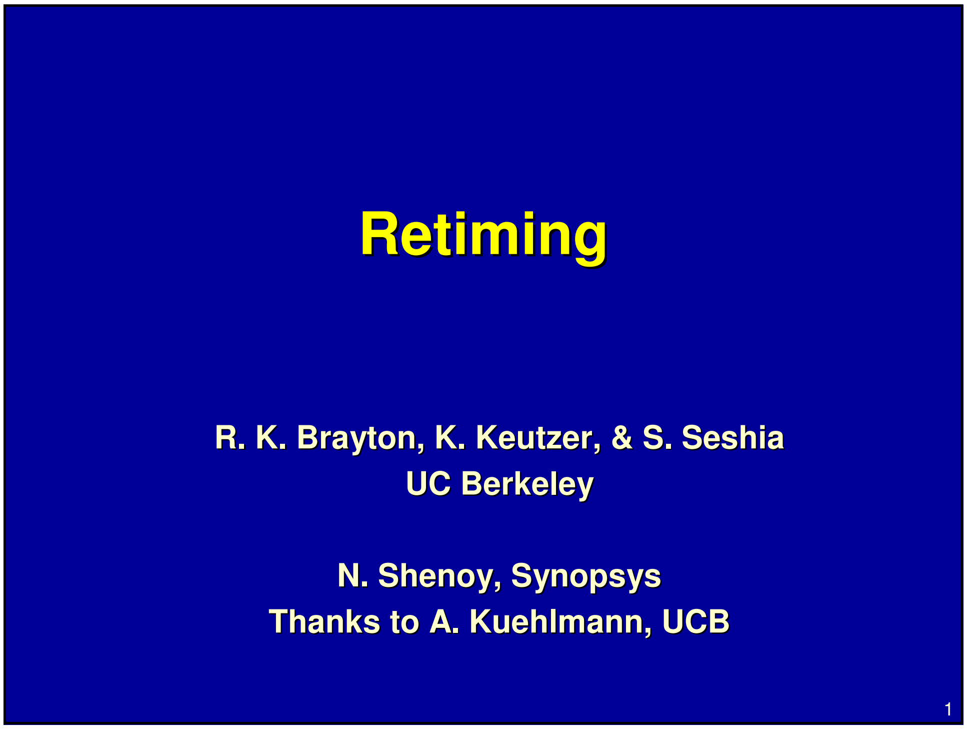
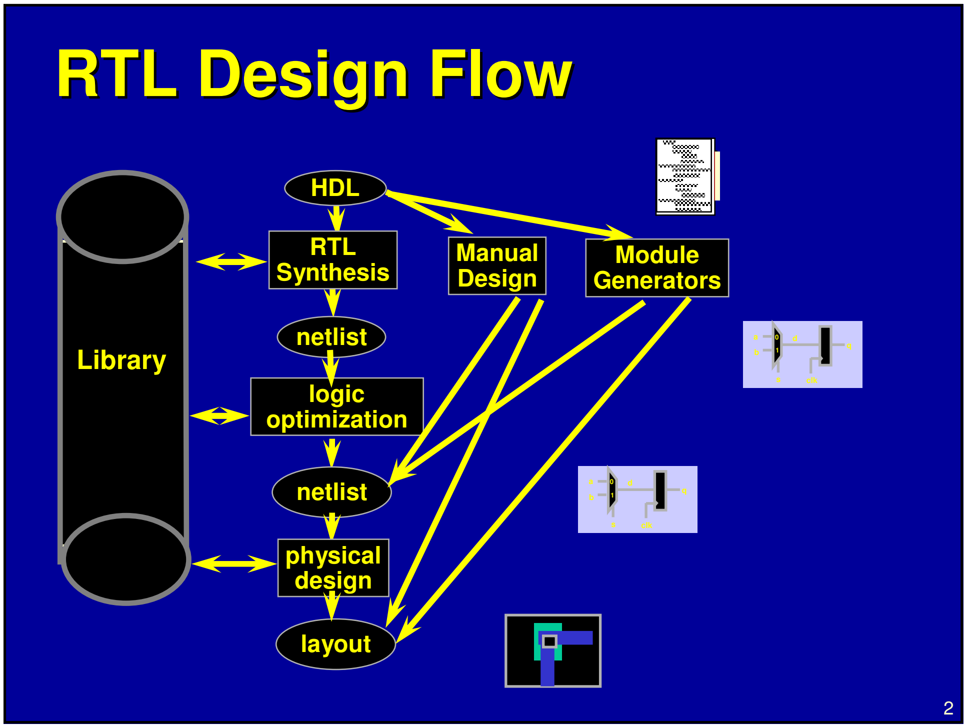


Retiming

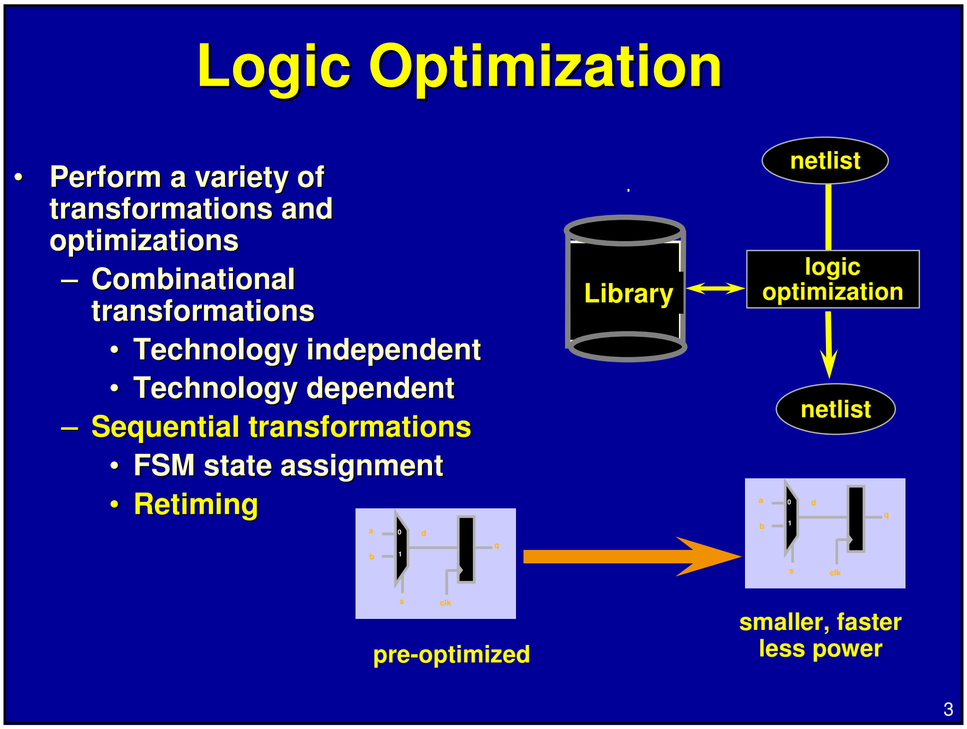
R. K. Brayton, K. Keutzer, & S. Seshia
UC Berkeley

N. Shenoy, Synopsys
Thanks to A. Kuehlmann, UCB



RTL Design Flow

HDL
RTL Synthesis
Manual Design
Module Generators
Library
netlist
logic optimization
netlist
physical design
layout



Logic Optimization

- Perform a variety of transformations and optimizations
  - – Combinational transformations
    - Technology independent
    - Technology dependent
  - – Sequential transformations
    - FSM state assignment
    - Retiming

