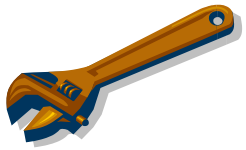


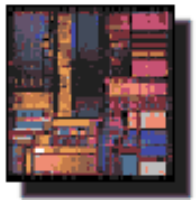




ECE 595Z
Digital VLSI Design Automation

Module 7 (Lectures 24-27): Sequential Logic Optimization

Lecture 27



Anand Raghunathan

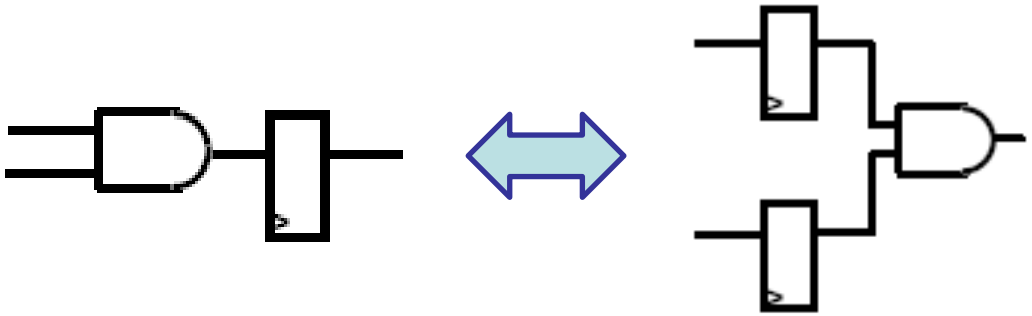
MSEE 318

raghunathan@purdue.edu

Retiming

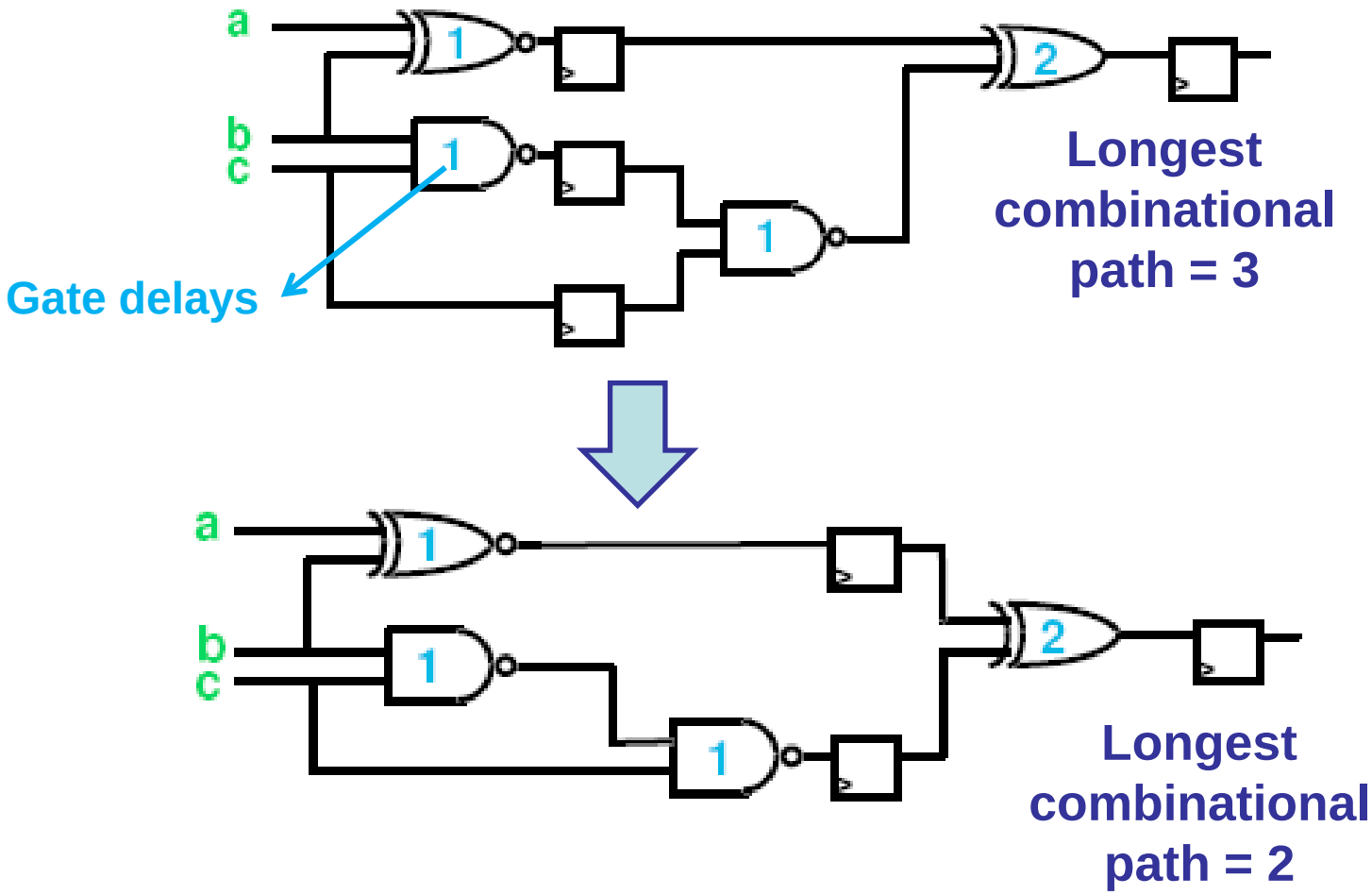
- Recall De Morgan's law?
  - Moving "bubbles" across gates

- It turns out you can do the same thing with flip-flops!
  - Does not change I/O behavior



