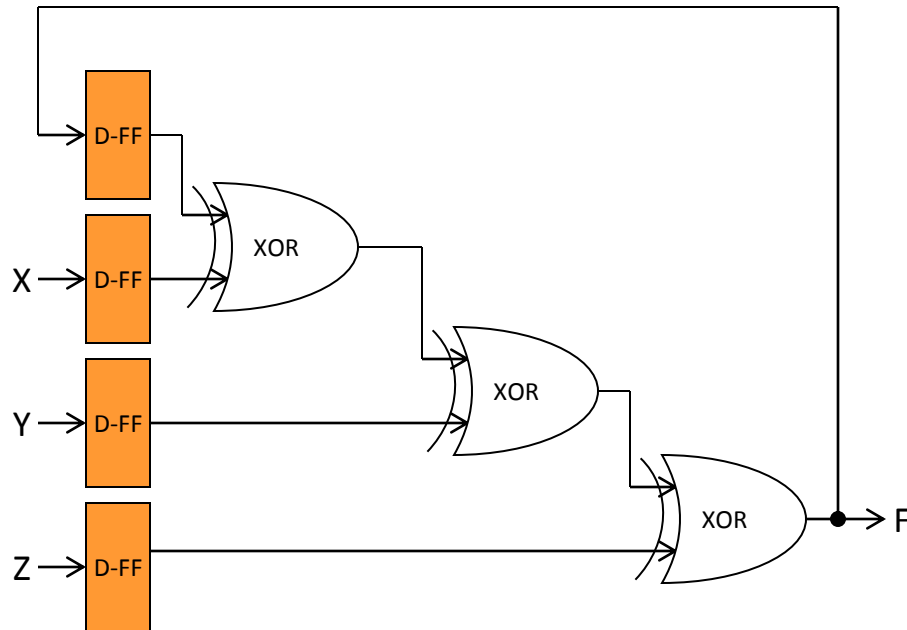


Retiming

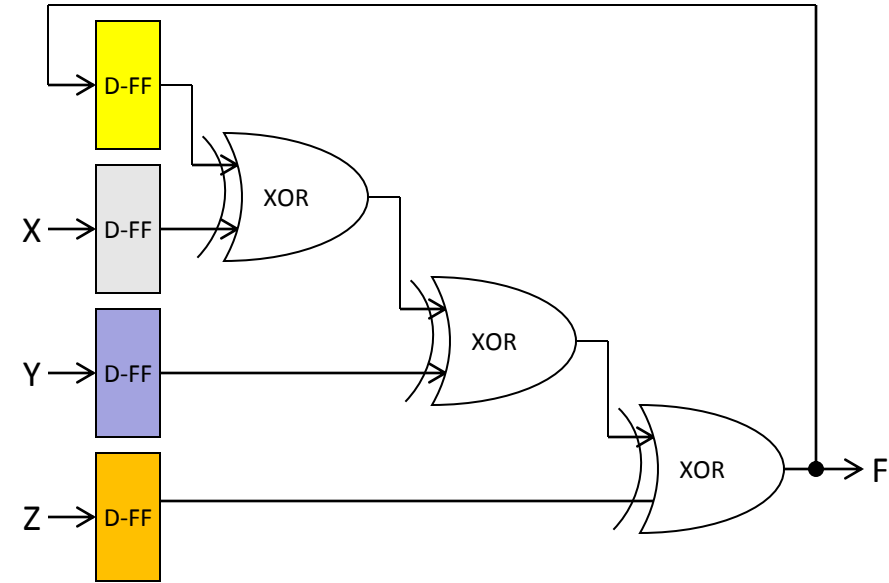
Consider the Following Circuit

- Suppose $T_{\text{XOR}} = 3 \text{ ns}$, $T_{\text{pcq}} = 1 \text{ ns}$, $T_{\text{setup}} = 1 \text{ ns}$, then this circuit can be clocked at $1 \text{ ns} + (3 \times 3 \text{ ns}) + 1 \text{ ns} = 11 \text{ ns}$.



