Logic Synthesis and Verification

Jie-Hong Roland Jiang
江介宏

Department of Electrical Engineering
National Taiwan University

Fall 2013

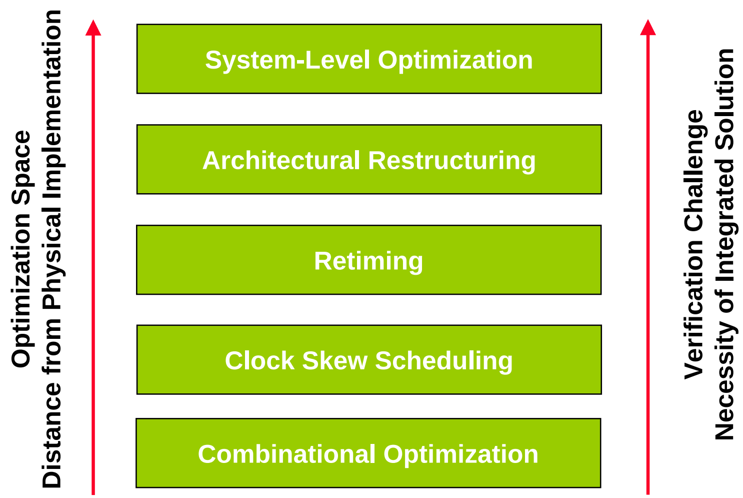
Sequential Synthesis

part of the following slides are by courtesy of Andreas Kuehlmann

Motivation

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

Overview of Circuit Optimization



Sequential Optimization Techniques

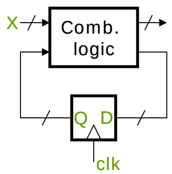
- Clock skew scheduling
  - balance path delays by adjusting the relative clocking schedule of individual registers
- Retiming
  - balance path delays by moving registers within circuit topology
  - can be interleaved with combinational optimization techniques
- Architectural restructuring
  - add sequential redundancy
    - fixed: does not change input/output behavior
    - flexible: change input output behavior
- System-level optimization

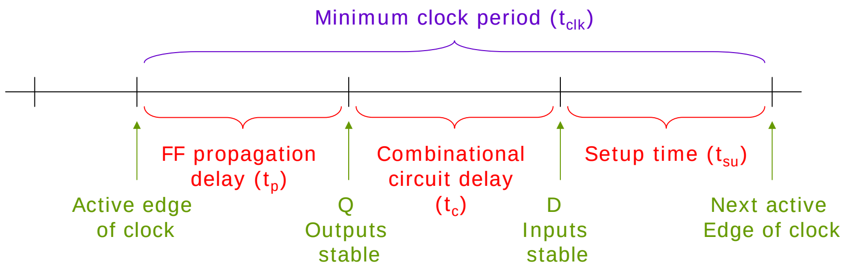
Integration in Design Flow

- Optimization space
  - significantly more optimization freedom at a higher level for improving performance, power, area, etc.
- Distance from physical implementation
  - difficult to accurately model impacts on final implementation
  - difficult to mathematically characterize optimization space
- Verification challenge
  - departure from combinational comparison model would impede formal equivalence checking
  - different simulation behaviors cause acceptance problems
- Necessity of tight tool integration!

Sequential Timing Constraints

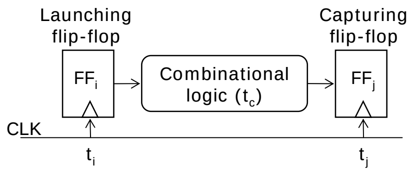
- Minimum clock period
  - $t_{clk}(min) = max\{t_p, t_x\} + t_c + t_{su}$, where t_x is the time after the active clock edge at which the X inputs are stable

