Optimizing Sequential Circuits by Retiming Netlist of Gates

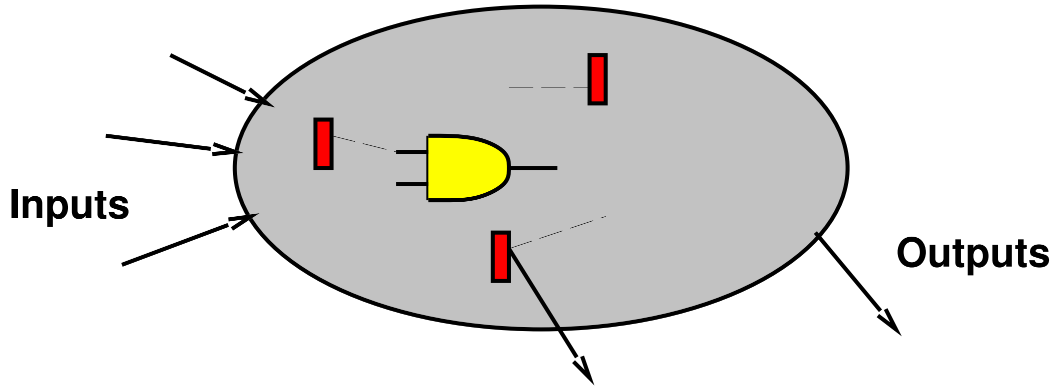
Outline:

1. Retiming

2. Retiming and Resynthesis (RnR)

3. Resynthesis of Pipelines

Netlist of gates and registers:



Various Goals:

1) Reduce clock cycle time
2) Reduce area
3) Reduce number of latches

Retiming

Problem:

- Pure combinational optimization can be myopic since relations across register boundaries are disregarded

Solutions:

- Retiming: Move registers so that
  i) clock cycle decreases, or
  number of registers decreases and
  ii) input-output behavior is preserved

- RnR: Combine retiming with combinational optimization techniques so that can optimize across boundaries

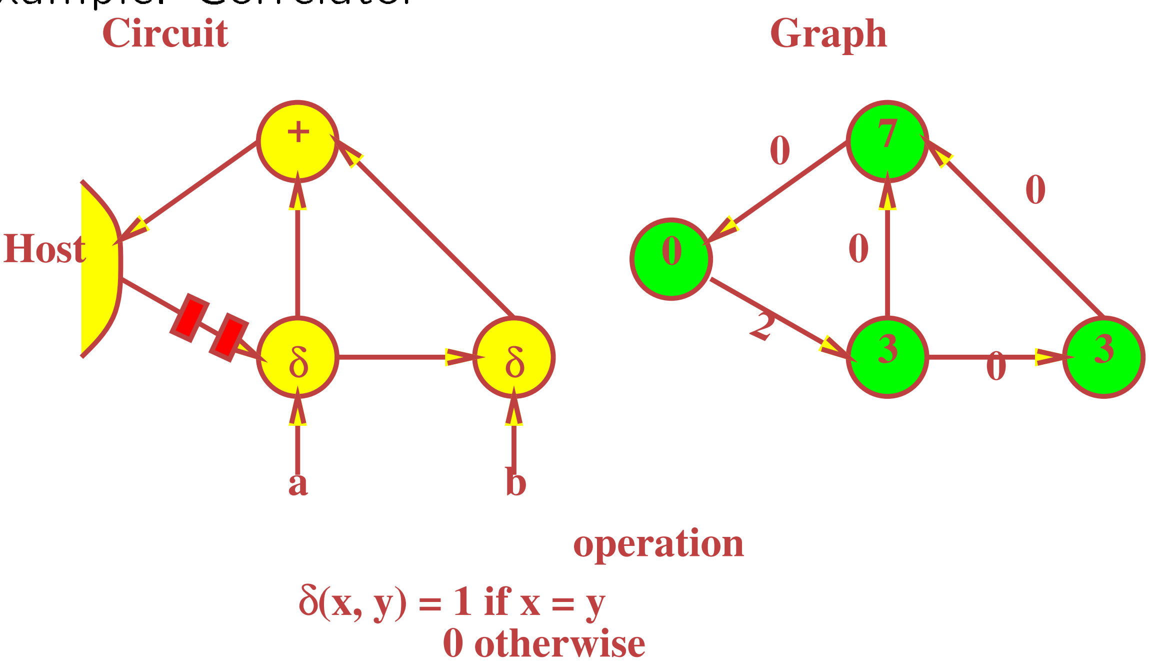
Retiming: circuit representation:

- Leiserson, Rose and Saxe (1983)

- Circuit representation: $G(V, E, d, w)$
  $V \leftrightarrow$ set of gates
  $E \leftrightarrow$ set of wires
  $d(v)$ = delay of gate/vertex v, ($d(v) \geq 0$)
  $w(e)$ = number of registers on edge e,
  ($w(e) \geq 0$)

Retiming: circuit representation

Example: Correlator



Operation	delay
δ	3
+	7

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

Retiming: preliminaries

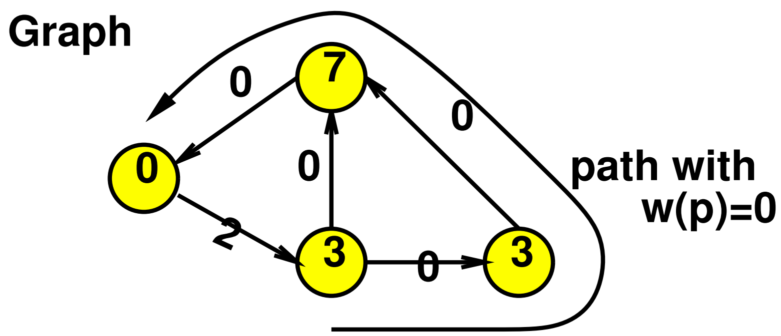
- For a path $p : v_0 \overset{e_0}{\rightarrow} v_1 \ldots \overset{e_{k-1}}{\rightarrow} v_k$

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

- Clock cycle

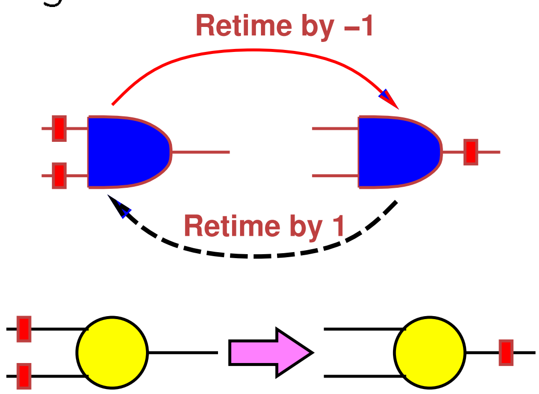
$$c = \max_{p:w(p)=0} \{d(p)\}$$

For correlator $c = 13$



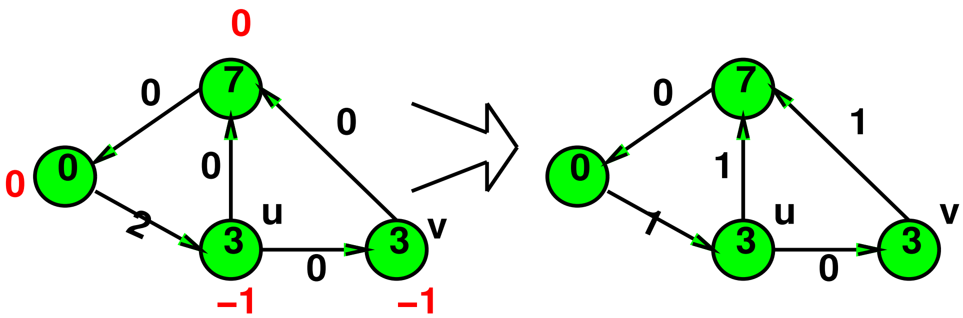
Retiming: Basic Operation

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



- Does not affect gate functionalities
- A mathematical definition: **retardation**
  $r : V \to Z$, an integer vertex labeling
  $w_r(e) = w(e) + r(v) - r(u)$ for edge $e = (u, v)$