netlist

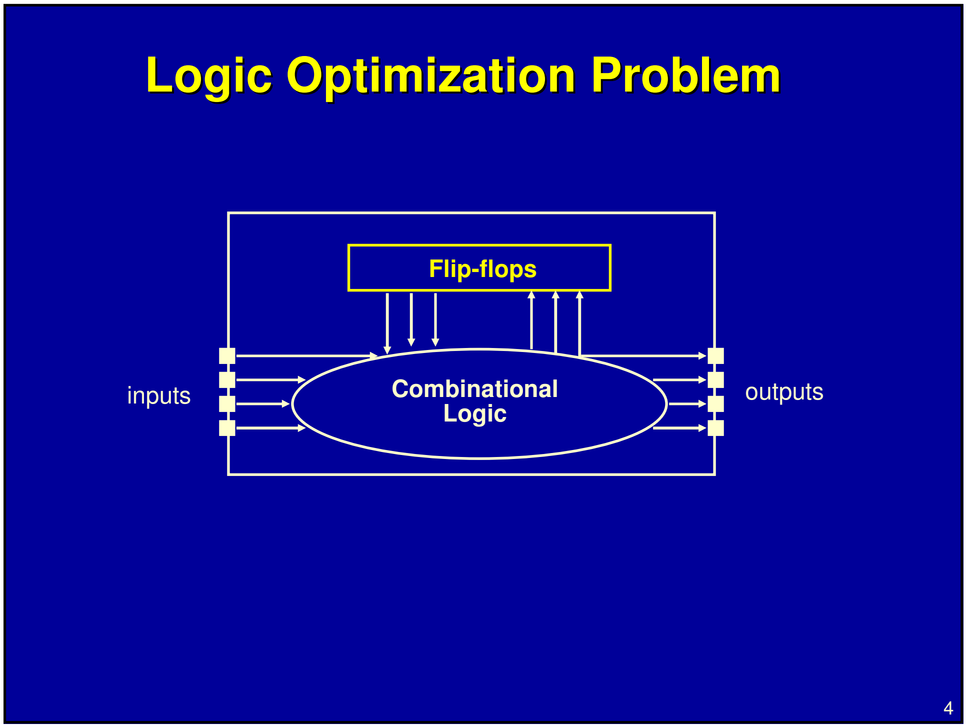
Library

logic optimization

netlist

pre-optimized

smaller, faster
less power



Logic Optimization Problem

Flip-flops

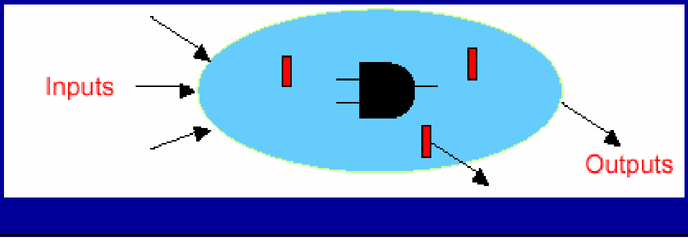
inputs

Combinational Logic

outputs

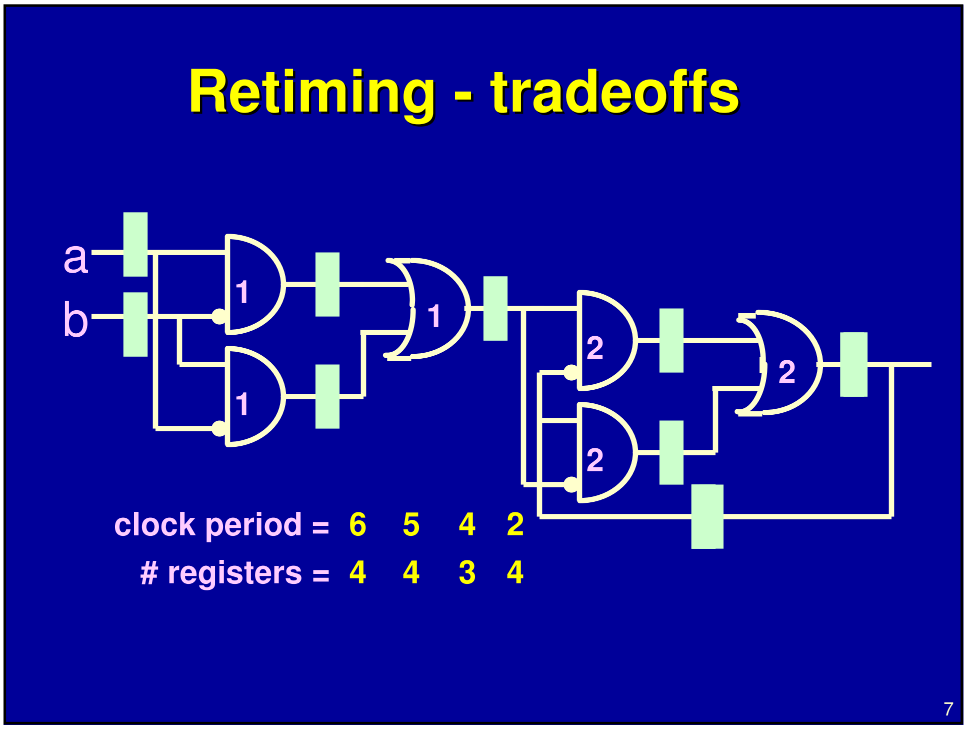
What about the Registers?

- Pure combinational optimization can be suboptimal since relations across register boundaries are disregarded
- Optimize a sequential circuit by optimally placing registers. Move register(s) so that
  - clock cycle decreases, or number of registers decreases and
  - input-output behavior is preserved
- Also, can combine retiming with combinational optimization techniques
  - Move latches out of the way temporarily
  - optimize larger blocks of combinational



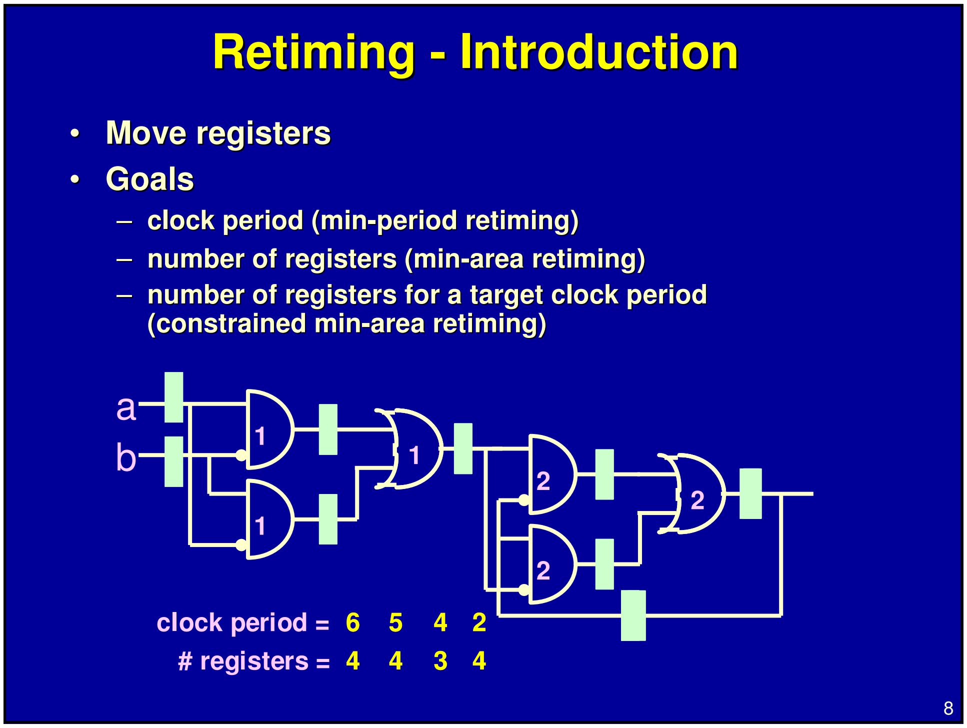
Lecture Outline

- Why is retiming important?
- Basic Model and Algorithms
- Combining with Combinational Optimization



Retiming - tradeoffs

clock period =	6	5	4	2
# registers =	4	4	3	4



Retiming - Introduction

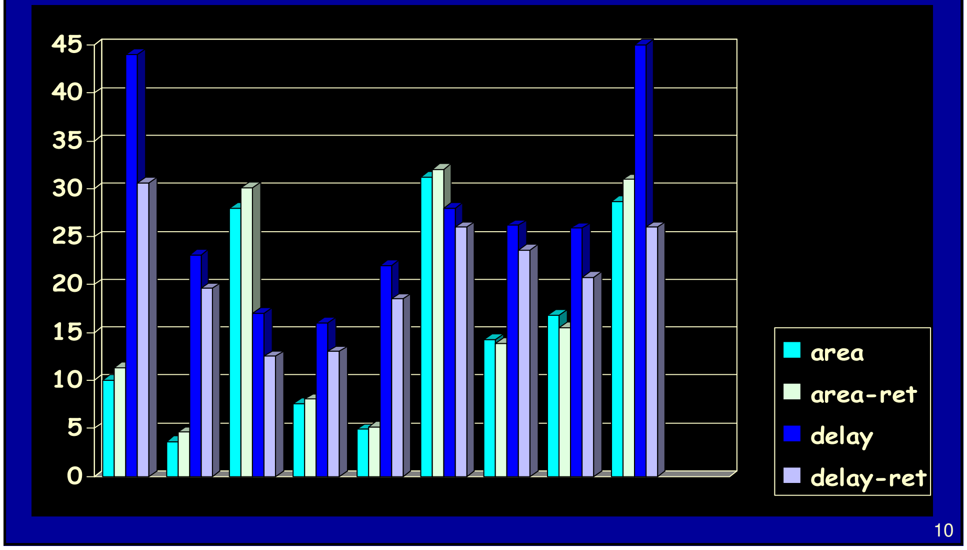
- Move registers
- Goals
  - clock period (min-period retiming)
  - number of registers (min-area retiming)
  - number of registers for a target clock period (constrained min-area retiming)

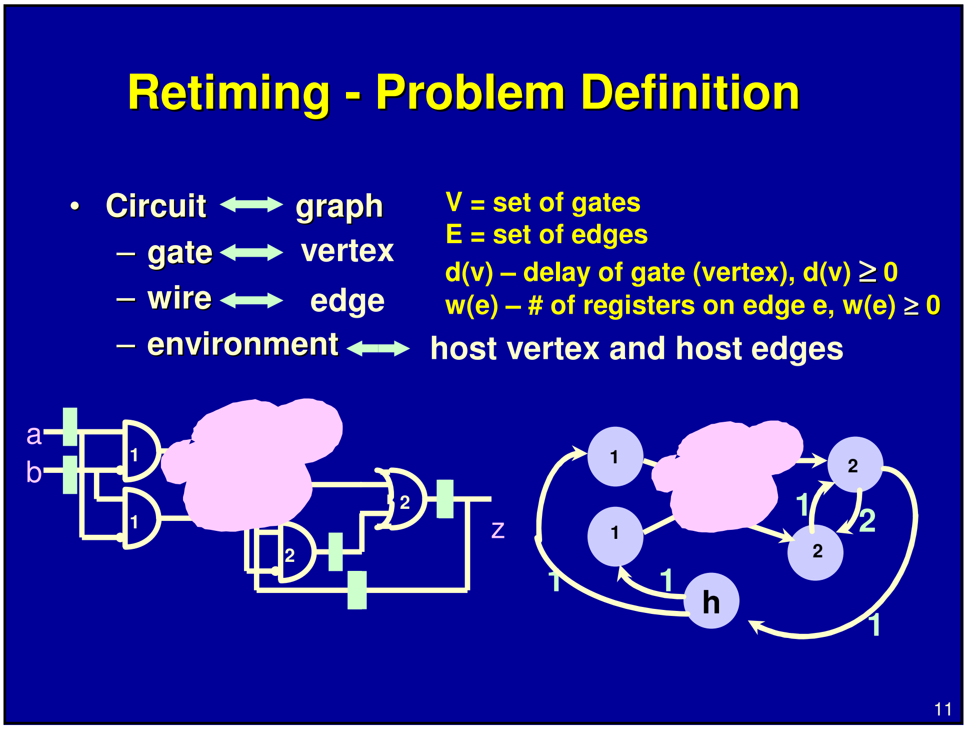
clock period =	6	5	4	2
# registers =	4	4	3	4

