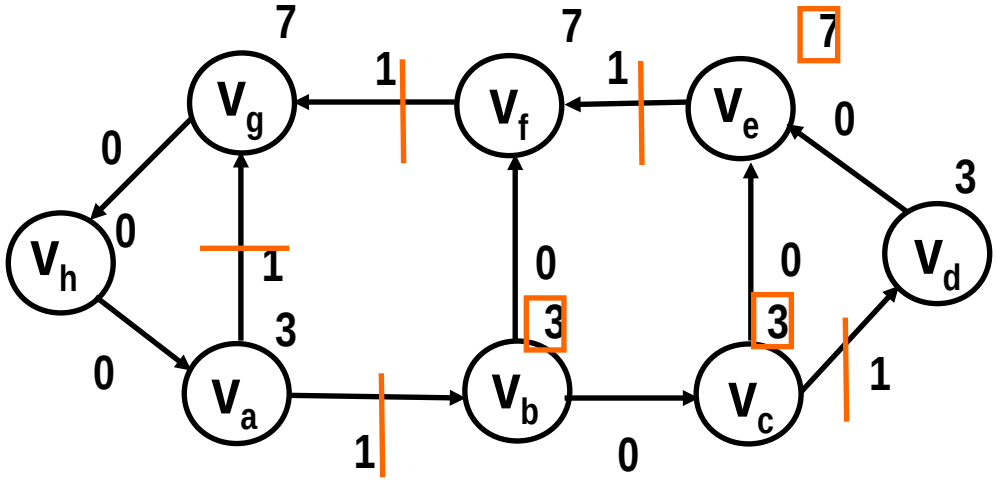
Example 9.3.4

Original graph



Delay: 24

Retimed graph

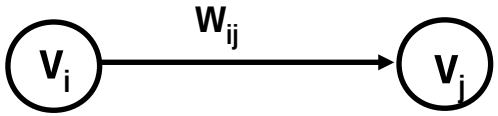


Delay: 13

TU Delft

Definitions and properties

- Definitions:
  - $(v_i, \ldots, v_j)$ path from v_i to v_j
  - $w(v_i, \ldots, v_j)$ weight on path from v_i to v_j
  - $d(v_i, \ldots, v_j)$ combinational delay on path from v_i to v_j
- Properties:
  - Retiming of an edge (v_i, v_j)
    - $\hat{w}_{ij} = w_{ij} + r_j - r_i$
  - Retiming of a path $(v_i, \ldots, v_j)$
    - $\hat{w} (v_i, \ldots, v_j) = w (v_i, \ldots, v_j) + r_j - r_i$
  - Cycle weights are invariant



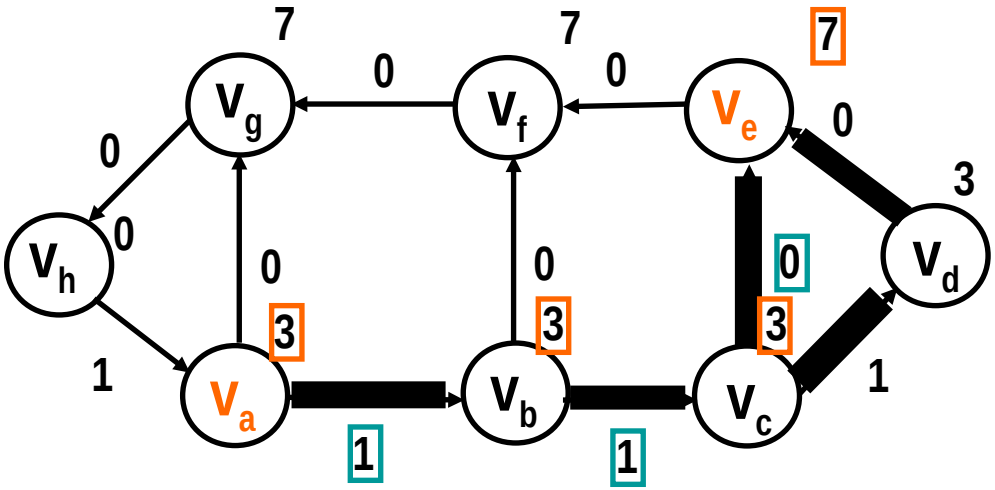
Legal retiming

- Clock period φ
- A retiming vector is legal if:
  - No edge weight is negative
    - $\hat{w}_{ij}(v_i, v_j) = w_{ij}(v_i, v_j) + r_j - r_i \geq 0$ for all i, j
  - Each path $(v_i, \ldots, v_j)$ with $d(v_i, \ldots, v_j) > \varphi$ has at least one register:
    - $\hat{w}(v_i, \ldots, v_j) = w(v_i, \ldots, v_j) + r_j - r_i \geq 1$ for all i, j
  - Equivalently, each combinational path delay is less than φ

Refined analysis

- Least-register path
  - $W(v_i, v_j) = \min w(v_i, \ldots, v_j)$ over all paths between v_i and v_j
- Critical delay:
  - $D(v_i, v_j) = \max d(v_i, \ldots, v_j)$ over all paths between v_i and v_j with weight $W(v_i, v_j)$
- There exist a vertex pair (v_i, v_j) whose delay $D(v_i, v_j)$ bounds the cycle time

Example 9.3.9



- **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)**
- **$W(v_a, v_e) = 2$**
- **$D(v_a, v_e) = 16$**

Maximum delay on path with minimum register count

Other problems related to retiming

- Retiming pipelined circuits
  - Balance pipe stages by using retiming
  - Trade-off latency versus cycle time
- Peripheral retiming
  - Use retiming to move registers to periphery of a circuit
  - Restore registers after optimizing combinational logic
- Wire pipelining
  - Use retiming to pipeline interconnection wires
  - Model sequential and combinational macros
  - Consider wire delay and buffering

Summary of retiming

- Sequential optimization technique for:
  - Cycle time or register area
- Applicable to
  - Synchronous logic networks
  - Architectural models of data paths
    - Vertices represent complex (arithmetic) operators
  - Exact algorithm in polynomial time
- Extension and issues
  - Delay modeling
  - Network granularity