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



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

A retiming r is **legal** if $w_r(e) \geq 0$, $\forall e \in E$

Retiming for minimum clock cycle

Problem statement: (Minimum cycle time)
Given $G(V, E, d, w)$, find a **legal** retiming r so that

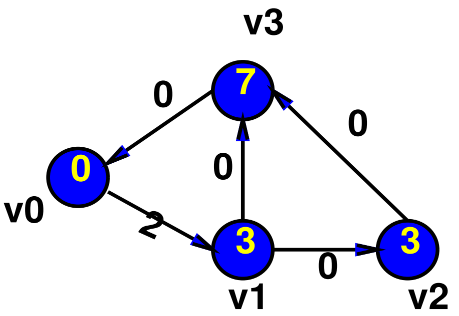
$$c = \max_{p: w_r(p)=0} \{d(p)\}$$

is minimized.

Retiming: 2 important matrices

- Register weight matrix
  $W(u, v) = \min\{w(p) : u \overset{p}{\rightsquigarrow} v\}$

- Delay matrix
  $D(u, v) = \max\{d(p) : u \overset{p}{\rightsquigarrow} v, w(p) = W(u, v)\}$



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

$$c \le \alpha \Leftrightarrow \forall p,\ \text{if } d(p) > \alpha \text{ then } w(p) \ge 1$$

Conditions for Retiming

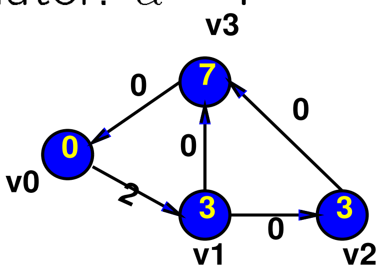
- Assume that we are asked to check if a retiming exists for a clock cycle α
- *Legal retiming*: $w_r(e) \geq 0$ for all e. Hence $w_r(e) = w(e) + r(v) - r(u) \geq 0$ or
$$\boxed{r(u) - r(v) \leq w(e)}$$
- For all paths $p : u \rightsquigarrow v$ such that $d(p) > \alpha$, we require $w_r(p) \geq 1$
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 taking the least $w(p)$ (tightest constraint)
$$\boxed{r(u) - r(v) \leq W(u,v) - 1}$$

Note: this is independent of the path from u to v, so we just need to apply it to u, v such that $D(u,v) > \alpha$.

Solving the constraints

- All constraints in difference of 2 variable form
- Related to shortest path problem

Correlator: $\alpha = 7$



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

$D \geq 7$:
$r(u) - r(v) \leq W(u,v) - 1$

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

Legal: $r(u) - r(v) \leq w(e)$

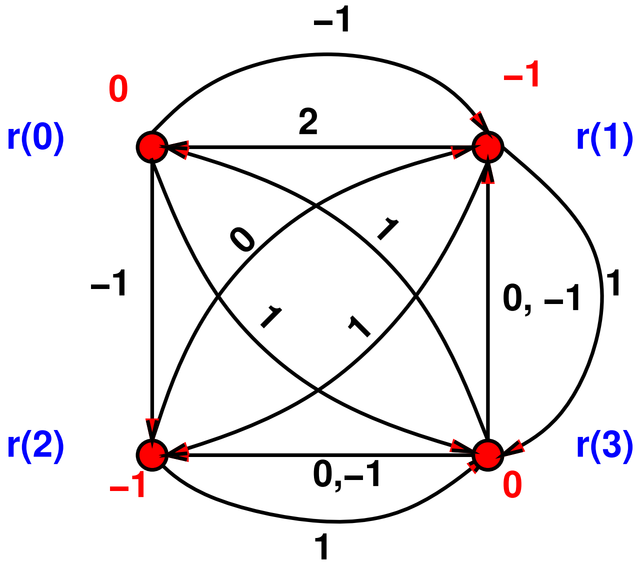
$$\begin{aligned} r(v_0) - r(v_1) &\leq 2 \\ r(v_1) - r(v_2) &\leq 0 \\ r(v_1) - r(v_3) &\leq 0 \\ r(v_2) - r(v_3) &\leq 0 \\ r(v_3) - r(v_0) &\leq 0 \end{aligned}$$

Solving the constraints

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

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

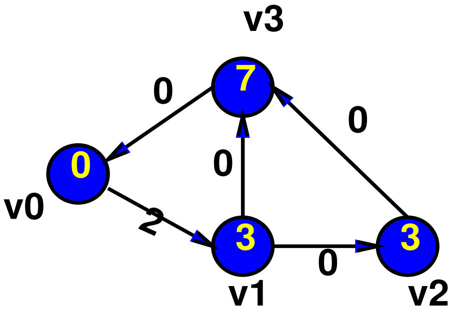
Do shortest path on constraint graph- ($O(|V|^3)$. A solution exists if and only if there exists no negative weighted cycle.



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 $O(|V|^3 \log |V|)$



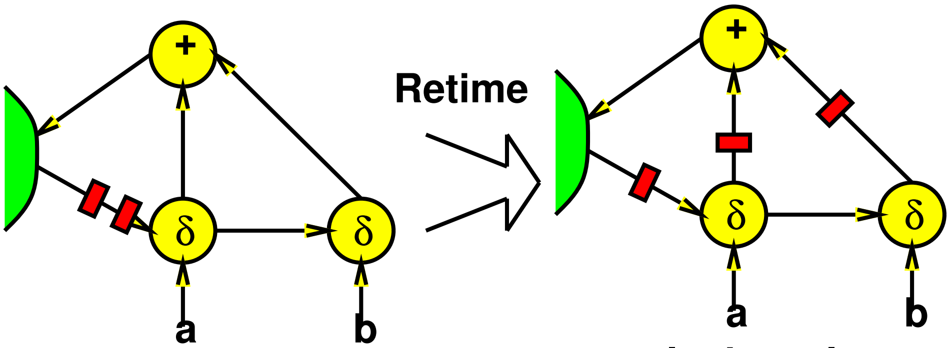
W

	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:

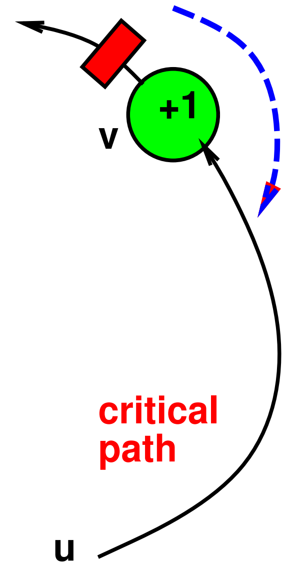


clock cycle = 3 + 3 + 7 = 13

clock cycle = 7

Retiming : 2 more algorithms

1. Relaxation based: Repeatedly find *critical path*; retime vertex at end of path by $+1$
   $O(|V||E|\log|V|)$



2. Mixed Integer Linear Program formulation

Retiming for minimum state

Goal: minimize number of registers used

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

where a_v is a constant.