Importance of Retiming

- Practical sequential optimization
- Global optimality for clock period and register positioning
- Must for HDL synthesis
  - lowers dependency on user description
- Low power strategy
  - decrease #registers with no loss in performance

Practical Importance of Retiming

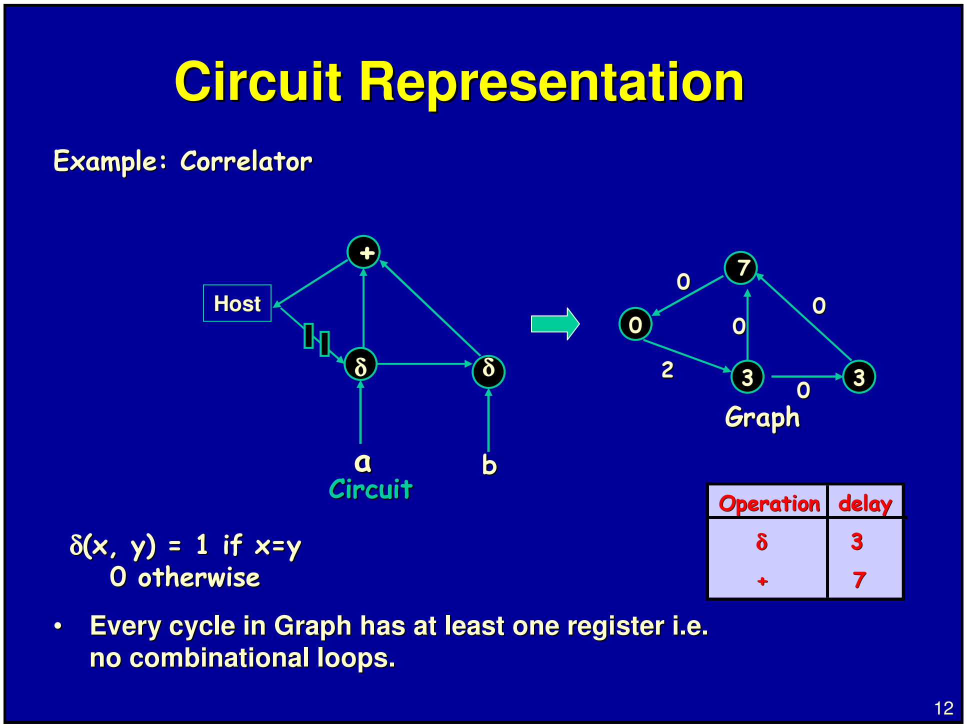




Retiming - Problem Definition

- Circuit ⟷ graph
  - gate ⟷ vertex
  - wire ⟷ edge
  - environment ⟷ host vertex and host edges

V = set of gates
E = set of edges
d(v) – delay of gate (vertex), d(v) ≥ 0
w(e) – # of registers on edge e, w(e) ≥ 0



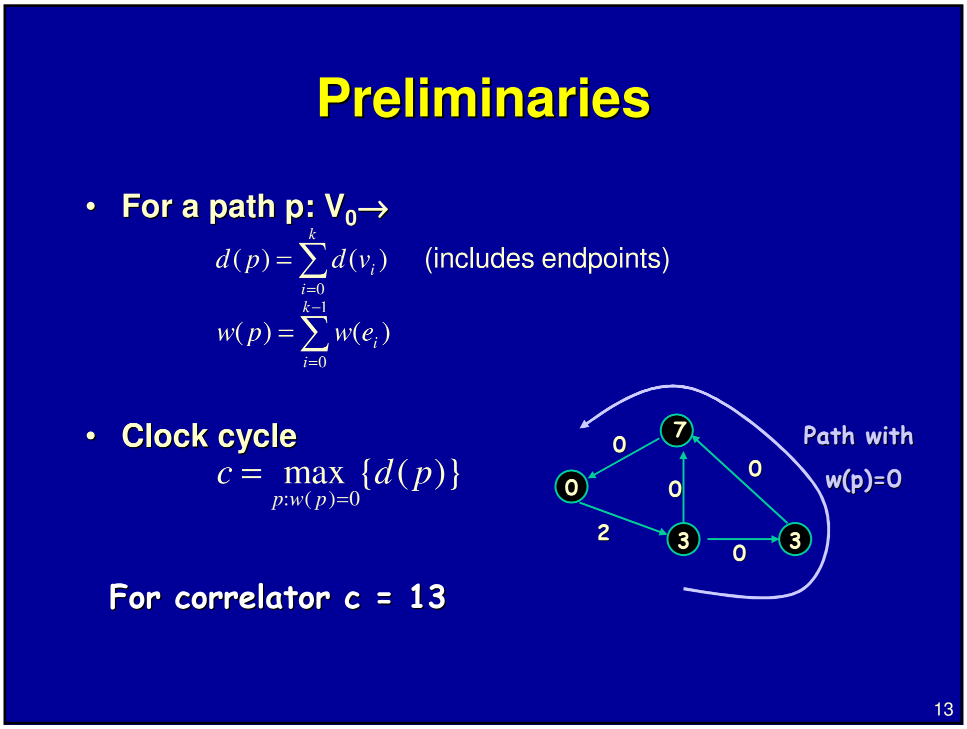
Circuit Representation

Example: Correlator

Operation	delay
δ	3
+	7

δ(x, y) = 1 if x=y
0 otherwise

- Every cycle in Graph has at least one register i.e. no combinational loops.



Preliminaries

- For a path p: $V_0 \rightarrow$

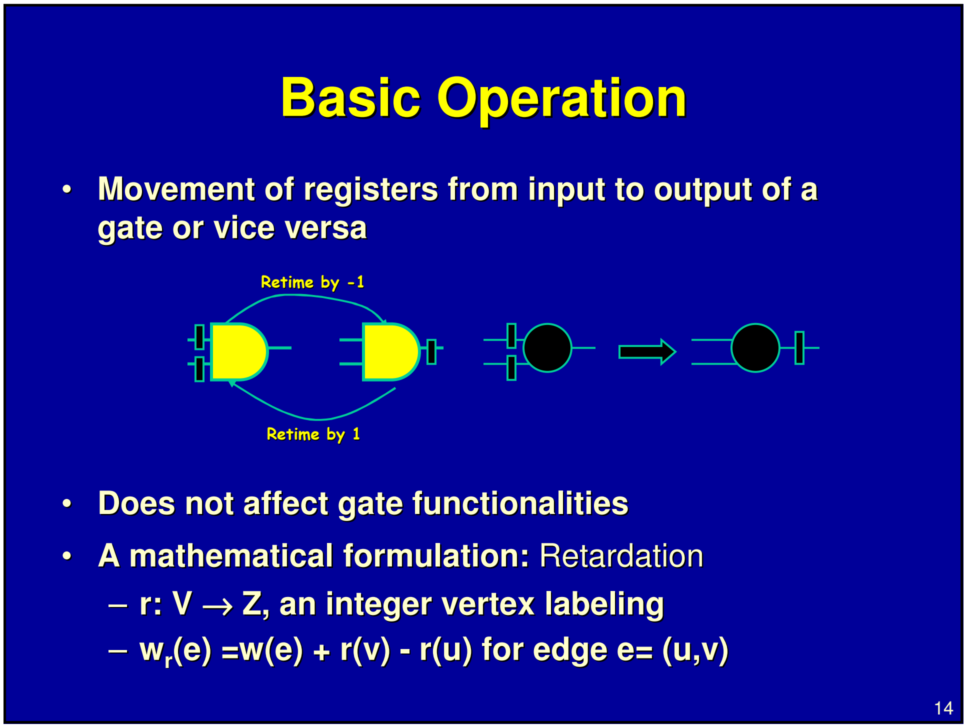
$$d(p) = \sum_{i=0}^{k} d(v_i) \quad \text{(includes endpoints)}$$

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

- Clock cycle

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

For correlator c = 13



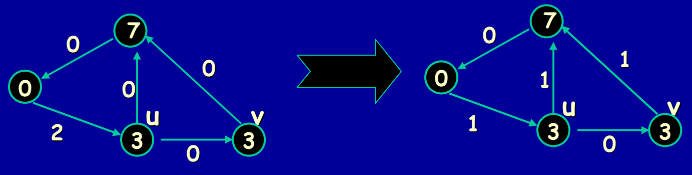
Basic Operation

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

- Does not affect gate functionalities
- A mathematical formulation: Retardation
  - r: V → Z, an integer vertex labeling
  - $w_r(e) = w(e) + r(v) - r(u)$ for edge e= (u,v)