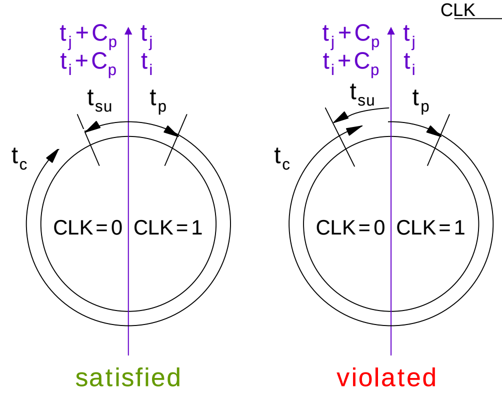




Sequential Timing Constraints

- Setup-time constraint
  - $C_p \geq t_p + t_c^{max} + t_{su}$

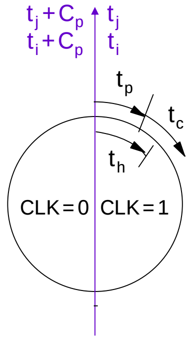




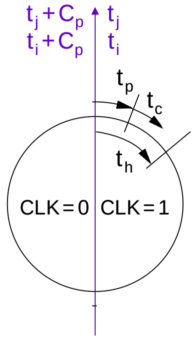
C_p: clock period

Sequential Timing Constraints

- Hold-time constraint
  - $t_p + t_c^{min} \geq t_h$

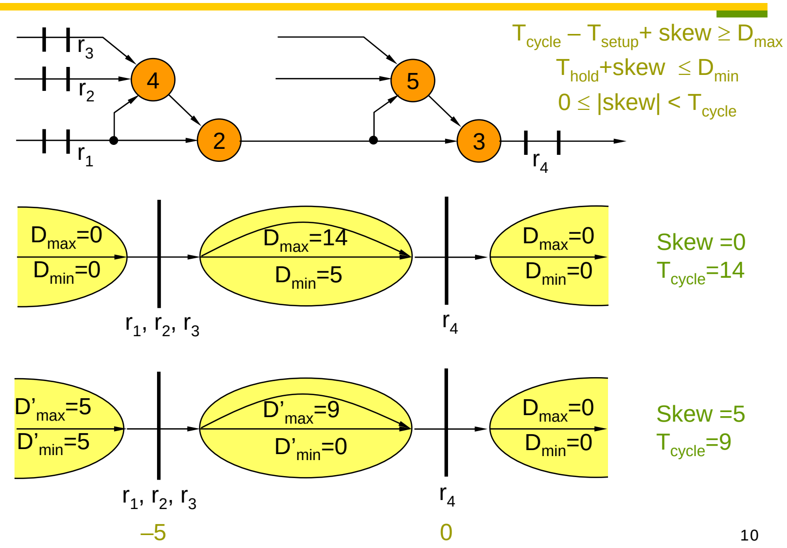


satisfied



violated

Clock Skew Scheduling



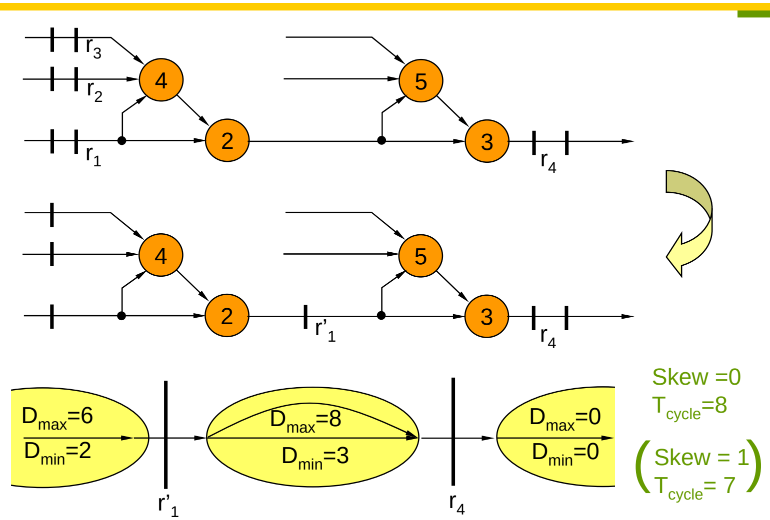
Clock Skew Scheduling

- By controlling clock delays on registers, clock frequency may be increased
  - Do not change transition and output functions (not the case in retiming)
    - Good for functional verification
  - May require sophisticated timing verification
- Clock skew: clock signal arrives at different registers at different times
  - Positive skew: the sending register gets the clock earlier than the receiving register
  - Negative skew: the receiving register gets the clock earlier than the sending register