Minimum registers - formulation

- minimize: $\sum_{v \in V} a_v r(v)$

  subject to

  $w_r(e) = w(e) + r(v) - r(u) \geq 0$

- Reducible to a flow problem

- Care to be taken when registers fanout to registers

- Handle multi-way forks with care

Retiming and Resynthesis: motivation

- Goal: incorporate combinational optimization into sequential optimization

- Naive approach: carve out combinational regions, do optimization on each region
  Only local gains made.

- Can we do any better?

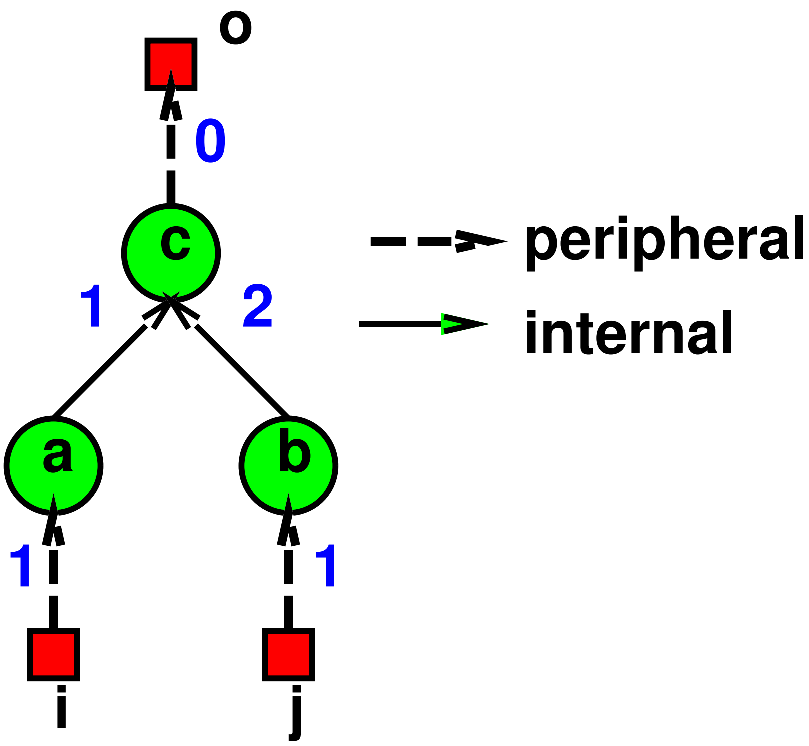
RnR: a new approach

- Sentovich, Malik, Brayton and Sangiovanni-Vincentelli (89)

- 3 step approach
  1: Move registers to boundary of circuit
  2: Optimize network
  3: Move registers back in an optimal way

RnR: circuit representation

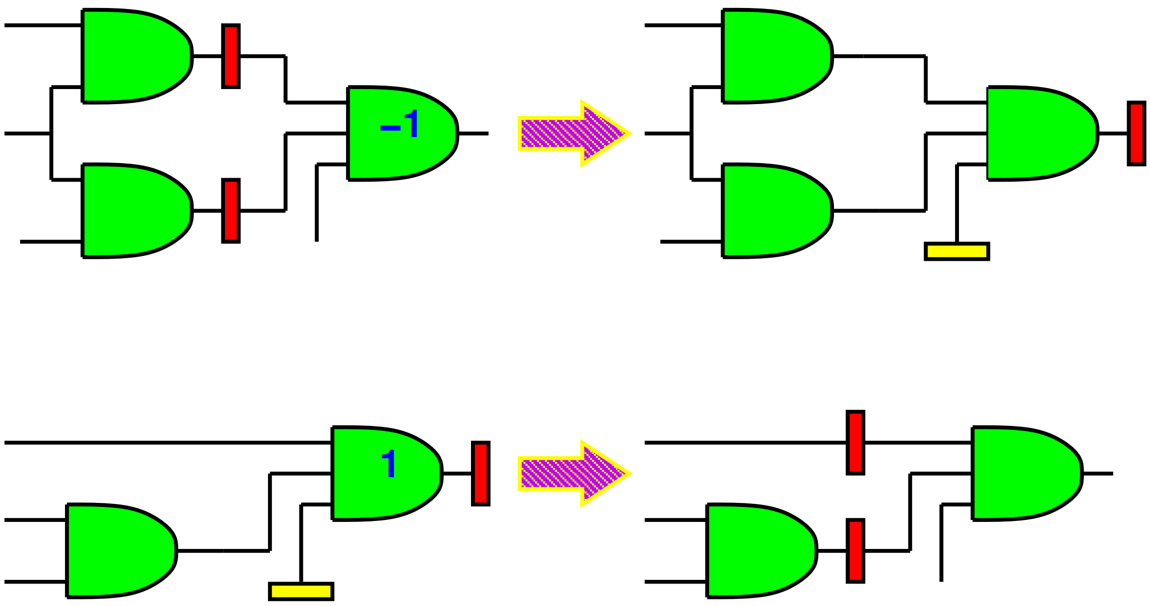
- Circuit representation: communication graph



internal/peripheral edges
edge-weight = register count

Extended Retiming

- Move registers to the periphery
- Negative edge-weights permitted

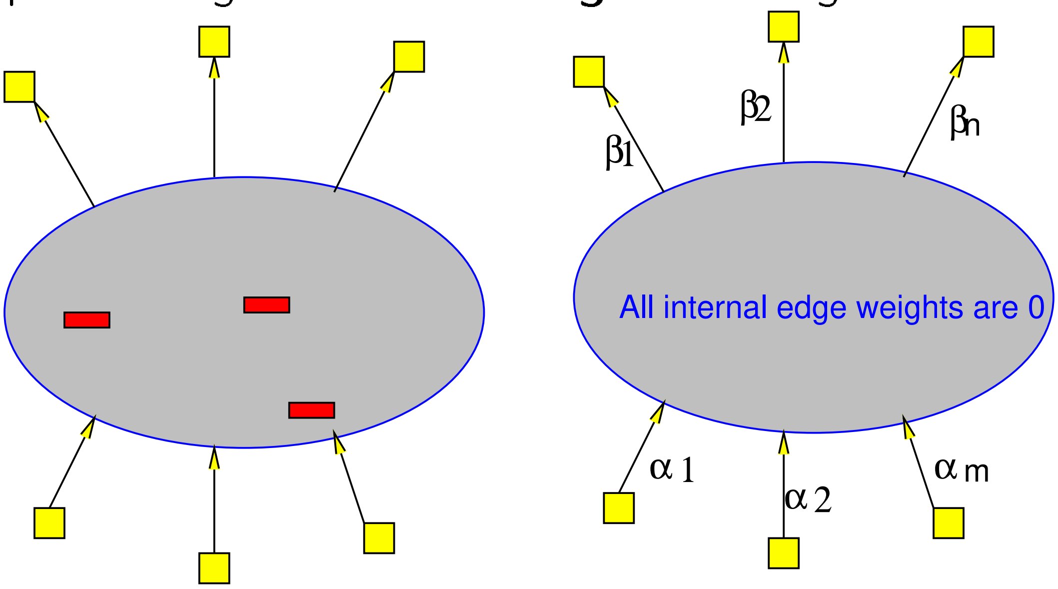


- A negative latch has the interpretation that it **advances** its input by 1 clock cycle instead of delaying it by one clock cycle.