Basic Operation

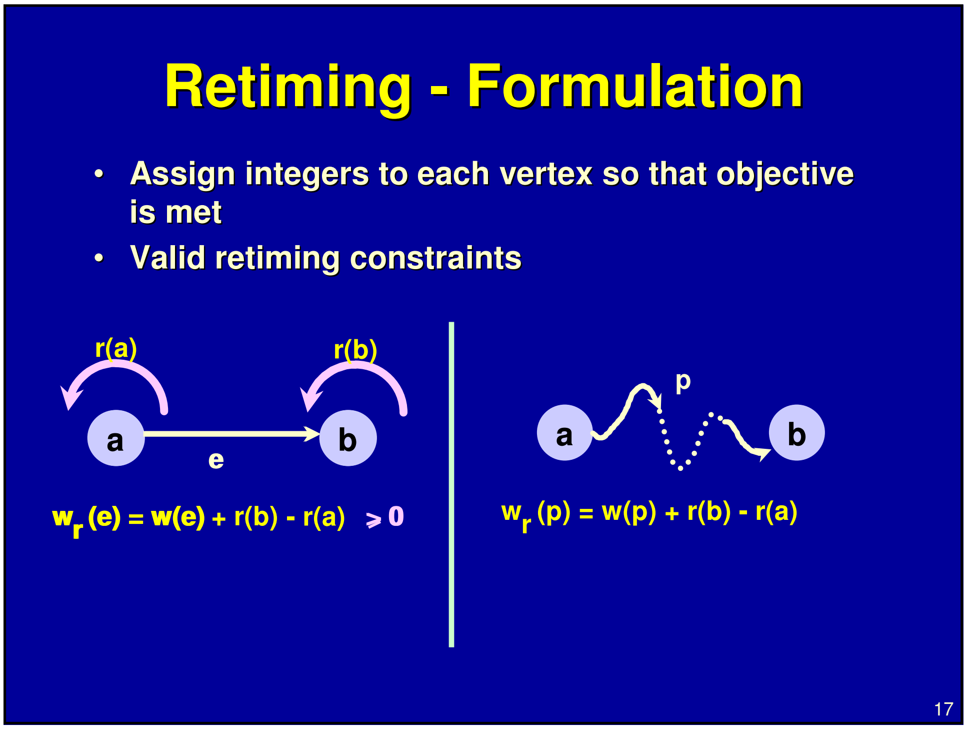
- Thus in the example, r(u) = -1, r(v) = -1 results in



- For a path p: s→t, $W_r(p) = w(p) + r(t) - r(s)$
- Retiming
  - r: V→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 $w_r(e) \geq 0, \forall e \in E$

Retiming - Assumptions

- Each loop in circuit contains at least one register
- Circuit uses single clock and edge-triggered elements (identical skew)
- Gate delay is constant (and non-negative)
- Registers are ideal (set-up, drive independent of load)
- Any power-up state of the design can be safely handled by the environment (initial state assumption)

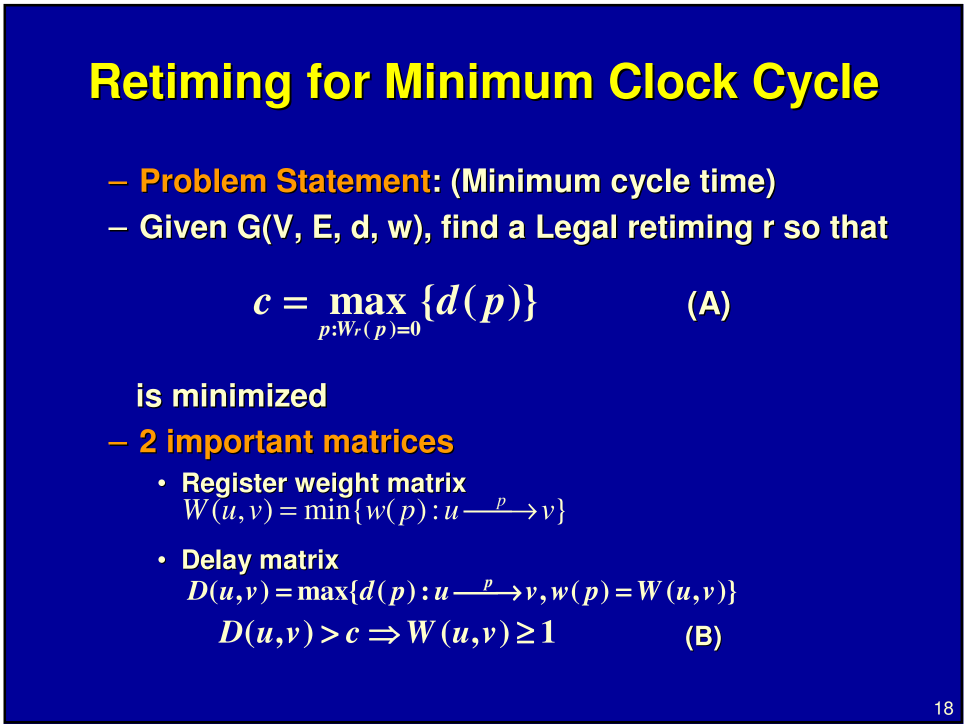


Retiming - Formulation

- Assign integers to each vertex so that objective is met
- Valid retiming constraints

$w_r(e) = w(e) + r(b) - r(a) \geq 0$

$w_r(p) = w(p) + r(b) - r(a)$



Retiming for Minimum Clock Cycle

– Problem Statement: (Minimum cycle time)

– Given G(V, E, d, w), find a Legal retiming r so that

$$c = \max_{p:W_r(p)=0} \{d(p)\} \qquad \text{(A)}$$

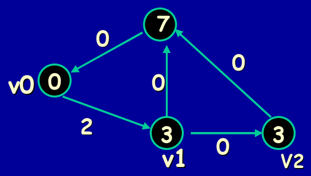
is minimized

– 2 important matrices

- Register weight matrix
  $W(u,v) = \min\{w(p) : u \xrightarrow{p} v\}$
- Delay matrix
  $D(u,v) = \max\{d(p) : u \xrightarrow{p} v, w(p) = W(u,v)\}$

$$D(u,v) > c \Rightarrow W(u,v) \geq 1 \qquad \text{(B)}$$

Retiming for Minimum Clock Cycle



W – register path weight matrix, min # of registers on all paths between u and v
D – path delay matrix, max delay among all paths between u and v with W(u,v) registers

W

$$\begin{array}{c|cccc} & V0 & V1 & V2 & V3 \\ \hline V0 & 0 & 2 & 2 & 2 \\ V1 & 0 & 0 & 0 & 0 \\ V2 & 0 & 2 & 0 & 0 \\ V3 & 0 & 2 & 2 & 0 \end{array}$$

D

$$\begin{array}{cccc|c} V0 & V1 & V2 & V3 & \\ \hline 0 & 3 & 6 & 13 & V0 \\ 13 & 3 & 6 & 13 & V1 \\ 10 & 13 & 3 & 10 & V2 \\ 7 & 10 & 13 & 7 & V3 \end{array}$$

$C \leq \alpha \Leftrightarrow \forall p$, if $d(p) > \alpha$ then $w(p) \geq 1$

i.e. for the clock cycle to be less than α there must be a latch in the path

Conditions for Retiming

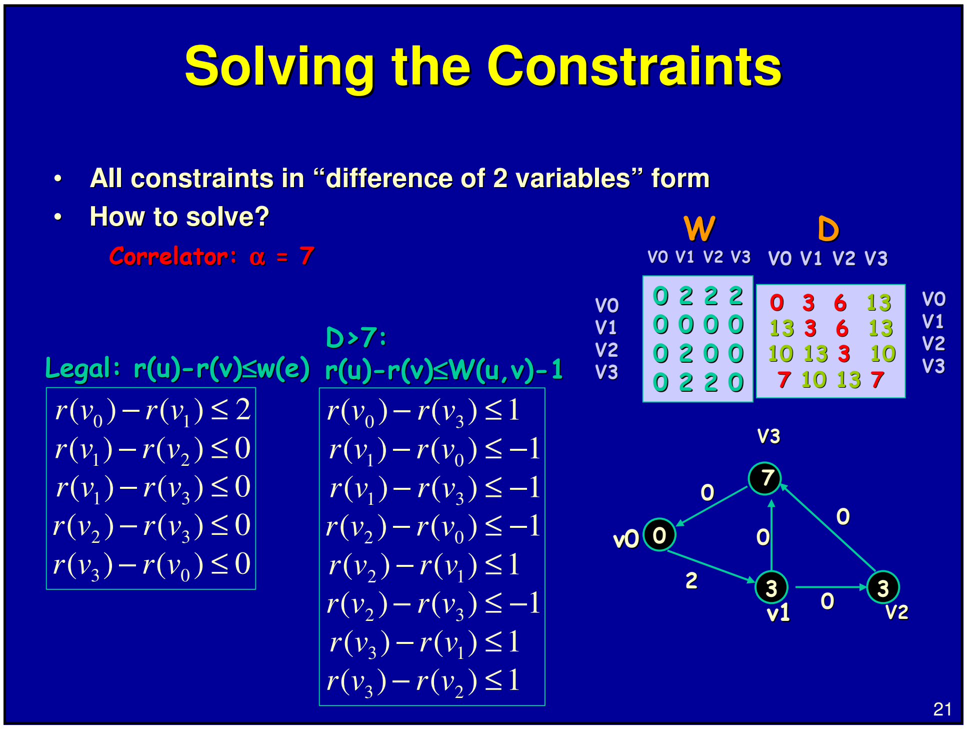
- Suppose we need to check if a retiming exists for a clock cycle α
- Legal retiming: $w_r(e) \geq 0$ for all e. Hence
  $w_r(e) = w(e) + r(v) - r(u) \geq 0$ or
  $r(u) - r(v) \leq w(e)$
- For all paths p: u → v such that $d(p) \geq \alpha$, we require $w_r(p) \geq 1$
  - Thus

$$\begin{aligned} 1 \leq w_r(p) &= \sum_{i=0}^{k-1} w_r(e_i) \\ &= \sum_{i=0}^{k-1} [w(e_i) + r(v_{i+1}) - r(v_i)] \\ &= w(p) + r(v_k) - r(v_0) \\ &= w(p) + r(v) - r(u) \end{aligned}$$