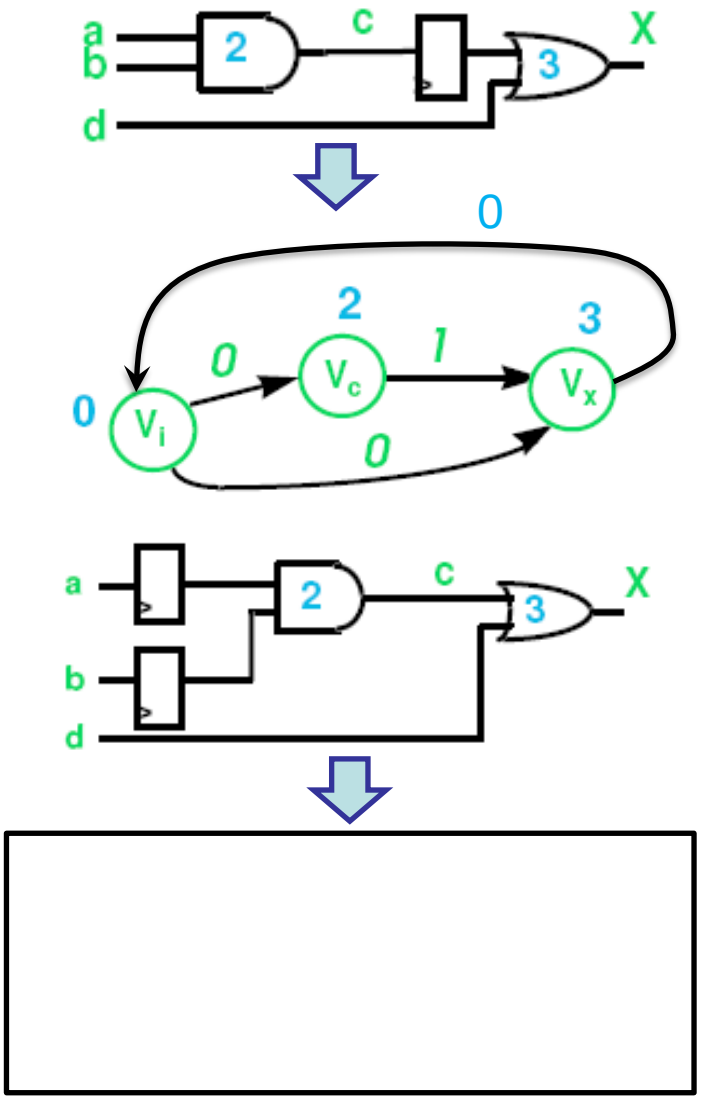
C. E. Leiserson, F. M. Rose, and J. B. Saxe, "Optimizing synchronous circuitry by retiming," Proc. 3rd Caltech Conf. on VLSI, 1983.