Peripheral Retiming

A retiming is called a **peripheral retiming** if it results in all internal edges having zero weight

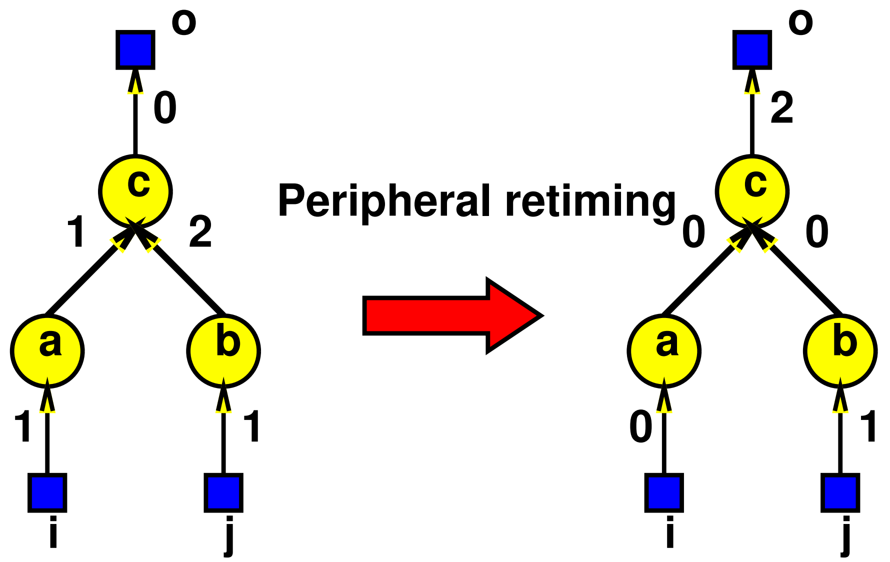
Peripheral edges can have **negative** weight



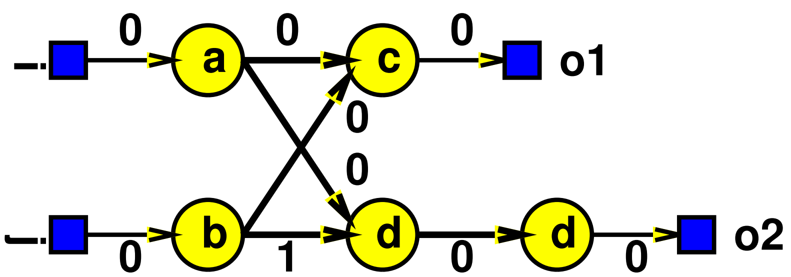
A circuit that undergoes peripheral retiming followed by a legal retiming, i.e. one that results in all weights ≥ 0 is **functionally equivalent** to the original circuit

Functional equivalence: equivalence of finite state automata (*but we have to be careful about the initial conditions and initializing sequences. The resulting circuit may only exhibit equivalence after an appropriate delay. See Singhal et.al ICCAD 1995. This holds even for regular retiming.*)

Peripheral Retiming - an example



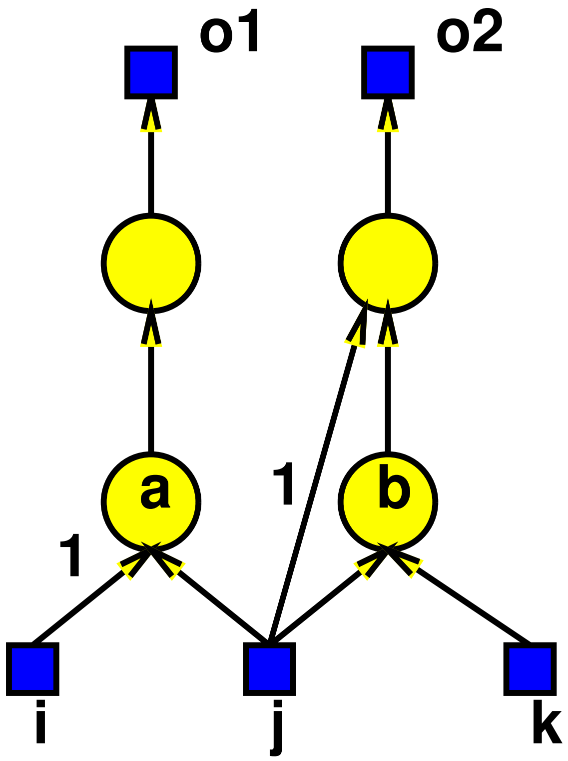
Not possible for all circuits:



Path Weight Matrix (PWM)

- Matrix W
  row - inputs, columns - outputs

$$W_{ij} = \begin{cases} * & \text{no path } i \leadsto j \\ \sum_{i \leadsto j} w(e) & \text{same weight for all paths} \\ \sim & \text{if paths with different weights} \end{cases}$$



	o1	o2
i	1	*
j	0	~
k	*	0

PWM and Peripheral Retiming

Satisfiable path weight matrix:

1. $W_{ij} \neq \sim, \qquad \forall i, \forall j$ and
2. $\exists \alpha_i, \beta_j,$ such that $W_{ij} = \alpha_i + \beta_j$, $\forall W_{i,j} \neq *$

- Peripheral retiming possible $\Leftrightarrow$ matrix is satisfiable

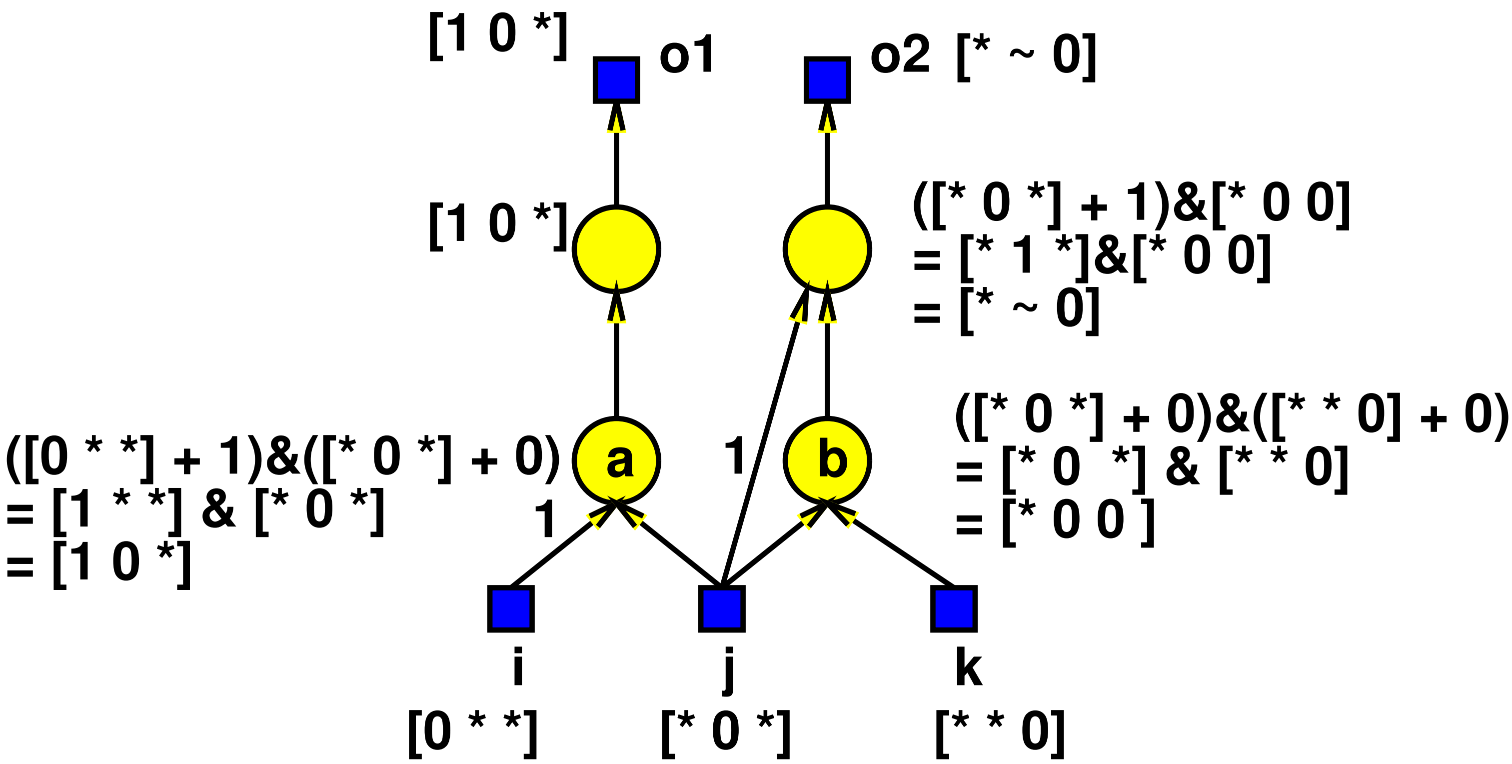
  α_i, β_j specify registers on peripheral edge
  some α_i, β_j can be negative