Or take the least w(p) (tightest constraint) $r(u)-r(v) \leq W(u,v)-1$

i.e. there are many paths *p*, choose the *p* that gives tightest constraint

Note: we just need to apply it to (u, v) such that $D(u,v) > \alpha$



Solving the Constraints

- All constraints in “difference of 2 variables” form
- How to solve?

Correlator: α = 7

W

	V0	V1	V2	V3
V0	0	2	2	2
V1	0	0	0	0
V2	0	2	0	0
V3	0	2	2	0

D

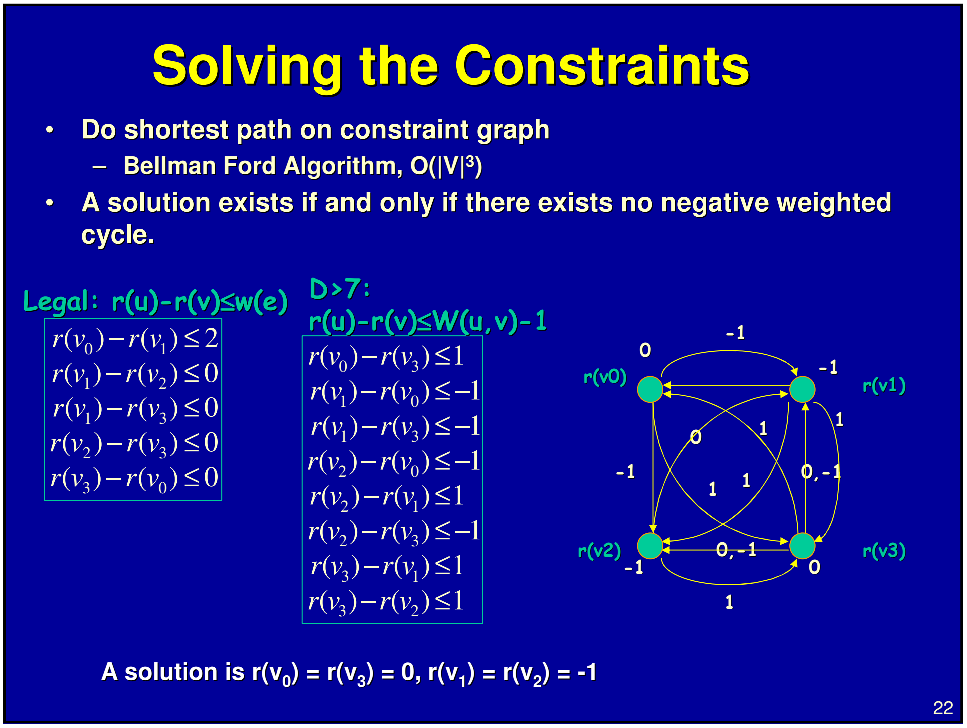
V0	V1	V2	V3	
0	3	6	13	V0
13	3	6	13	V1
10	13	3	10	V2
7	10	13	7	V3

Legal: r(u)-r(v)≤w(e)

$$\begin{aligned} r(v_0)-r(v_1) &\le 2 \\ r(v_1)-r(v_2) &\le 0 \\ r(v_1)-r(v_3) &\le 0 \\ r(v_2)-r(v_3) &\le 0 \\ r(v_3)-r(v_0) &\le 0 \end{aligned}$$

D>7: r(u)-r(v)≤W(u,v)-1

$$\begin{aligned} r(v_0)-r(v_3) &\le 1 \\ r(v_1)-r(v_0) &\le -1 \\ r(v_1)-r(v_3) &\le -1 \\ r(v_2)-r(v_0) &\le -1 \\ r(v_2)-r(v_1) &\le 1 \\ r(v_2)-r(v_3) &\le -1 \\ r(v_3)-r(v_1) &\le 1 \\ r(v_3)-r(v_2) &\le 1 \end{aligned}$$



Solving the Constraints

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

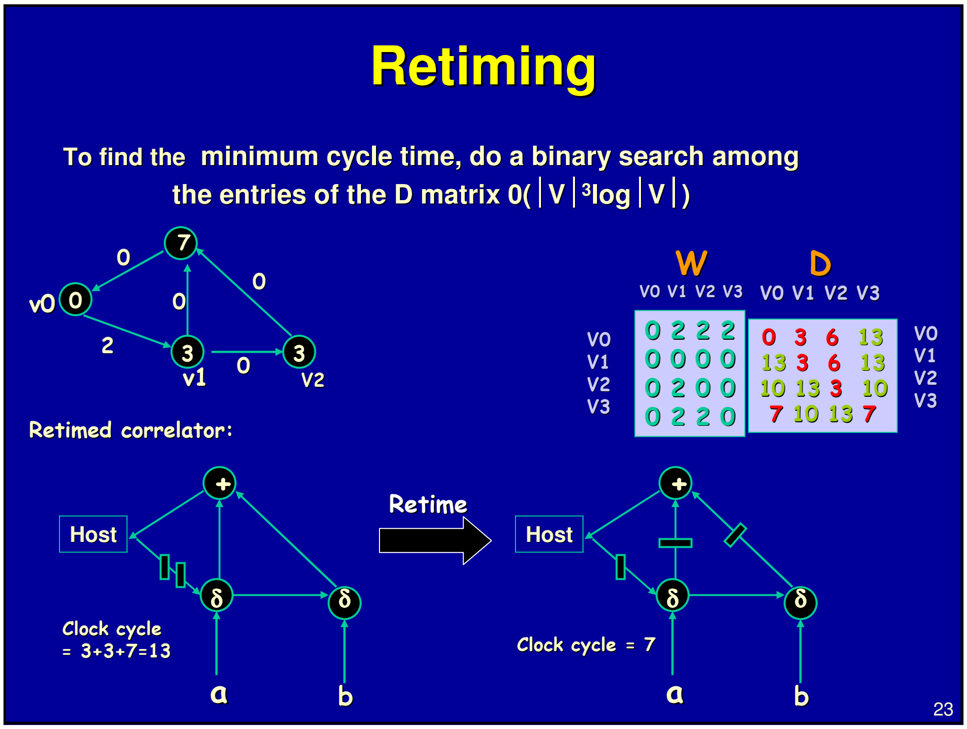
Legal: r(u)-r(v)≤w(e)

$$\begin{aligned} r(v_0)-r(v_1) &\le 2 \\ r(v_1)-r(v_2) &\le 0 \\ r(v_1)-r(v_3) &\le 0 \\ r(v_2)-r(v_3) &\le 0 \\ r(v_3)-r(v_0) &\le 0 \end{aligned}$$

D>7: r(u)-r(v)≤W(u,v)-1

$$\begin{aligned} r(v_0)-r(v_3) &\le 1 \\ r(v_1)-r(v_0) &\le -1 \\ r(v_1)-r(v_3) &\le -1 \\ r(v_2)-r(v_0) &\le -1 \\ r(v_2)-r(v_1) &\le 1 \\ r(v_2)-r(v_3) &\le -1 \\ r(v_3)-r(v_1) &\le 1 \\ r(v_3)-r(v_2) &\le 1 \end{aligned}$$

A solution is r(v$_0$) = r(v$_3$) = 0, r(v$_1$) = r(v$_2$) = -1



Retiming

To find the minimum cycle time, do a binary search among the entries of the D matrix $0(|V|^3\log|V|)$