Retiming: Example

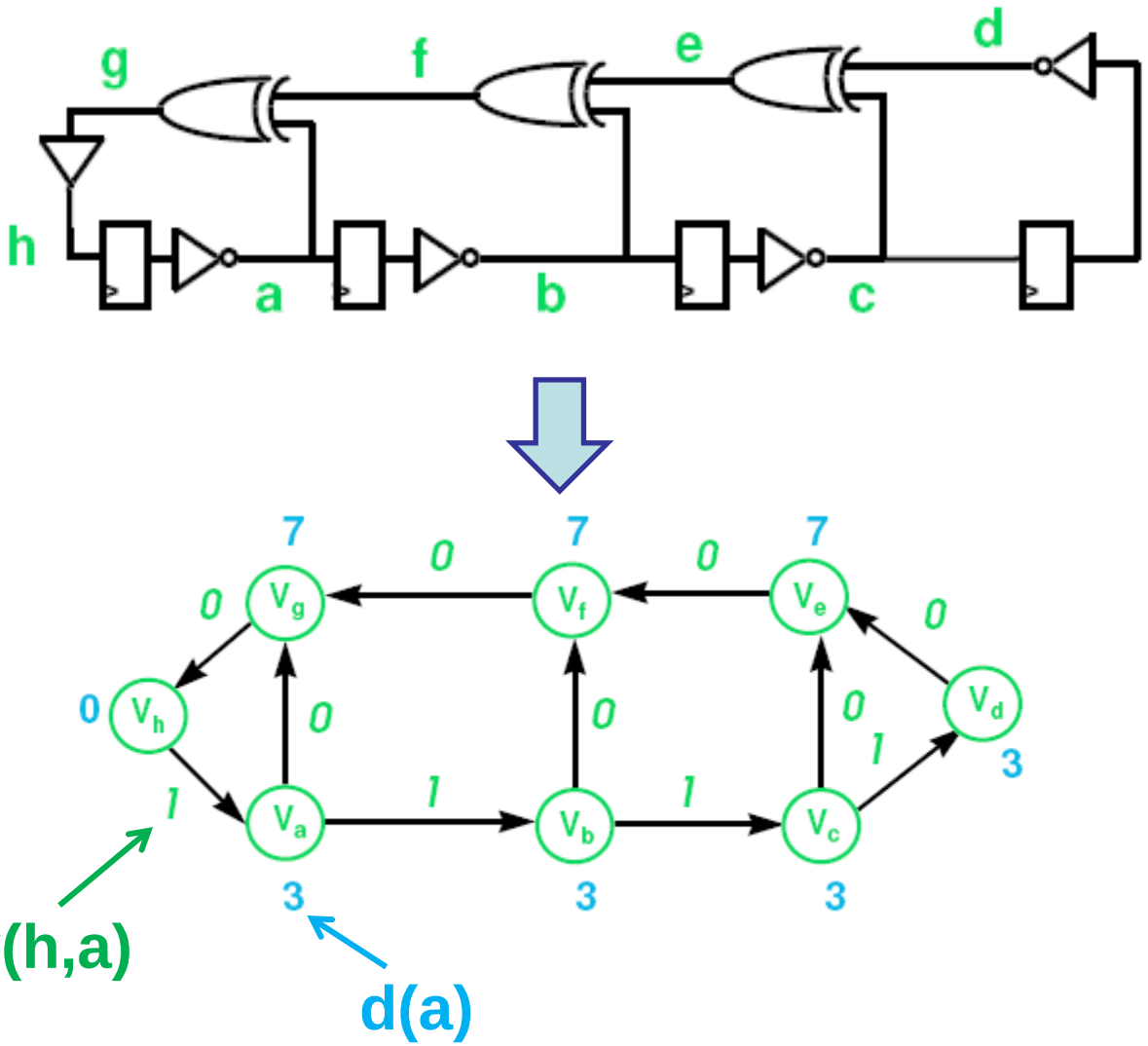


Performing Retiming Systematically: Circuit Model

- Circuit representation: Directed Graph G(V,E,d,w)
  - V ↔ set of gates
    - One additional vertex representing the "environment"
  - E ↔ set of wires
  - d(v) = delay of gate/vertex v
    - d(v) ≥ 0
  - w(e) = number of registers on edge e
    - w(e) ≥ 0

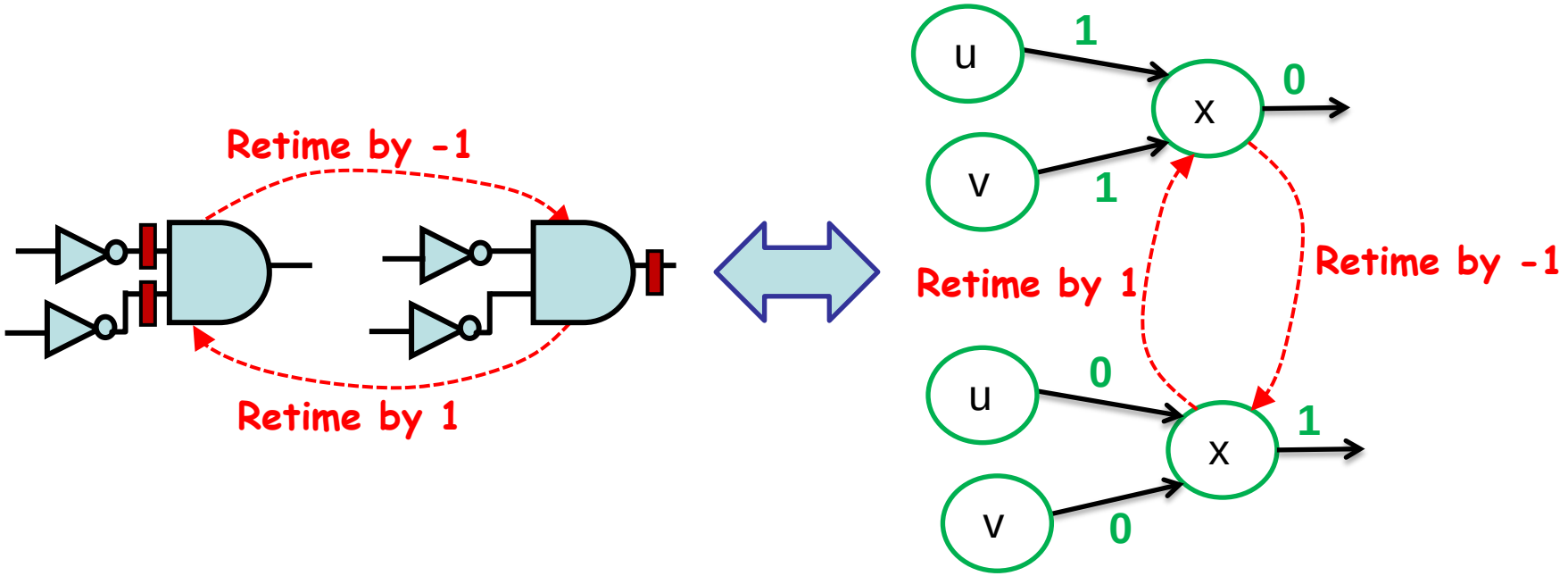


Circuit Model: Example



Retiming: Basic Operation

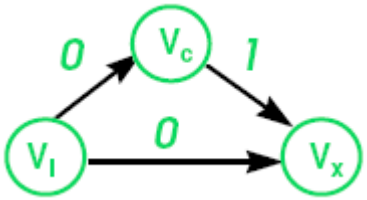
- Movement of FFs from input to output of a gate or vice versa
  - Applies to any block of combinational logic!
- Equivalent to labeling vertices in the graph with integers representing the number of FFs moved across the gate (backward movement = +ve)



Retiming: Basic Operation

- Vertex labeling r: V → Z
- New edge weights (after retiming) can be computed from vertex labels