- Complexity: linear in size of communication graph

Path Weight Matrix - generation

- DFS skeleton - generate one output column of W at a time

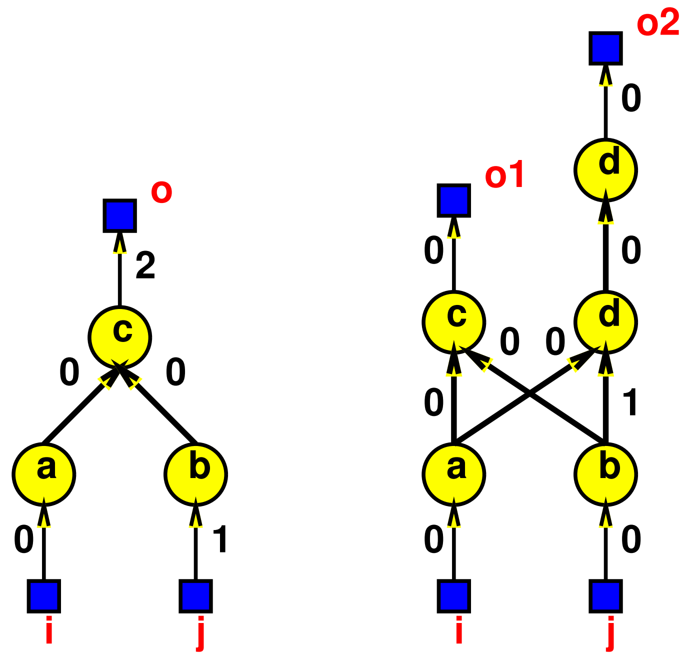
Example:



Computing α, β

Each constraint of the form $\alpha_i + \beta_j = W_{ij}$

- Procedure
  1. set $\alpha_1 = 0$
  2. use first row to generate β's
  3. determine α's from β's
  4. check for consistency



	o
i	2
j	3

$\alpha 1 = 0$
$\alpha 2 = 1$
$\beta 1 = 2$

	o1	o2
i	0	0
j	0	1

$\alpha 1 + \beta 1 = 0 \quad \alpha 2 + \beta 1 = 0$
$\alpha 1 + \beta 2 = 0 \quad \alpha 2 + \beta 2 = 1$

$\beta 1 - \beta 2 = 0 \quad \beta 1 - \beta 2 = -1$

Optimizing Acyclic Sequential Circuits

1) Acyclic circuits with **satisfiable** W

Do peripheral retiming putting α_i, β_j registers at the I/O

Resynthesize interior

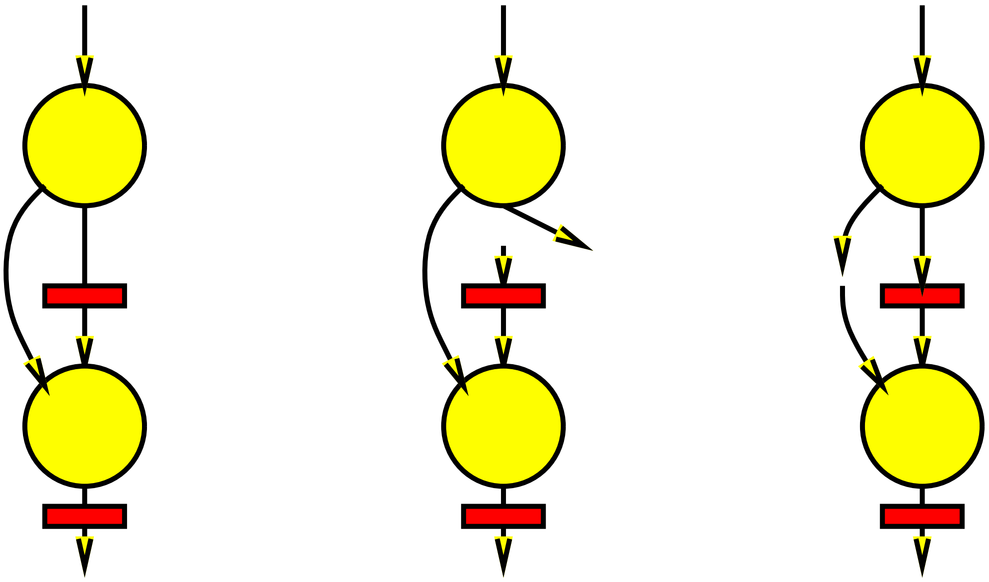
Do a legal retiming (move registers in)

May not always be possible.

2) Acyclic circuits with unsatisfiable W

Identify maximal sub-circuits with satisfiable W. Cut connections.

Repeat procedure 1

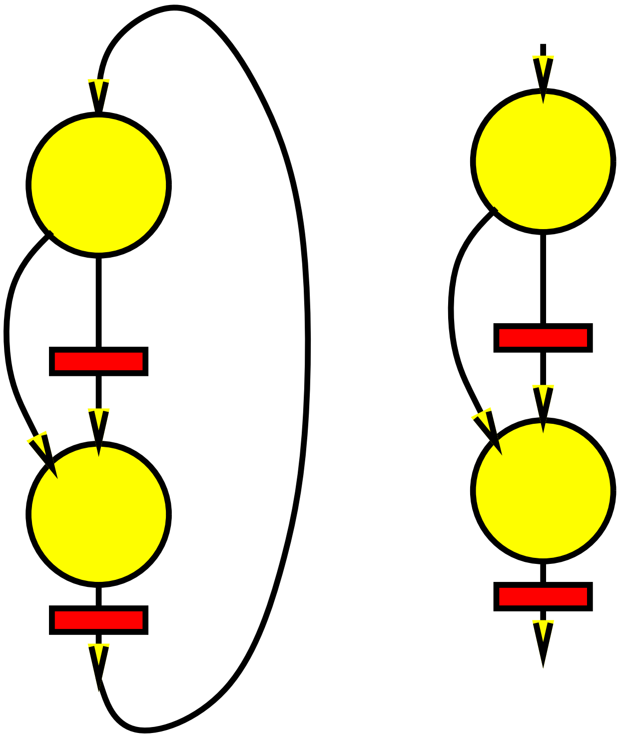


Note that both of the cut circuits can be peripherally retimed, whereas the first on cannot.