W

	V0	V1	V2	V3
V0	0	2	2	2
V1	0	0	0	0
V2	0	2	0	0
V3	0	2	2	0

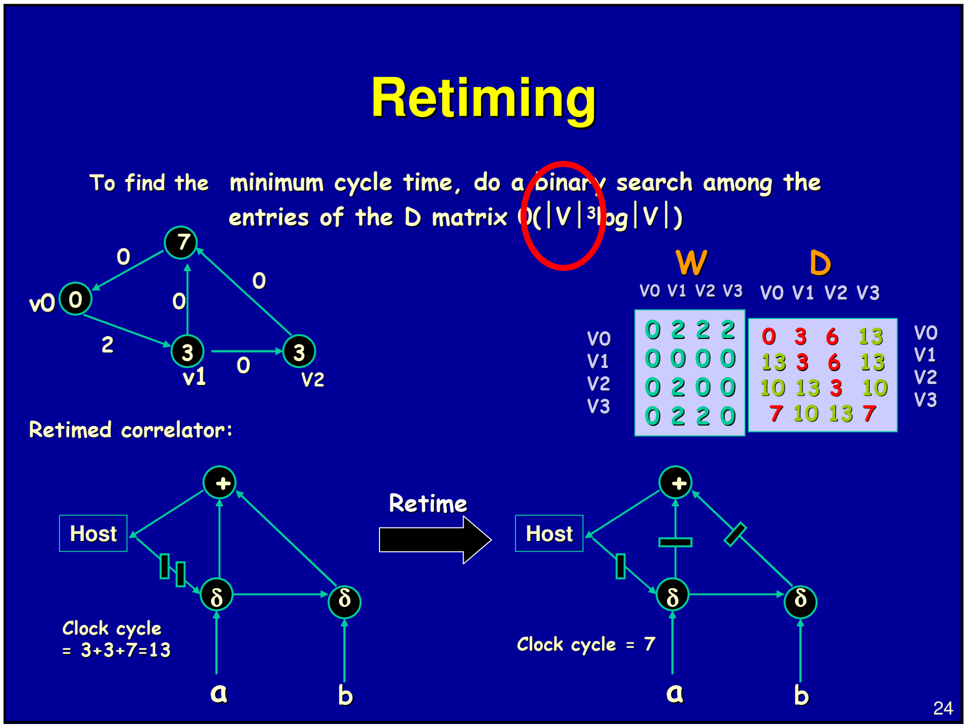
D

V0	V1	V2	V3	
0	3	6	13	V0
13	3	6	13	V1
10	13	3	10	V2
7	10	13	7	V3

Retimed correlator:

Clock cycle = 3+3+7=13

Clock cycle = 7



Retiming

To find the minimum cycle time, do a binary search among the entries of the D matrix $0(|V|^3\log|V|)$

W

	V0	V1	V2	V3
V0	0	2	2	2
V1	0	0	0	0
V2	0	2	0	0
V3	0	2	2	0

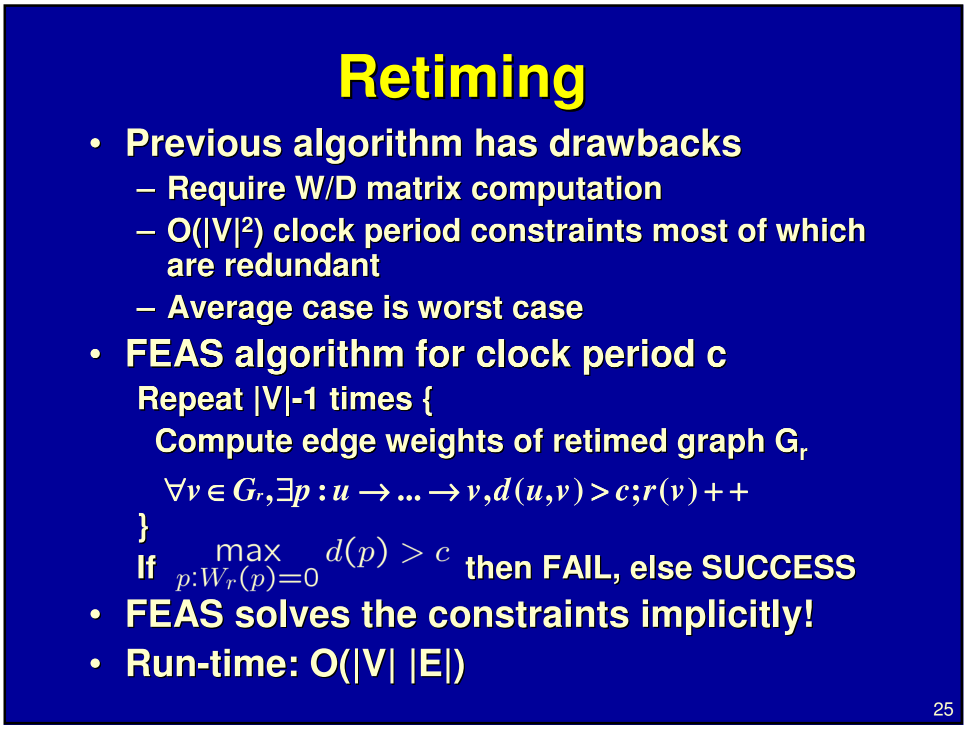
D

V0	V1	V2	V3	
0	3	6	13	V0
13	3	6	13	V1
10	13	3	10	V2
7	10	13	7	V3

Retimed correlator:

Clock cycle = 3+3+7=13

Clock cycle = 7

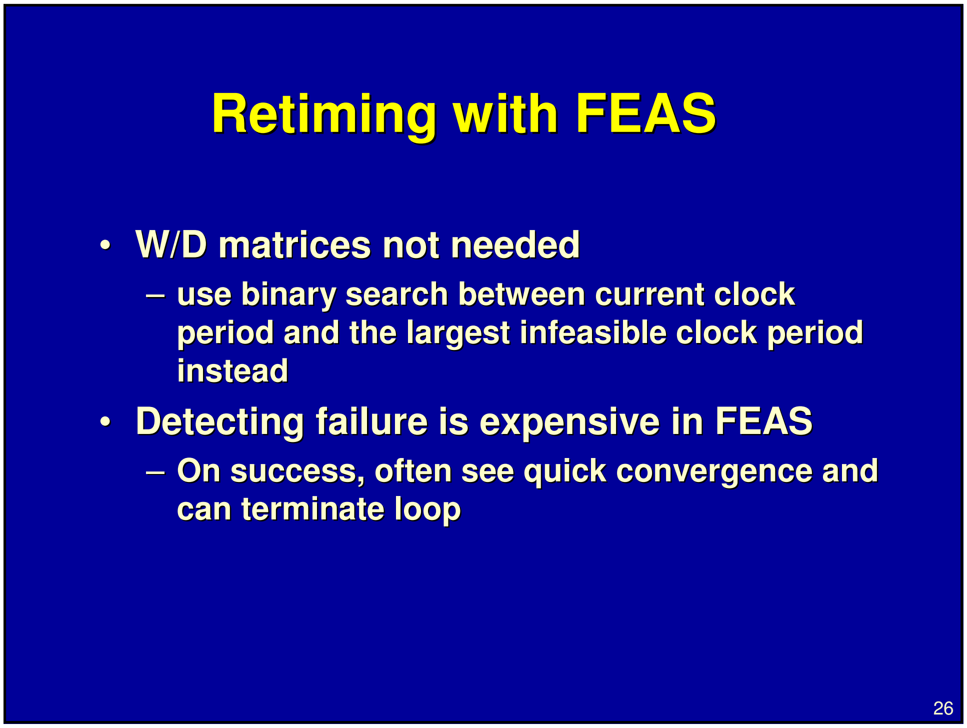


Retiming

- Previous algorithm has drawbacks
  - Require W/D matrix computation
  - O(|V|^2) clock period constraints most of which are redundant
  - Average case is worst case
- FEAS algorithm for clock period c

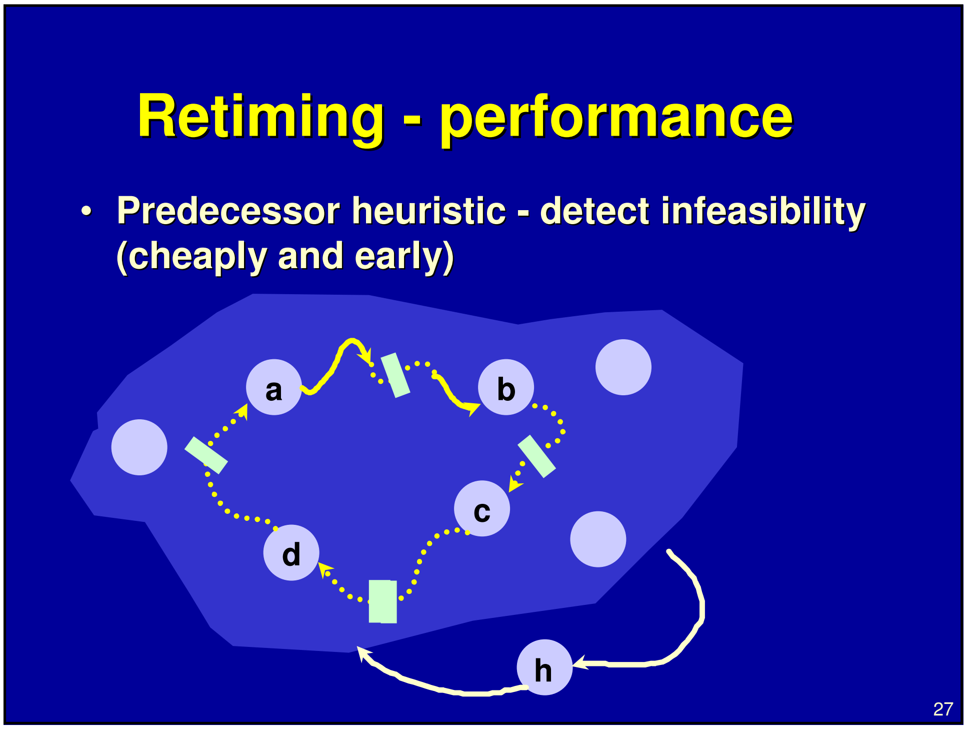
```
Repeat |V|-1 times {
  Compute edge weights of retimed graph $G_r$
   $\forall v \in G_r, \exists p : u \rightarrow \ldots \rightarrow v, d(u,v) > c; r(v)++$
}
If $\max_{p:W_r(p)=0} d(p) > c$ then FAIL, else SUCCESS
```