Initial circuit

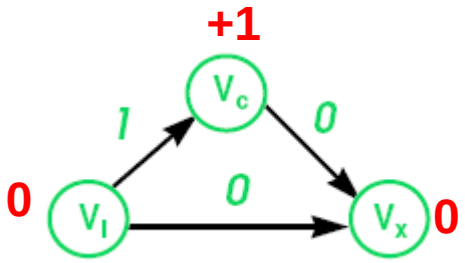


w(i,c) = 0
w(c,x) = 1
w(i,x) = 0

Retiming

r(i) = 0
r(c) = 1
r(x) = 0

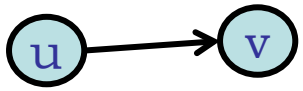
Retimed circuit



$w_r(i,c) = 1$
$w_r(c,x) = 0$
$w_r(i,x) = 0$

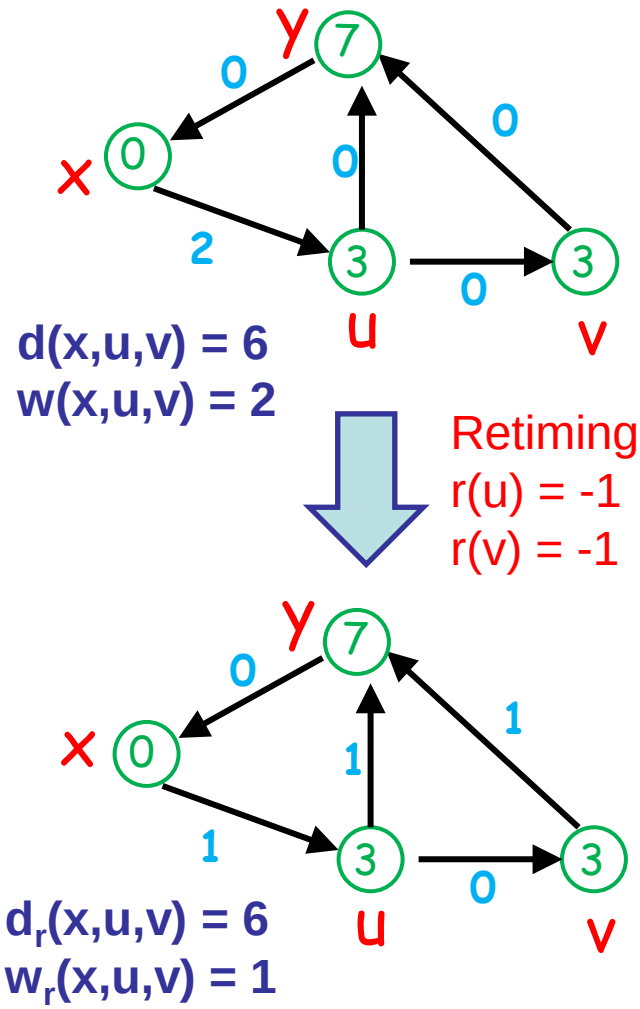
- General equation

$$w_r(u,v) = w(u,v) + r(v) - r(u)$$



Retiming: Path Weights and Delays

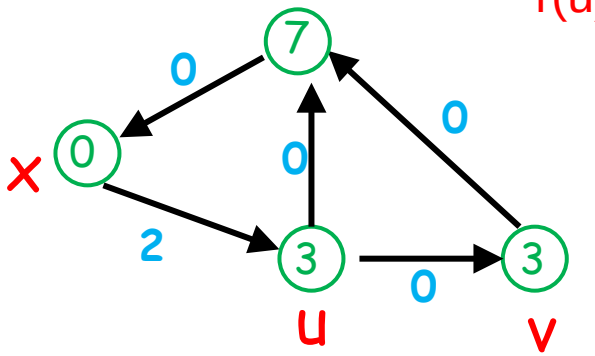
- **Path delay**: Sum of delays of gates along path
  - $d(v_i, \ldots, v_j) = \sum d(v_k), v_k \in (v_i, \ldots, v_j)$
- **Path weight**: Sum of edge weights along path
  - $w(v_i, \ldots, v_j) = \sum w(v_k, v_l),$ $v_k, v_l \in (v_i, \ldots, v_j)$
- What happens to a path's weight and delay under retiming?
  - Delay does not change
  - Weight changes, but only depends on labels of first and last nodes on path!



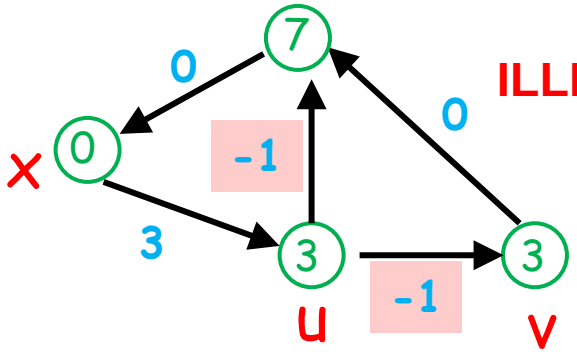
For a path p: $s \rightarrow t$, $w_r(p) = w(p) + r(t) - r(s)$

Legal Retiming

- When is a retiming legal?
  - A retiming r is **legal** if $w_r(e) \geq 0, \forall e \in E$
  - Cannot have any edges with "negative" number of FFs!

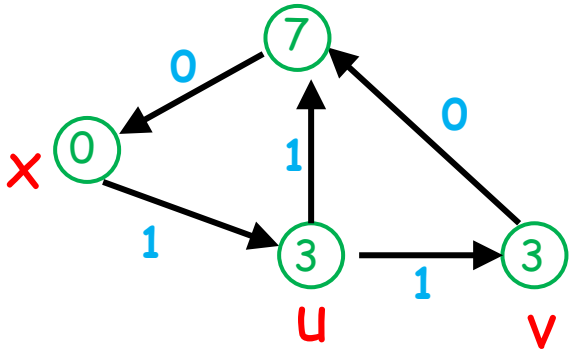


Retiming
r(u) = 1



ILLEGAL

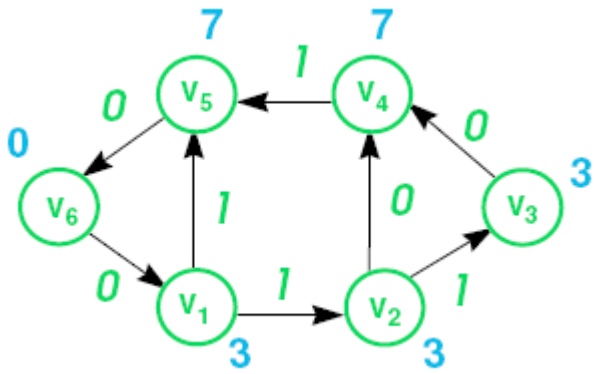
Retiming
r(u) = -1



LEGAL

Clock Period Constraints

- When does a network meet a given clock period?
- Definition:
  - $\underline{W}(v_i, v_j)$ = **min** $w(v_i, \ldots, v_j)$ over all paths from v_i to v_j
  - $\underline{D}(v_i, v_j)$ = **max** $d(v_i, \ldots, v_j)$ over all paths from v_i to v_j with weight $\underline{W}(v_i, v_j)$