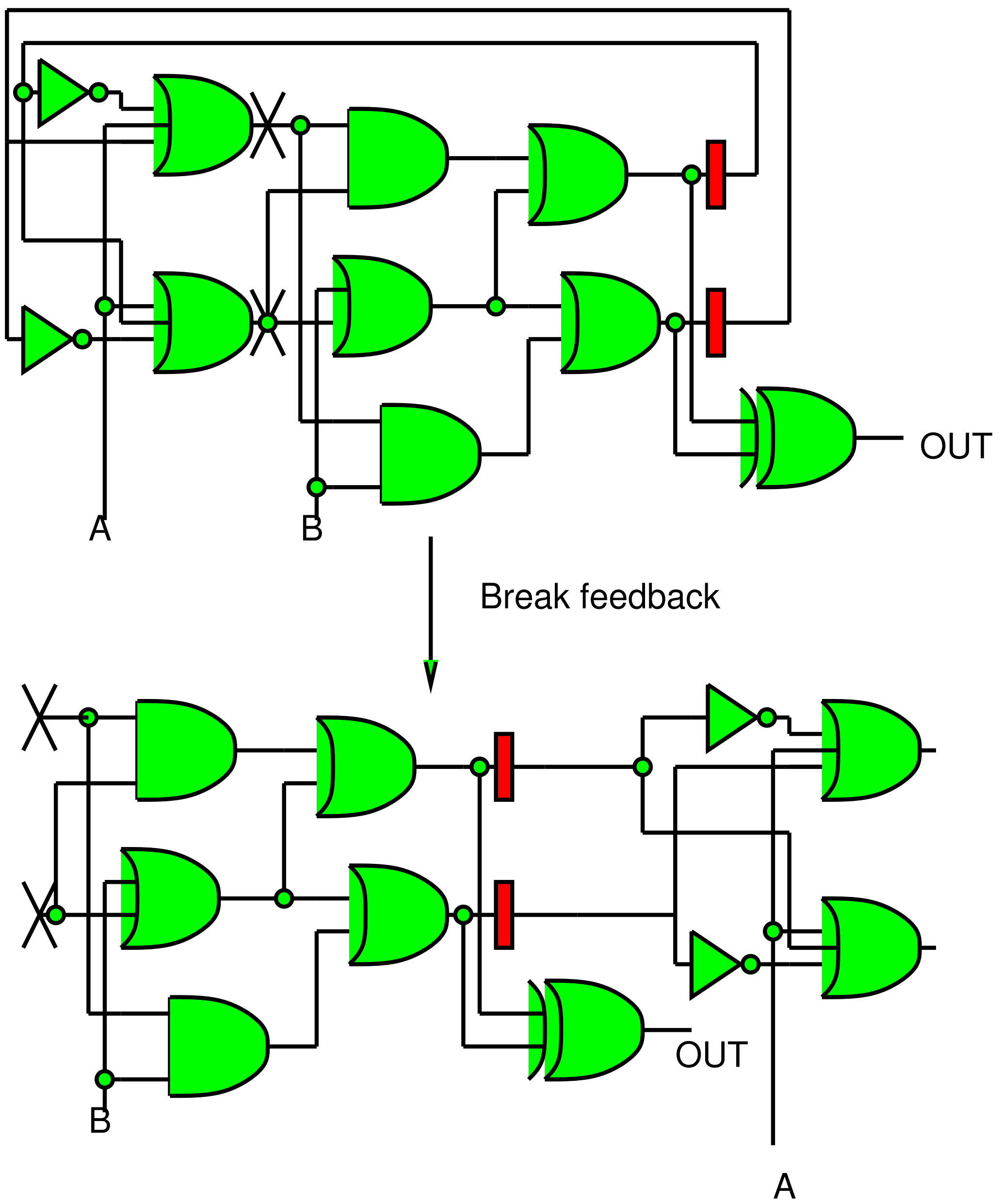
Optimizing Cyclic Sequential Circuits

Cyclic Circuits:

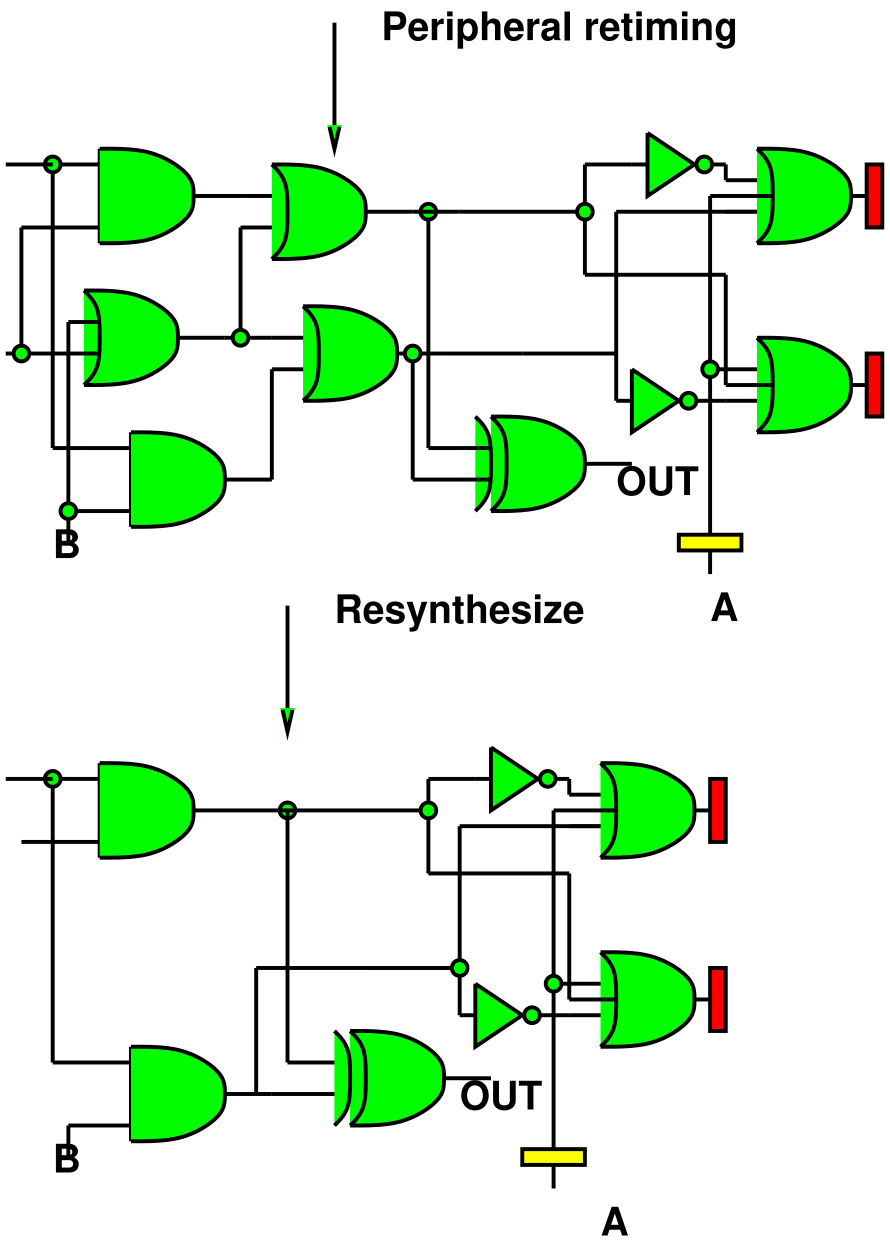
Make circuit acyclic by breaking cycles
Different feedback-cuts give different W's
Find sub-circuits with satisfiable W's
Repeat procedure 1