Clock Skew Scheduling

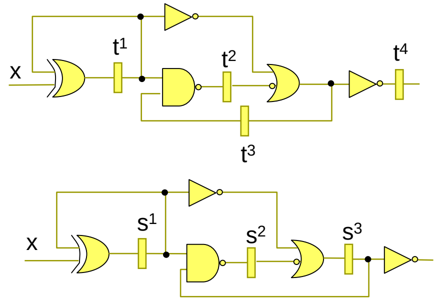
- Pros
  - Inexpensive "post synthesis" technique to further reduce clock period
  - Combinational design model is preserved
- Cons
  - Setup **and** hold time constraints must be obeyed
    - including hold time constraints from scan chain
  - Interleaving with combinational optimizations impossible
  - Replication of clocking tree required

Retiming



Retiming

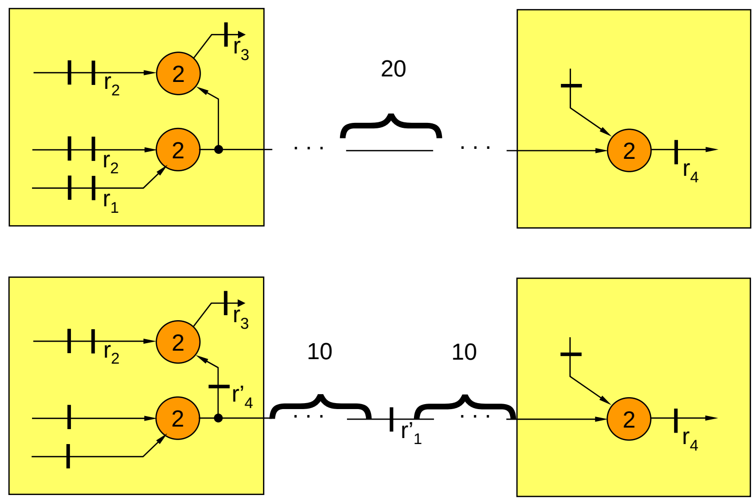
- Optimize sequential circuits by repositioning registers
  - Move registers so that clock cycle decreases or register count decreases
  - Input-output behavior is preserved; however, transition and output functions are changed due to the register movement



Retiming

- Pros
  - Only setup time constraint (0 clock skew)
  - Simple integration with other logical (e.g. combinational) or physical optimizations
    - E.g., iterative retiming and resynthesis
  - Easy combination with clock skew scheduling to obtain global optimum
- Cons
  - Change combinational model of design
    - Severe impact on verification methodology
  - Inaccurate delay model
  - Computation of equivalent reset state required

Architectural Retiming



Architectural Retiming

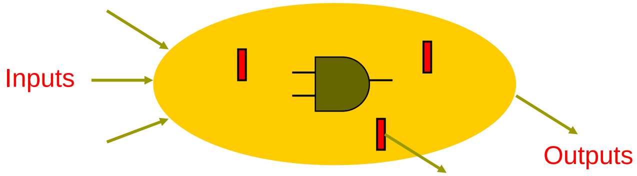
- Pros
  - Smooth extension of regular retiming
  - Potential to alleviate global performance bottlenecks by adding sequential redundancy and pipelining
- Cons
  - Significant change of design structure
    - substantial impact on verification methodology
  - Flexible architectural restructuring changes I/O behavior
    - existing RTL specification methods not always applicable