- FEAS solves the constraints implicitly!
- Run-time: O(|V| |E|)



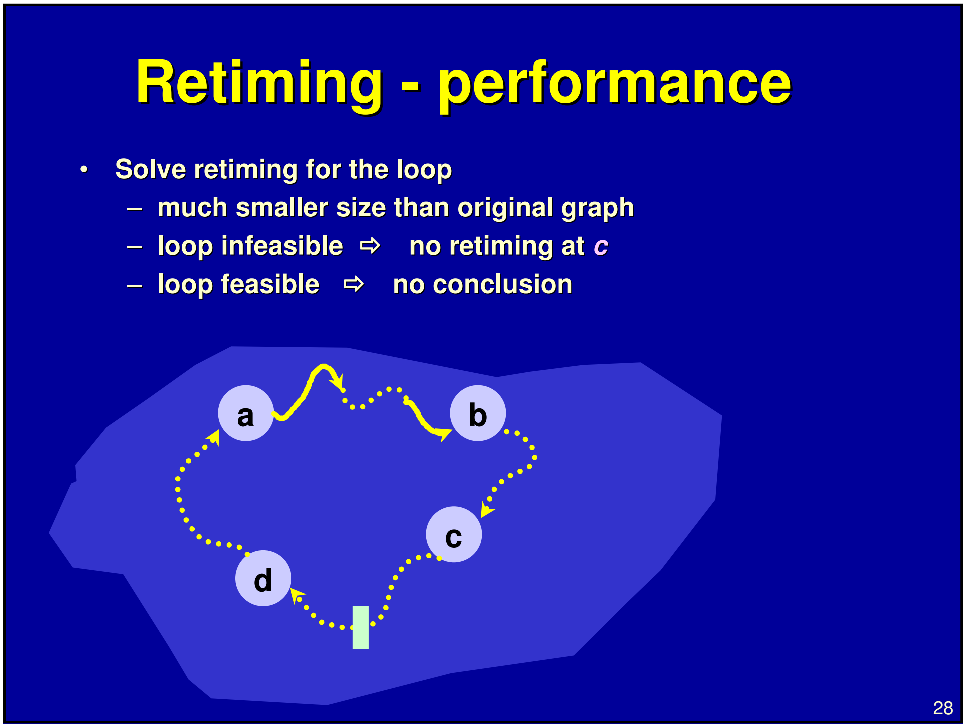
Retiming with FEAS

- W/D matrices not needed
  - use binary search between current clock period and the largest infeasible clock period instead
- Detecting failure is expensive in FEAS
  - On success, often see quick convergence and can terminate loop



Retiming - performance

- Predecessor heuristic - detect infeasibility (cheaply and early)



Retiming - performance

- Solve retiming for the loop
  - much smaller size than original graph
  - loop infeasible ➪ no retiming at *c*
  - loop feasible ➪ no conclusion

Retiming For Minimum Area at Fixed Clock Period (“Constrained Min Area”)

Goal: minimize number of registers used

$$
\begin{aligned}
\min N_r &= \sum_{e \in E} w_r(e) \\
&= \sum_{e:u \to v} (w(e) + r(v) - r(u)) \\
&= \sum_{e \in E} w(e) + \sum_{e:u \to v} (r(v) - r(u)) \\
&= N + \sum_{u \to v} (r(v) - r(u)) \\
&= N + \sum_{v \in V} \left[r(v)(\#\, fanin(v) - \#\, fanout(v) \right] \\
&= N + \sum_{v \in V} a_v r(v)
\end{aligned}
$$

where a_v is a constant.

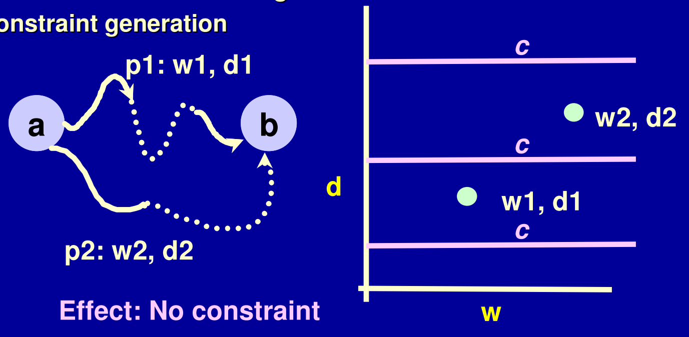
Other constraints same as before

Solved by solving the dual linear program

- A minimum cost circulation problem

Retiming - performance

- Constrained min-area retiming
  - constraint generation



Effect: No constraint

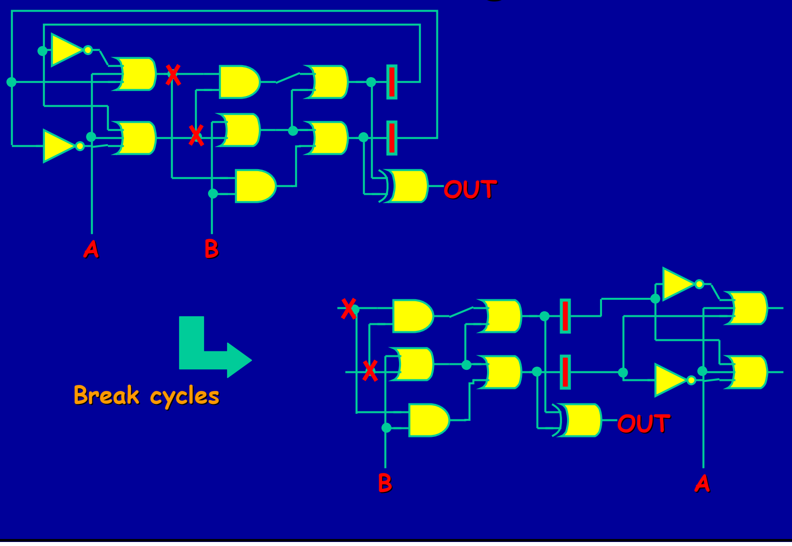
Effect: No constraint

Effect: w_r (p1) > 1

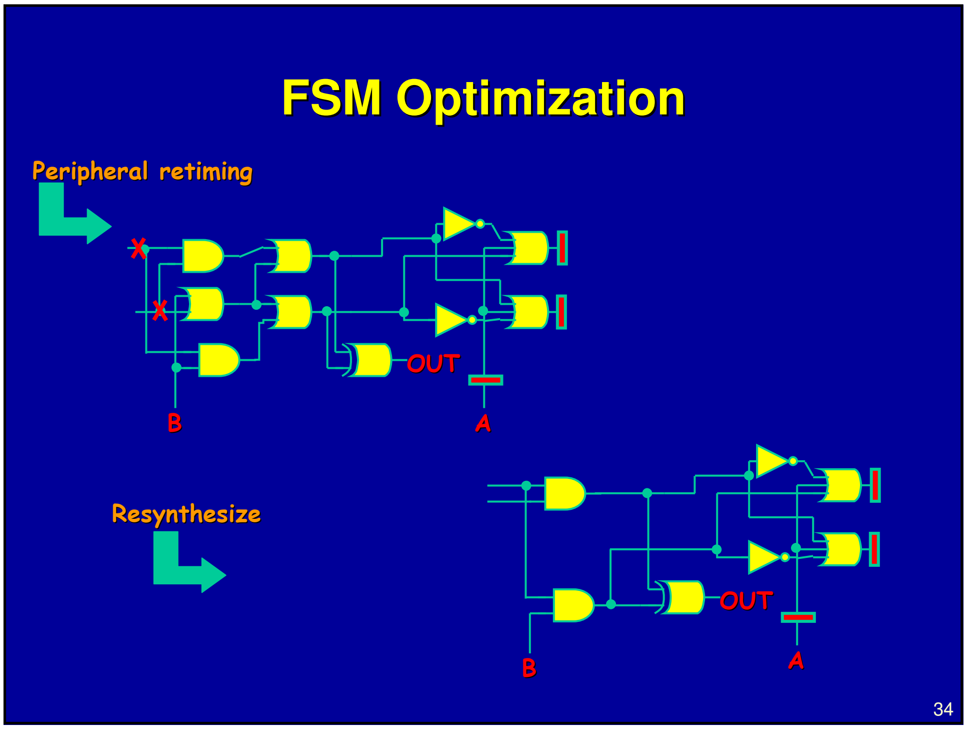
Retiming For Minimum Area

- In practice
  - We need W & D matrices to add clock period edges
    - Compute row of matrix at a time and avoid redundant edges
  - Use minimum cost scaling to solve circulation problem
    - Numeric precision needs big integers!