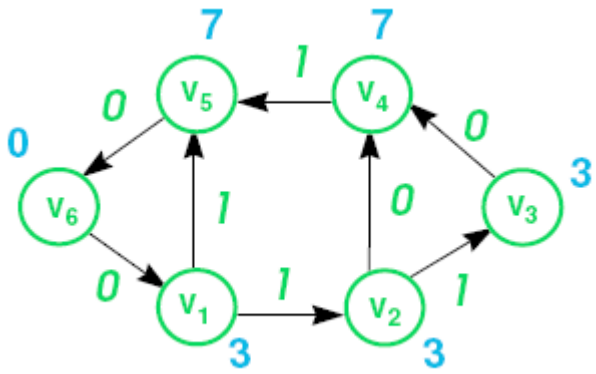
v_1, v_4

$\underline{W}(v_1, v_4) = 1$

$\underline{D}(v_1, v_4) = 13$

Clock Period Constraints

- **Theorem**: A network **G(V, E, d, w)** is timing feasible for a given clock period **T** if and only if **W(v_i,v_j)** ≥ **1** for all v_i,v_j ∈ **V** such that **D(v_i, v_j)** > **T**
  - **All paths longer than the clock period must have at least one FF**

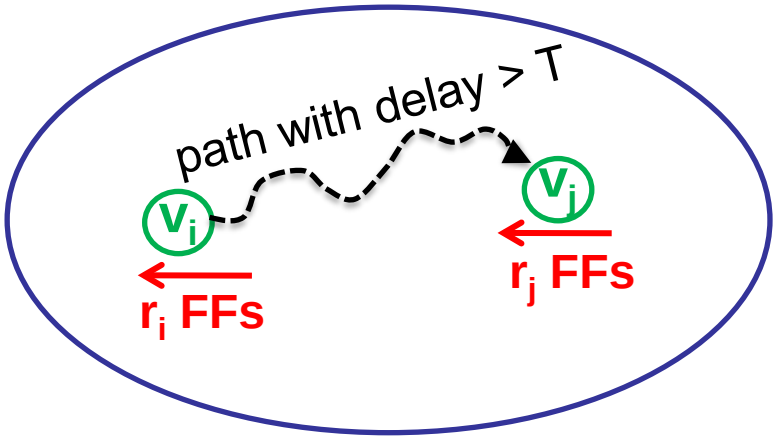


Is the network timing feasible at clock period 13?

Is the network timing feasible at clock period 7?

Re-timing and Clock Period Constraints

- **Theorem**: A retimed network $G_r(V, E, d, w)$ is timing feasible for a given clock period T if and only if
  - $r_i - r_j \leq w_{ij} \; \forall \; (v_i, v_j) \in E$
  - $r_i - r_j \leq \underline{W}(v_i, v_j) - 1 \; \forall \; v_i, v_j$ such that $\underline{D}(v_i, v_j) > T$



Retiming for Minimum Clock Period

- Problem Statement:
  - Given G (V, E, d, w), find a legal retiming r that minimizes

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

Retiming for Minimum Clock Period: Algorithm

1. Compute register weight matrix **W** and delay matrix **D**
   - **W(u,v)**: Minimum register count between u and v
   - **D(u,v)**: Maximum delay on a path with minimum register count between u and v
2. Sort elements in **D** and perform binary search on them. For each element **T**
   - Construct constraints for legal and timing-feasible retiming
   - Solve constraints
3. The lowest **T** for which a legal retiming exists is the minimum clock period

Retiming for Minimum Clock Period: Algorithm

```
Retime_min_clock(G) {
        W,D = compute_WD(G);
        candidate_periods[ ] = sort(elements of D);
        for each T determined by binary search on candidate_periods [ ] {
                I = construct_inequalities (G,T) ;
                r = solve_inequalities(I);
        }
        return (r, T*) determined by last successful solution of inequalities;
}
```

Computing W and D matrices

- Apply Floyd-Warshall algorithm for solving all-pairs shortest-paths problem in an edge-weighted graph

Floyd-Warshall (G = (V, E, W))

1. for i ← 1 to |V| do
2. for j ← 1 to |V| do
3. If $e_{ij} \in E$ *then* $u_{ij} \leftarrow w(e_{ij})$
   else $u_{ij} \leftarrow \infty$;
4. for k ← 1 to |V| do
5. for i ← 1 to |V| do
6. for j ← 1 to |V| do
7. If $u_{ij} > u_{ik} + u_{kj}$ then $u_{ij} = u_{ik} + u_{kj}$;

Complexity: $O(|V|^3)$

How can we compute both W and D in one shot using the Floyd-Warshall algorithm?

Computing W and D matrices

- Algorithm **compute_WD** (G = (V,E,d,w))

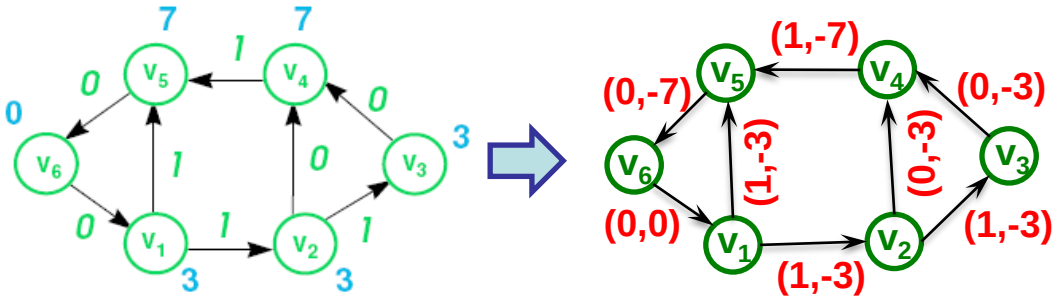
  1. Weight each edge e: u →v in E with the ordered pair (w(e), -d(u))

  2. Using the weights from step 1, compute the weight of the shortest path joining each connected pair of vertices by using the Floyd-Warshall all pairs shortest-paths algorithm.