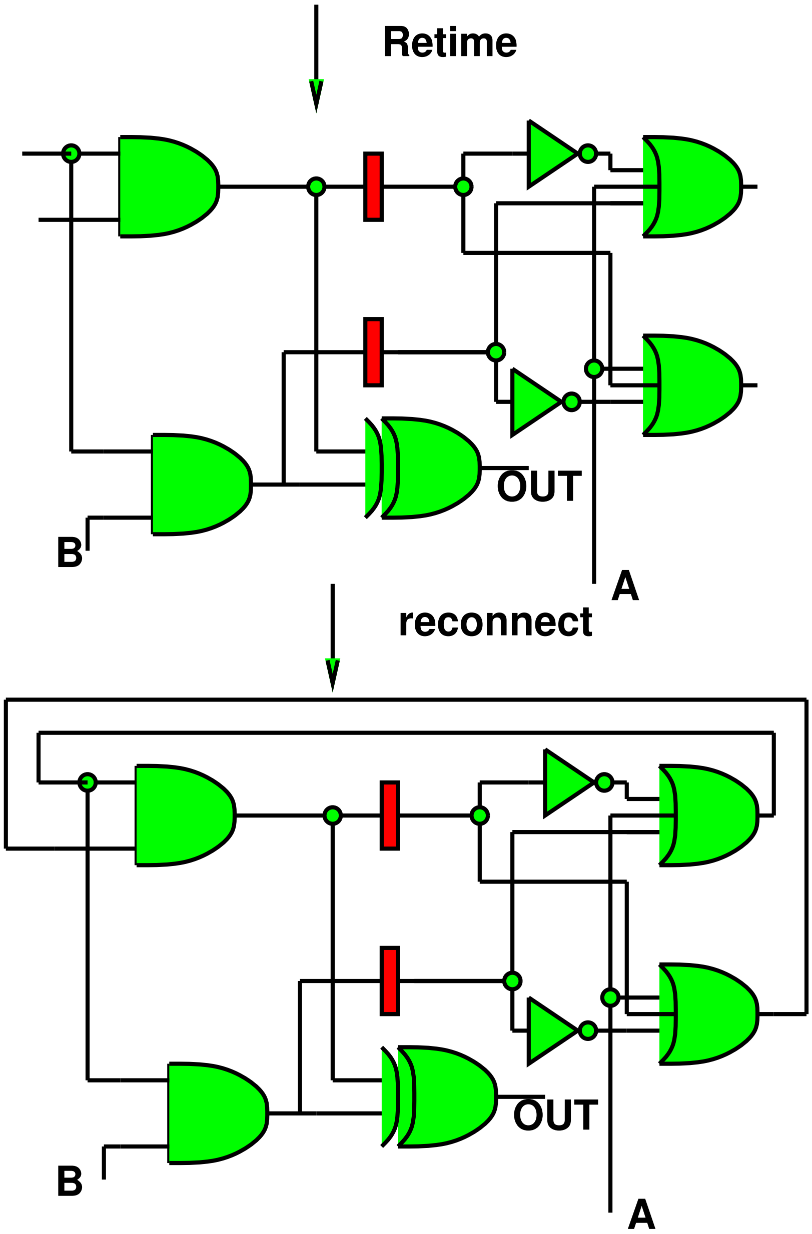
FSM Optimization



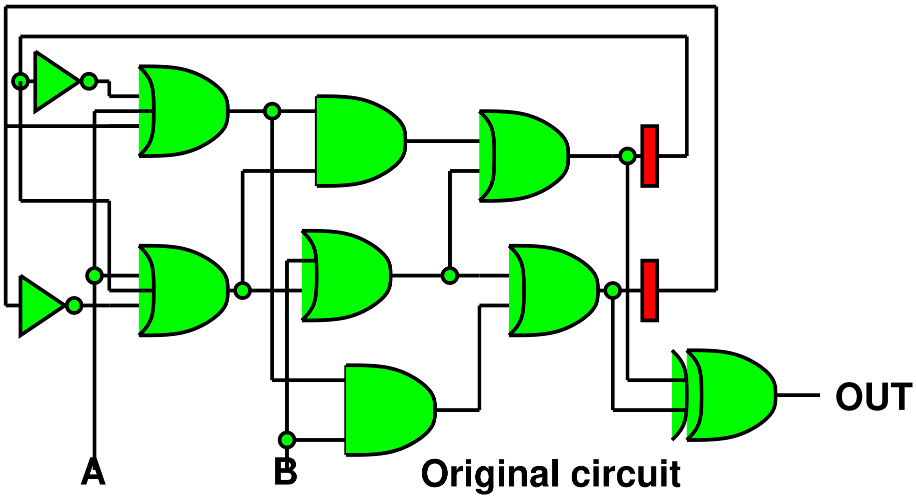
FSM Optimization



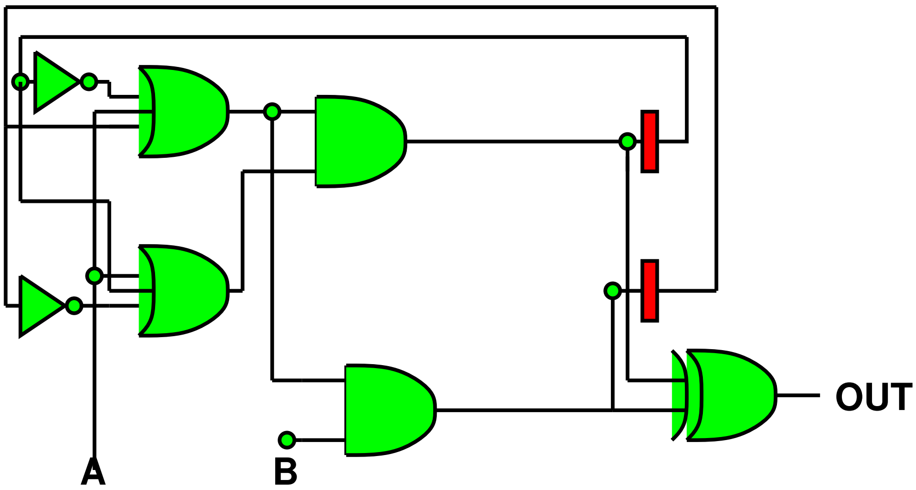
FSM Optimization



FSM Optimization



Original circuit

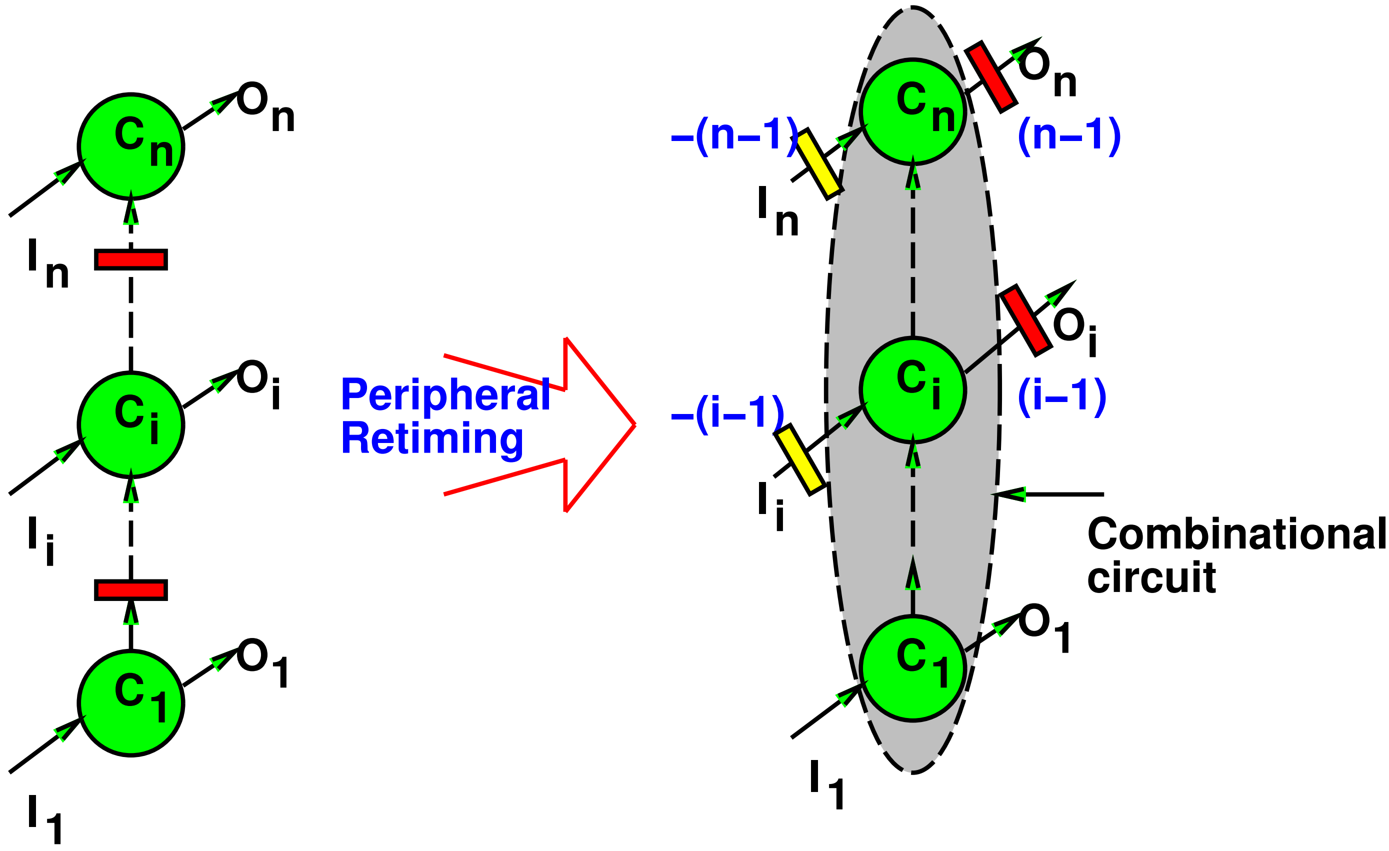


Resynthesized circuit

Resynthesis of Pipelines

Goal: Performance optimization of pipeline circuits

Example: Pipeline circuit $\mathcal{C}$



Resynthesis of Pipelines

Parameters:

A- vector of arrival times for inputs
R - vector of required times for outputs
c - target clock cycle
$\mathcal{C}$ - circuit

Pipeline performance problem: $\mathcal{P}_P(\mathcal{C}_P, c, A_P, R_P)$
Combinational performance problem: $\mathcal{P}_C(\mathcal{C}_C, A_C, R_C)$

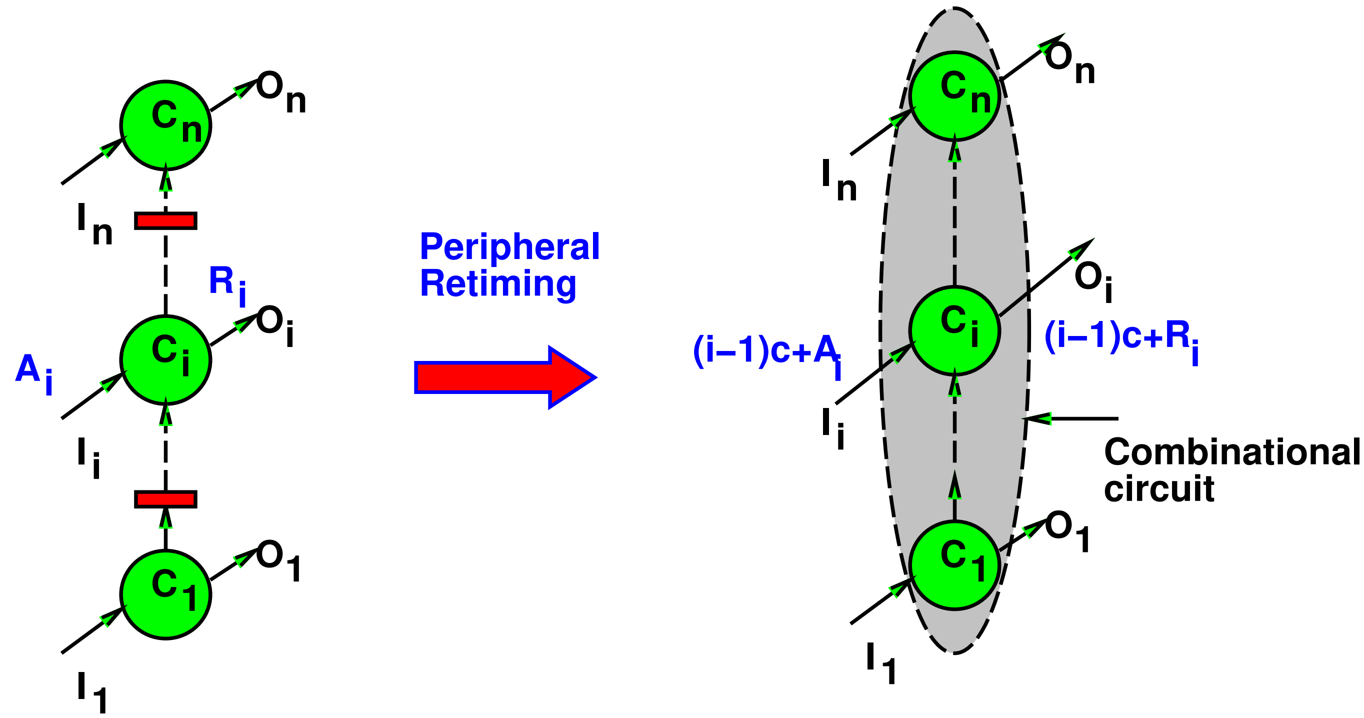
Problem Transformation:

- Relax $\mathcal{P}_P(\mathcal{C}_P, c, A_P, R_P)$ to $\mathcal{P}'_P(\mathcal{C}_P, c + \delta, A_P, R_P)$
  where $\delta =$ largest possible single gate delay
- Convert: pipeline $\mathcal{P}'_P$ to combinational problem $\mathcal{P}'_C$

$$\mathcal{C}_P \xrightarrow{\text{peripheral retime}} \mathcal{C}_C$$
$$R_C^{stage_i} = (i-1) \times c + R_P^{stage_i}$$
$$A_C^{stage_i} = (i-1) \times c + A_P^{stage_i}$$

Performance Synthesis of Pipelines

Arrival and Required Times for $\mathcal{P}_C^{'}$:



Theoretical Contribution:

- If $\mathcal{P}_C^{'}$ has a solution then retiming yields a solution to $\mathcal{P}_P^{'}$.
- If there is a solution to $\mathcal{P}_P$ then peripheral retiming yields a solution to $\mathcal{P}_C^{'}$