Verification Issues

- Timing verification unchanged
- Functional verification affected
  - Except for clock skew scheduling, sequential optimization does change register (transition) functions
  - Traditional combinational equivalence checking not applicable
  - Simulation runs not recognizable by designers - acceptance problems
  - Solution:
    - preserve retime function (mapping function) from synthesis for:
      - reducing sequential EC problem back to combinational case
        - *no false positives possible!*
      - modifying simulation model to reproduce original simulation output

Retiming Circuits

- Objectives:
  - Reduce clock cycle time
  - Reduce register count (area)
  - Reduce power, etc.
- Input: A netlist of gates and registers

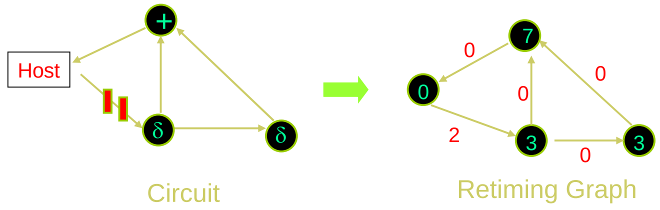


Retiming Circuits

- Circuit represented as retiming graph G(V, E) [Leiserson and Saxe 1983, 1991]
  - V: vertex set representing logic gates
  - E: edge set representing connections
  - d(v) = delay of gate/vertex v, $(d(v) \geq 0)$
  - w(e) = number of registers on edge e, $(w(e) \geq 0)$

Retiming Circuits

- Example
  - Synchronous circuit assumption: every cycle of a circuit has at least one register, i.e., no combinational loop



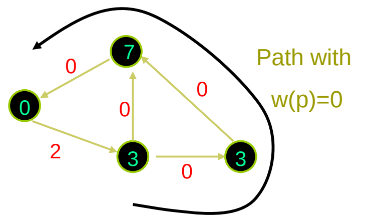
The host node represents the environment that interacts with the circuit via the primary inputs and outputs

Operation	delay
δ	3
+	7

Retiming Circuits

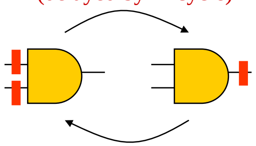
- For a path *p* : $v_0 \xrightarrow{e_0} v_1 \xrightarrow{e_1} \cdots v_{k-1} \xrightarrow{e_{k-1}} v_k$
  - Path delay $d(p) = \sum_{i=0}^{k} d(v_i)$ (includes endpoints)
  - Path weight $w(p) = \sum_{i=0}^{k-1} w(e_i)$
- Minimum clock cycle

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



Retiming Circuits

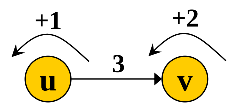
- Atomic operation
  - Move registers across a gate in a forward or backward direction



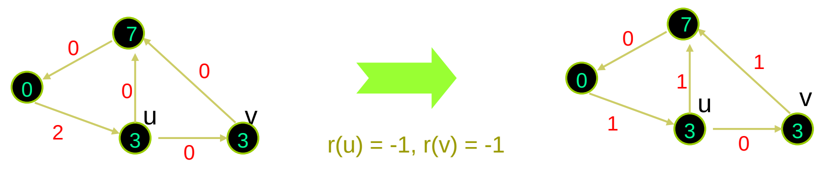
  - Does not affect gate functionality, but timing

Retiming Circuits

- Retiming can be formalized with a retime function *r*: V → ℤ, where ℤ is the set of integers
  - I.e., a retime function performs integer labeling on vertices
- Weight update after retiming with *r*
  - $w_r(e) = w(e) + r(v) - r(u)$, for edge e= (u,v)
  - $w_r(p) = w(p) + r(t) - r(s)$, for path p from s to t
- A retiming with some *r* is legal if $w_r(e) \geq 0$, $\forall e \in E$



$w_r(u,v) = 3 + 2 - 1 = 4$



Min-Cycle Retiming

- **Problem Statement**: (minimum cycle retiming)
  Given $G(V, E)$ with delay function d and weight function w, find a legal retiming r so that