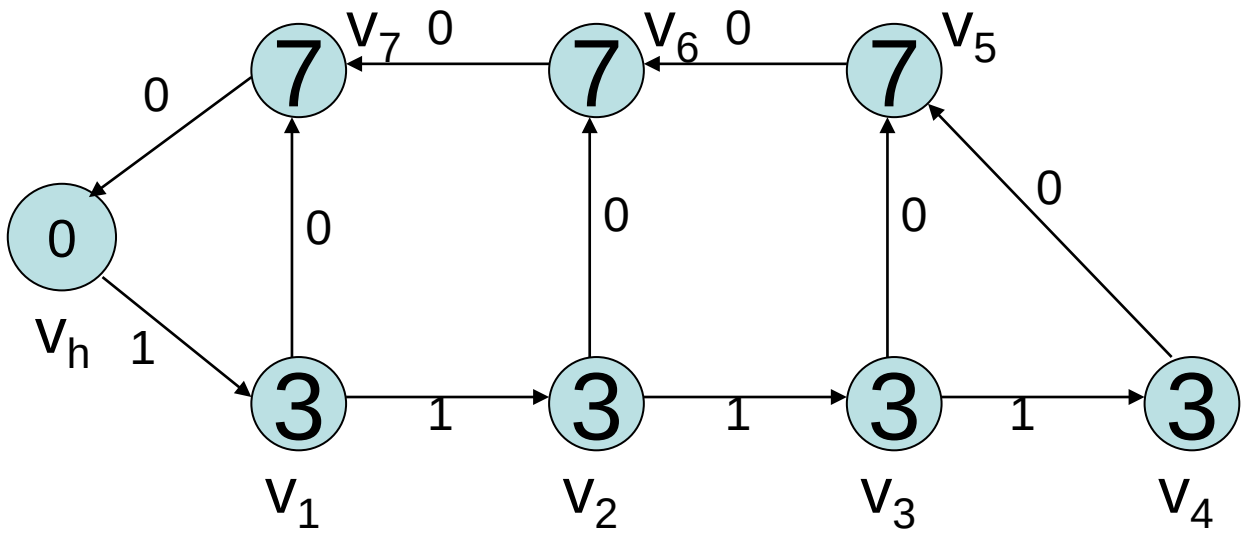
     (add weights by performing component-wise addition. compare weights using w as the primary key and -d as the secondary key)

  3. For each shortest-path weight(x, y) between two vertices u and v, set W(u,v) ← x and D(u,v) ← d(v) – y.

Complexity: $O(|V|^3)$



Computing W and D matrices: Example



W	vh	v1	v2	v3	v4	v5	v6	v7
vh	0	1	2	3	4	3	2	1
v1	0	0	1	2	3	2	1	0
v2	0	1	0	1	2	1	0	0
v3	0	1	2	0	1	0	0	0
v4	0	1	2	3	0	0	0	0
v5	0	1	2	3	4	0	0	0
v6	0	1	2	3	4	3	0	0
v7	0	1	2	3	4	3	2	0

D	vh	v1	v2	v3	v4	v5	v6	v7
vh	0	3	6	9	12	16	13	10
v1	10	3	6	9	12	16*	13	10
v2	17	20	3	6	9	13	10	17
v3	24	27	30	3	6	10	17	24
v4	24	27	30	33	3	10	17	24
v5	21	24	27	30	33	7	14*	21
v6	14	17	20	23	26	30	7	14*
v7	7	10	13	16*	19	23	20*	7

Constructing Inequalities (Retiming Constraints)

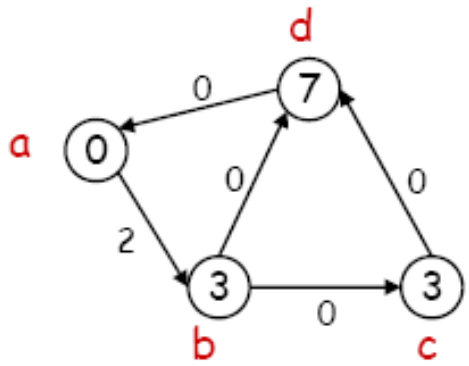
- Constraints that guarantee a legal retiming for a given clock period

$r_i - r_j \leq w_{ij} \; \forall \; (v_i, v_j) \in E$ → Cannot have "negative" FFs

$r_i - r_j \leq \underline{W}(v_i, v_j) - 1 \; \forall \; v_i, v_j$ such that $\underline{D}(v_i, v_j) > T$

All paths of delay > T have at least 1 FF in the retimed circuit

Constructing the Retiming Constraints



W

	a	b	c	d
a	0	2	2	2
b	0	0	0	0
c	0	2	0	0
d	0	2	2	0

D

	a	b	c	d
a	0	3	6	13
b	13	3	6	13
c	10	13	3	10
d	7	10	13	7

Clock period = 7

r(u) – r(v) ≤ w(u,v)

$r(a)-r(b) \le 2$
$r(b)-r(c) \le 0$
$r(b)-r(d) \le 0$
$r(c)-r(d) \le 0$
$r(d)-r(a) \le 0$

r(u) – r(v) ≤ W(u,v) – 1
∀ u,v such that D(u,v) > 7

$r(a)-r(d) \le 1$
$r(b)-r(a) \le -1$
$r(b)-r(d) \le -1$
$r(c)-r(a) \le -1$
$r(c)-r(b) \le 1$
$r(c)-r(d) \le -1$
$r(d)-r(a) \le 1$
$r(d)-r(c) \le 1$

Solving the Retiming Constraints

- Retiming inequalities form an Integer Linear Program (ILP)
  - Solving ILPs in general is NP-complete
  - However, the constraints have a special structure ($r(u) - r(v) \leq$ constant)