FSM Optimization: Combining Combinational Optimization and Retiming



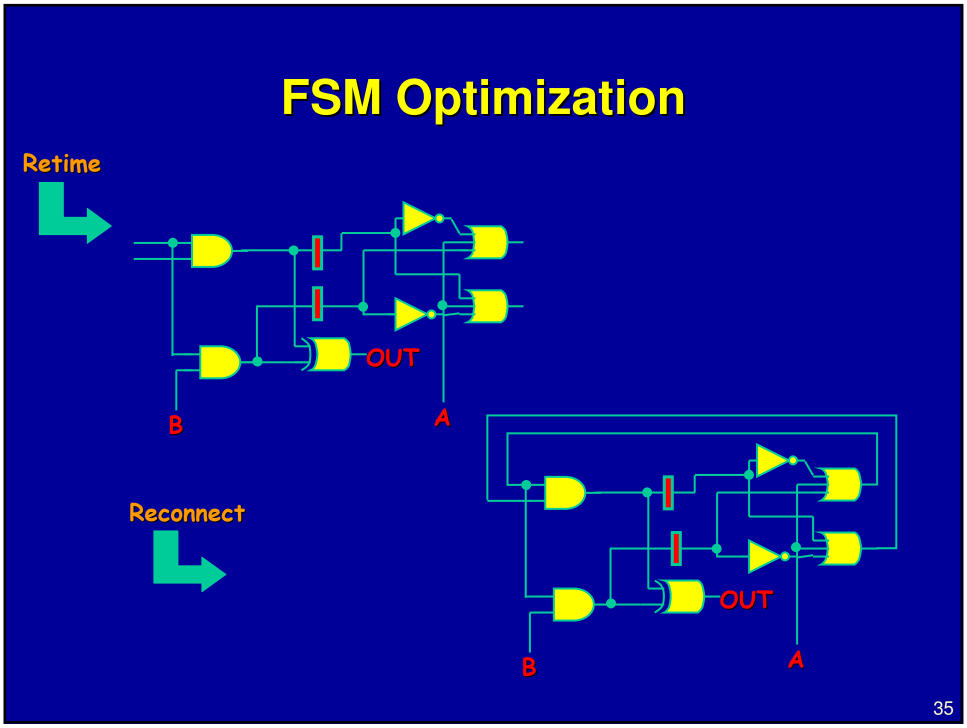
Break cycles



FSM Optimization

Peripheral retiming

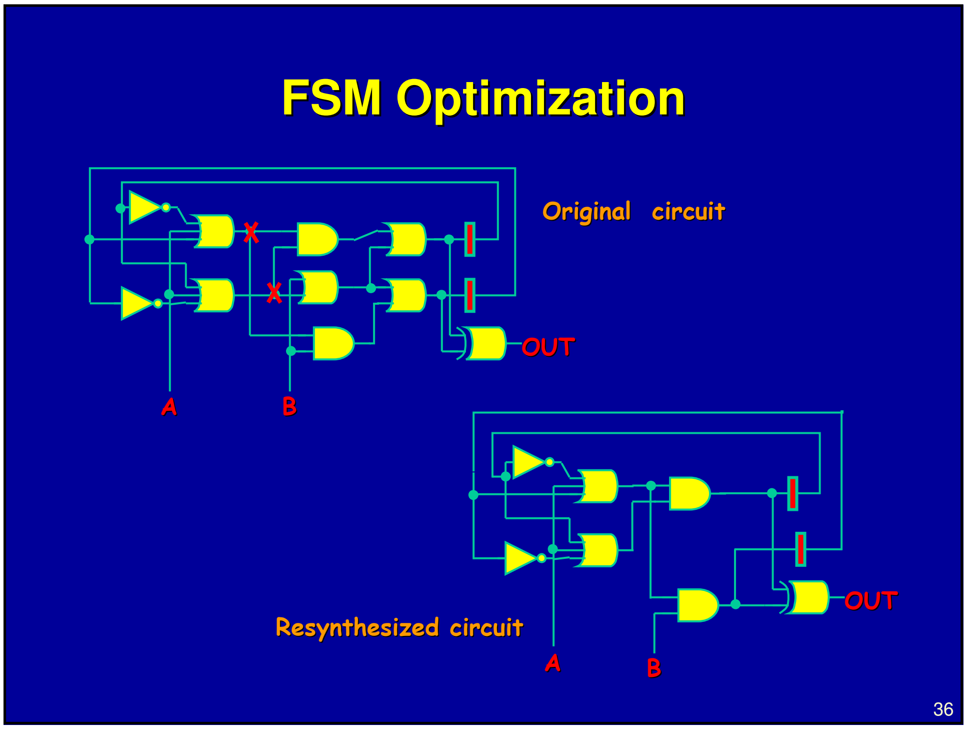
Resynthesize



FSM Optimization

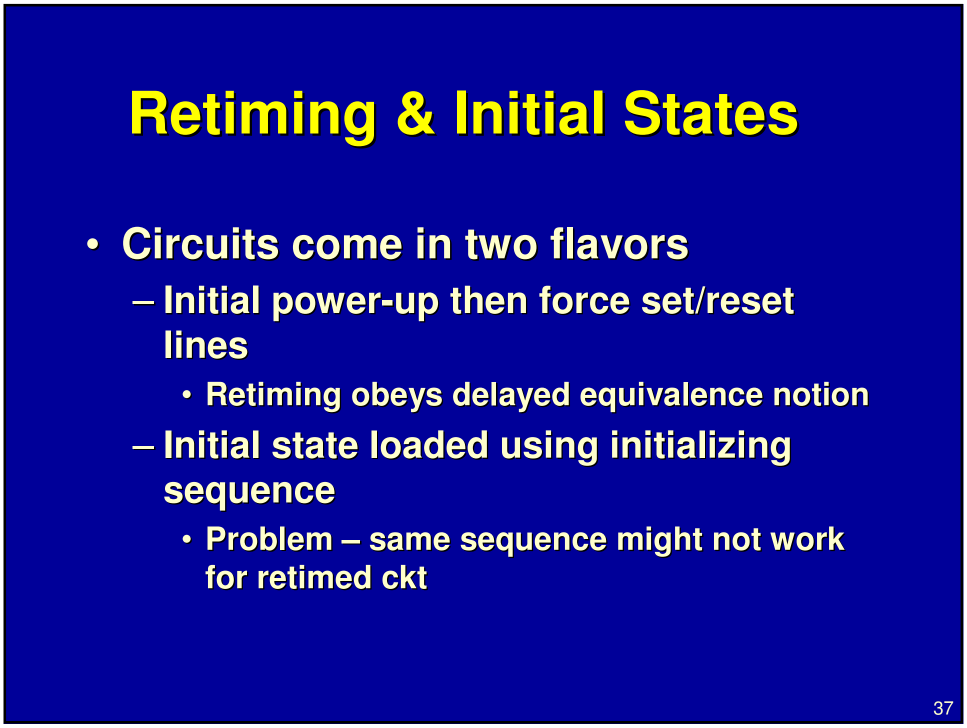
Retime

Reconnect



FSM Optimization

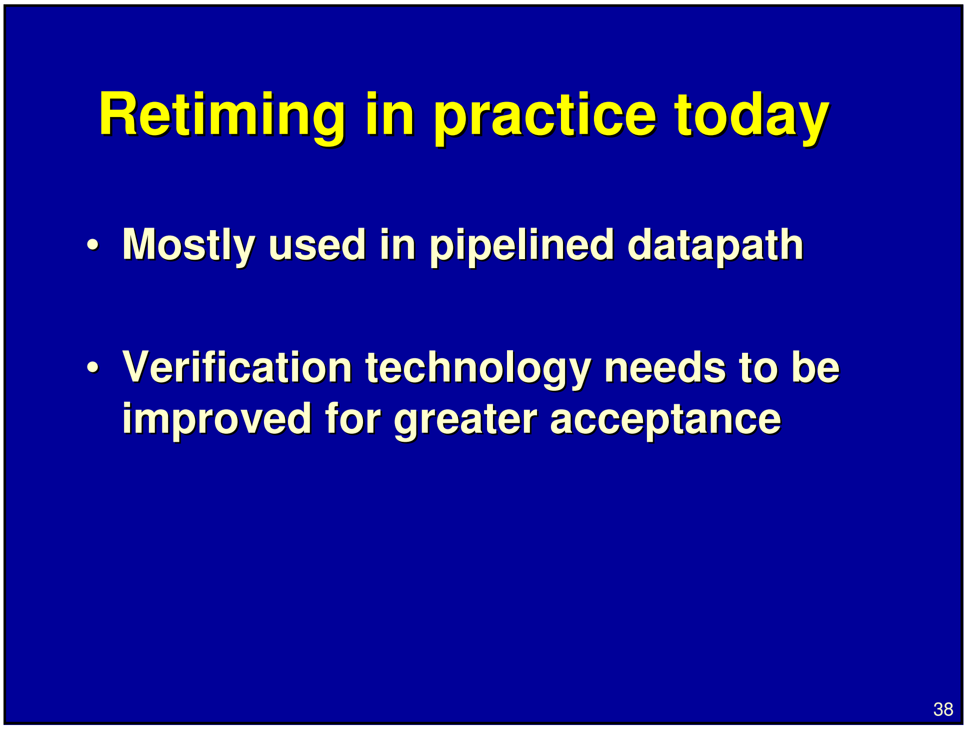
36



Retiming & Initial States

- Circuits come in two flavors
  - Initial power-up then force set/reset lines
    - Retiming obeys delayed equivalence notion
  - Initial state loaded using initializing sequence
    - Problem – same sequence might not work for retimed ckt

37



Retiming in practice today

- Mostly used in pipelined datapath
- Verification technology needs to be improved for greater acceptance

38