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

  is minimized

- **Retiming**: two important matrices
  - Register weight matrix

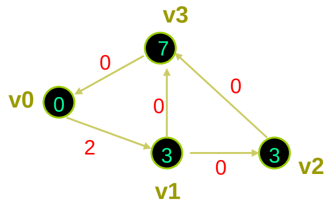
$$W(u,v) = \min_{p}\{w(p) : u \xrightarrow{p} v\}$$

  - Delay matrix

$$D(u,v) = \max_{p}\{d(p) : u \xrightarrow{p} v, w(p) = W(u,v)\}$$

Min-Cycle Retiming

- Example



For some constant α, minimum clock cycle $c \le \alpha \Leftrightarrow \forall p$, if $d(p) > \alpha$ then $w(p) \ge 1$

W = register path weight matrix (minimum # registers on all paths between u and v)
D = path delay matrix (maximum delay on the paths between u and v with w(p)=W(u,v))

W

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

D

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

Don't count paths passing through the host!

Min-Cycle Retiming

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

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

- Take the least w(p) (tightest constraint) $r(u)-r(v) \le W(u,v)-1$
  - Note: This is independent of the path from u to v, so we just need to apply it to u, v such that $D(u,v) > \alpha$

Min-Cycle Retiming

- Example

Assume α = 7

Legality: $r(u)-r(v) \le w(e)$

$$r(v_0)-r(v_1) \le 2$$
$$r(v_1)-r(v_2) \le 0$$
$$r(v_1)-r(v_3) \le 0$$
$$r(v_2)-r(v_3) \le 0$$
$$r(v_3)-r(v_0) \le 0$$

D>7: $r(u)-r(v) \le W(u,v)-1$

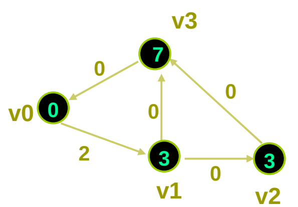
$$r(v_0)-r(v_3) \le 1$$
$$r(v_1)-r(v_0) \le -1$$
$$r(v_1)-r(v_3) \le -1$$
$$r(v_2)-r(v_0) \le -1$$
$$r(v_2)-r(v_3) \le -1$$

W

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

D

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



All constraints are in the difference-of-2-variable form and closely related to shortest path problem

Min-Cycle Retiming

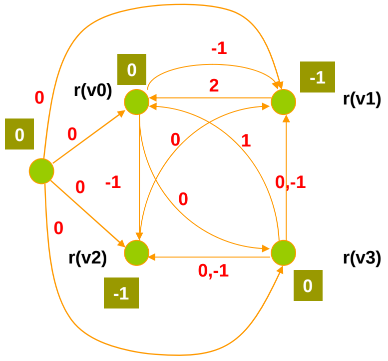
- **Example**

Legality: r(u)-r(v)≤w(e)

D>7: r(u)-r(v)≤W(u,v)-1

Legality	D>7
$r(v_0)-r(v_1)\le 2$	$r(v_0)-r(v_3)\le 1$
$r(v_1)-r(v_2)\le 0$	$r(v_1)-r(v_0)\le -1$
$r(v_1)-r(v_3)\le 0$	$r(v_1)-r(v_3)\le -1$
$r(v_2)-r(v_3)\le 0$	$r(v_2)-r(v_0)\le -1$
$r(v_3)-r(v_0)\le 0$	$r(v_2)-r(v_3)\le -1$

Constraint graph



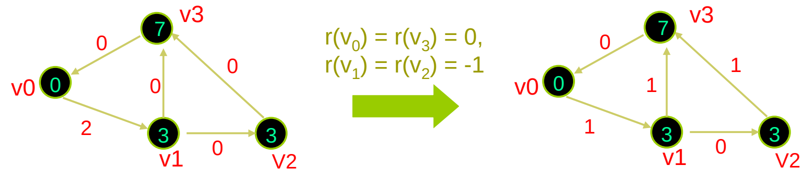
Search shortest path on constraint graph:
Bellman-Ford algorithm O(|V||E|) or O($|V|^3$)