Theorem: Let S be a set of m linear inequalities in variables $x_1, x_2, \ldots, x_n$ of the form $x_j - x_i \leq a_{ij}$, where the a_{ij} are given real constants. The problem of determining feasible values for the unknowns x_i, if they exist, can be solved using the Bellman-Ford single-source shortest-path algorithm

Solving the Retiming Constraints

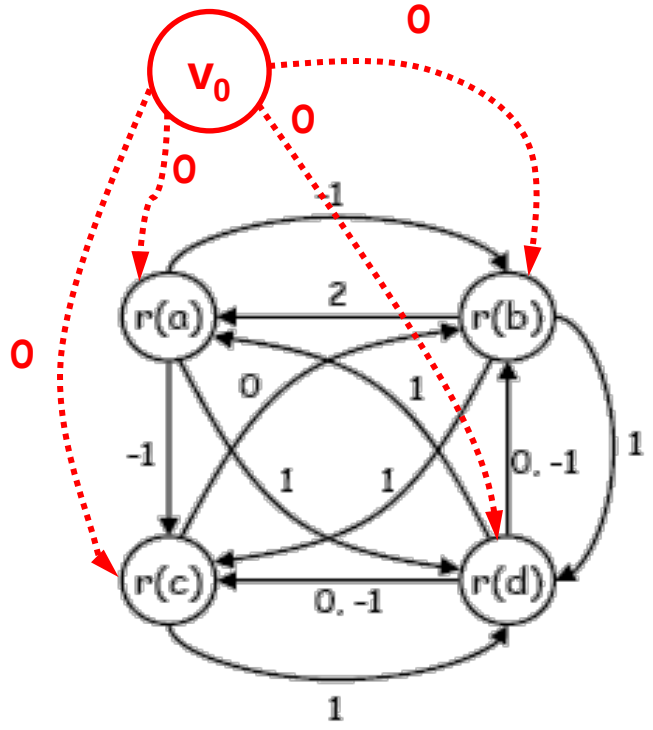
1. Represent the constraints as a directed graph
   - Vertices are the same, but edges represent constraints
2. Add a source vertex v_0 and add edges from v_0 to each vertex in the graph, with zero weight
3. Find shortest path from v_0 to all other vertices

$r(u) - r(v) \le \underline{W}(u,v) - 1$
$\forall$ u,v such that D(u,v) > 7

$r(u) - r(v) \le w(u,v)$

$r(a)-r(b) \le 2$
$r(b)-r(c) \le 0$
$r(b)-r(d) \le 0$
$r(c)-r(d) \le 0$
$r(d)-r(a) \le 0$

$r(a)-r(d) \le 1$
$r(b)-r(a) \le -1$
$r(b)-r(d) \le -1$
$r(c)-r(a) \le -1$
$r(c)-r(b) \le 1$
$r(c)-r(d) \le -1$
$r(d)-r(a) \le 1$
$r(d)-r(c) \le 1$

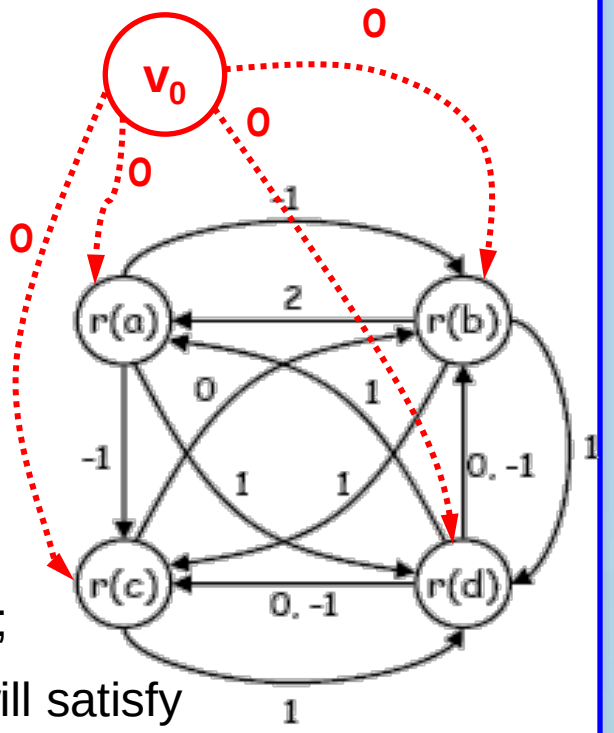


Solving the Retiming Constraints

1. Represent the constraints as a directed graph
2. Add a source vertex v_0 and add edges from v_0 to each vertex in the graph, with zero weight
3. Find shortest path from v_0 to all other vertices

Algorithm to find shortest path from v_0 to v_i

1. for i ← 0 to n do $u_i \leftarrow 0$;
2. for k ← 1 to n do
3. for $e_{ij} \in E$ *do*
4. If $u_j > u_i + w(e_{ij})$ then $u_j \leftarrow u_i + w(e_{ij})$;
5. *unsatisfiable* ← FALSE;
6. for each $e_{ij} \in E$ *do*
7. If $u_j > u_i + w(e_{ij})$ then *unsatisfiable* ← TRUE;

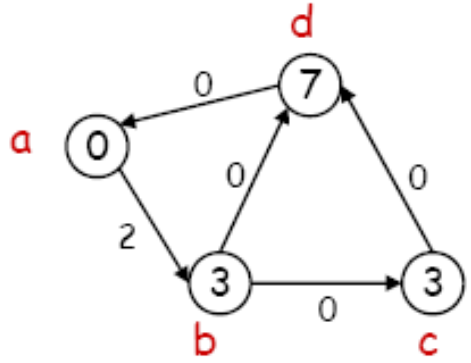


- At the end, if *unsatisfiable* = FALSE, then **$r(v_i) = u_i$** will satisfy retiming constraints;

  If *unsatisfiable* = TRUE, the graph contains a cycle with negative total weight, which means the constraints S are unsatisfiable.

Complexity: O(VE)

Solving the Retiming Constraints



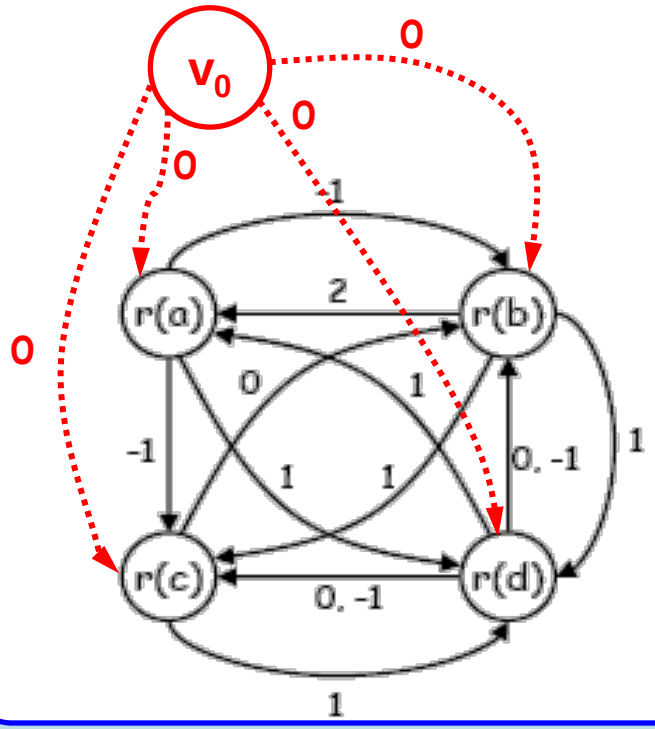
Clock Period = 7

$r(u) - r(v) \le w(u,v)$

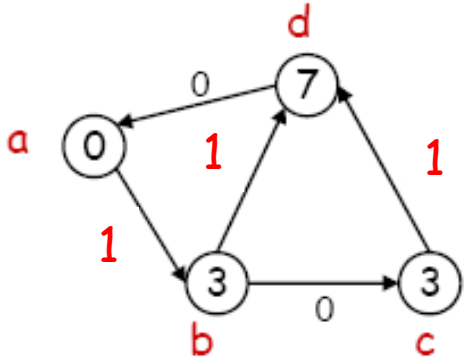
$r(a)-r(b) \le 2$
$r(b)-r(c) \le 0$
$r(b)-r(d) \le 0$
$r(c)-r(d) \le 0$
$r(d)-r(a) \le 0$

$r(u) - r(v) \le \underline{W}(u,v) - 1$
$\forall$ u,v such that $D(u,v) > 7$

$r(a)-r(d) \le 1$
$r(b)-r(a) \le -1$
$r(b)-r(d) \le -1$
$r(c)-r(a) \le -1$
$r(c)-r(b) \le 1$
$r(c)-r(d) \le -1$
$r(d)-r(a) \le 1$
$r(d)-r(c) \le 1$



r(a)=0, r(b) = -1, r(c) = -1, r(d) = 0



Retiming for Minimum Clock Period: Algorithm Revisited

1. Compute register weight matrix W and delay matrix D
2. Sort elements in D and perform binary search on them. For each element T
   - Construct constraints for legal and timing-feasible retiming
   - Consruct constraint graph and compute shortest paths using Bellman-Ford algorithm
3. The lowest T for which a legal retiming exists is the minimum clock period. Use the values for path lengths found by the Bellman-Ford algorithm as the optimal retiming labels for vertices

Complexity: $O(V^3 \log V)$

Summary: Retiming

- Retiming re-positions registers in a sequential circuit
  - Powerful transformation – changes the combinational logic boundaries
  - Retiming for minimum clock period
    - Lends to an elegant formulation and use of efficient graph algorithms (Floyd-Warshall, Bellman-Ford)
  - Other variants:
    - Retiming to minimize number of registers
    - Considering load-dependent delay model
    - Subsequent research used retiming to improve power, testability, …

General Principle

- Look for special cases of a problem that can be efficiently solved
  - Retiming constraints form an ILP, but have a special structure that enables a much more efficient algorithm

Reading

- De Micheli, Ch. 9.1-9.2, 9.3.1-9.3.2
  - We will not talk about networks of FSMs
- Additional References
  - Pranav Ashar, Srinivas Devadas, Arthur Richard Newton, “Sequential logic synthesis,” Kluwer Academic Publishers, 1991.
  - T. *Villa*, T. *Kam*, R. K. *Brayton*, and A. L. *Sangiovanni*-Vincentelli. Synthesis of FSMs: Logic Optimization. Kluwer Academic Publishers, 1997.
  - G. De Micheli, R. K. Brayton, and A. L. Sangiovanni-Vincentelli. Optimal State Assignment for Finite State Machines. *IEEE Transactions on Computer-Aided Design, pp. 269-285, July 1985.*
  - S. *Devadas*, "*Mustang*: State Assignment of Finite State Machines Targeting Multi-Level Logic Implementation" *IEEE Transactions on Computer-Aided Design, pp.* 1290-1300, Dec. 1988
  - C. E. Leiserson, F. M. Rose, and J. B. Saxe, “Optimizing synchronous circuitry by retiming,” Proc. 3rd Caltech Conf. on VLSI, 1983.
  - C. E. Leiserson and J. B. Saxe, “Retiming synchronous circuitry,” Algorithmica, Springer New York, Issue: Volume 6, Number 1, December 1991, Pages: 5 - 35
  - “Validity of retiming sequential circuits,” Singhal, Vigyan; Pixley, Carl; Rudell, Richard L; Brayton, Robert K, Proc. Design Automaton Conference, pp. 316-321. 1995