A solution exists if and only if there exists **no** negative weighted cycle

A solution is $r(v_0) = r(v_3) = 0$, $r(v_1) = r(v_2) = -1$

Min-Cycle Retiming

- To find the minimum cycle time, do a binary search among the entries of the D matrix O(|V||E| log|V|)



Clock cycle = 3+3+7=13

Clock cycle = 7

Min-Cycle Retiming

- Theorem: *r* is a legal retiming on *G* such that the clock cycle c ≤ α for some constant α if and only if
  1. $r(v_h)=0$
  2. $r(u)-r(v) \le w(e)$ for every edge $e(u,v)$
  3. $r(u)-r(v) \le W(u,v)-1$ (i.e. register count > 1) for every (u, v) with $D(u,v) > \alpha$

- Solve the integer linear programming problem
  - Bellman-Ford method O($|V|^3$)

Min-Cycle Retiming

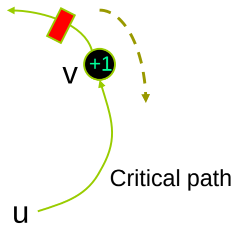
- Algorithm of optimal retiming:
  1. Compute *W* and *D*
  2. Binary search the minimum achievable clock period by applying Bellman-Ford algorithm to check the satisfication of the prior Theorem
  3. Derive *r*(*v*) under the minimum achievable clock period found in Step 2

- Complexity O($|V|^3 \lg|V|$)

Min-Cycle Retiming

- Two more algorithms:

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



2. Also, Mixed Integer Linear Program formulation

Min-Area Retiming

- Goal: minimize number of registers used

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

where a_v is a constant

Min-Area Retiming

- Minimize:

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

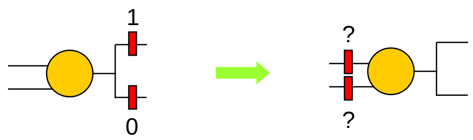
- Subject to:

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

- Note: It is reducible to a flow problem

Retiming Issues

- Computation of equivalent initial states
  - Equivalent initial states may not always exist



  - General solution requires replication of logic for initialization
- Timing models
  - Too far away from actual implementation

Retiming + Clock Scheduling

- Mathematical formulation
  - s: E→ℝ, a real edge labeling
  - s(e) denotes the clock signal delay of all registers of e

- In addition to the register weight matrix and delay matrix for the maximum delay, we also need the minimum paths delays

$$W(u,v) = \min_p \{w(p) : u \xrightarrow{p} v\}$$
$$D(u,v) = \max_p \{d(p) : u \xrightarrow{p} v, w(p) = w(u,v)\}$$
$$D_{\min}(u,v) = \min_p \{d(p) : u \xrightarrow{p} v, w(p) = w(u,v)\}$$

Retiming + Clock Scheduling

- A valid retiming and clock skew schedule is an assignment to r and s such that:

(1) $w_r \geq 0$

(2) $\forall (u',u),(v,v'):$

$$w(u',u) > 0 \wedge w(v,v') > 0 \wedge W(u,v) = 0 \Rightarrow$$
$$D_{\min}(u,v) + s(u',u) - s(v,v') \geq T_{hold} \wedge$$
$$D(u,v) + s(u',u) - s(v,v') \leq T_{clock} - T_{setup}$$

- Solution Mixed Integer Linear Program (MILP)

Retiming & Resynthesis

- Combine retiming and combinational optimization
  - Retime registers such that the circuit has a large combinational logic block for optimization
  - Resynthesize the combinational logic block with combinational logic minimization techniques
  - Retiming and resynthesis can be iterated
    - Can achieve any state re-encoding

Retiming & Resynthesis

- Example