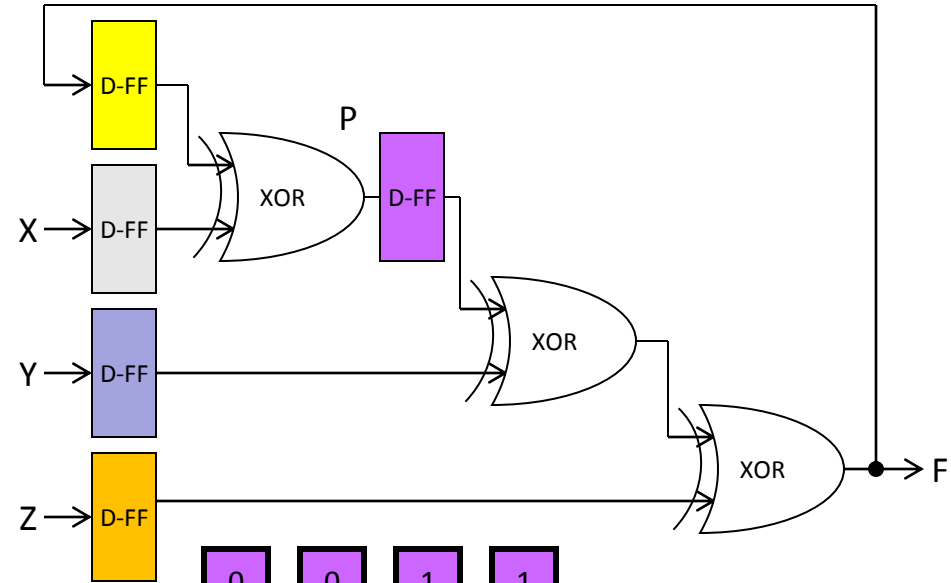
Why Are They Not Equivalent?

- Suppose FFs are initialized to 0



	0	0	1	0
	0	1	1	1
	0	1	1	1
	0	1	1	1

F	0	1	0	1
X	1	1	1	1
Y	1	1	1	1
Z	1	1	1	1

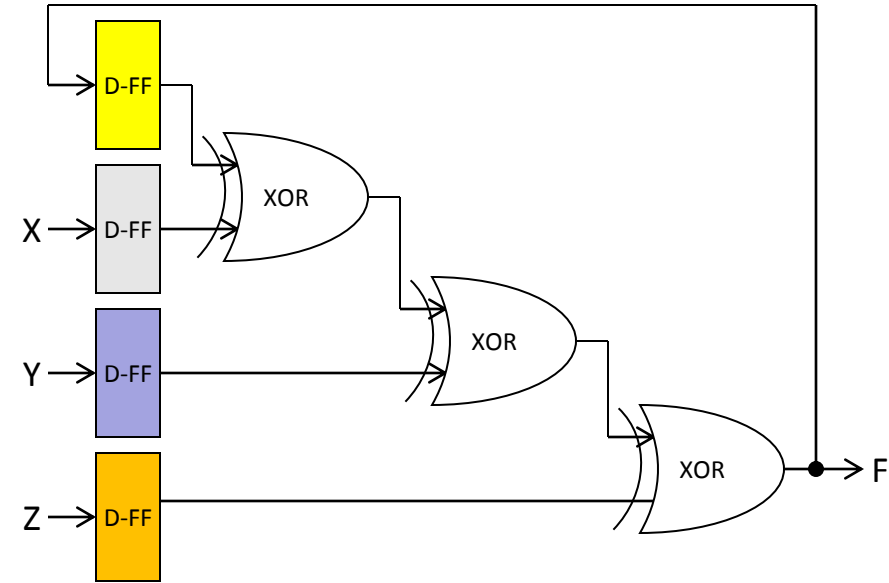


	0	0	1	1
	0	0	0	1
	0	1	1	1
	0	1	1	1

F	0	0	1	1
X	1	1	1	1
Y	1	1	1	1
Z	1	1	1	1

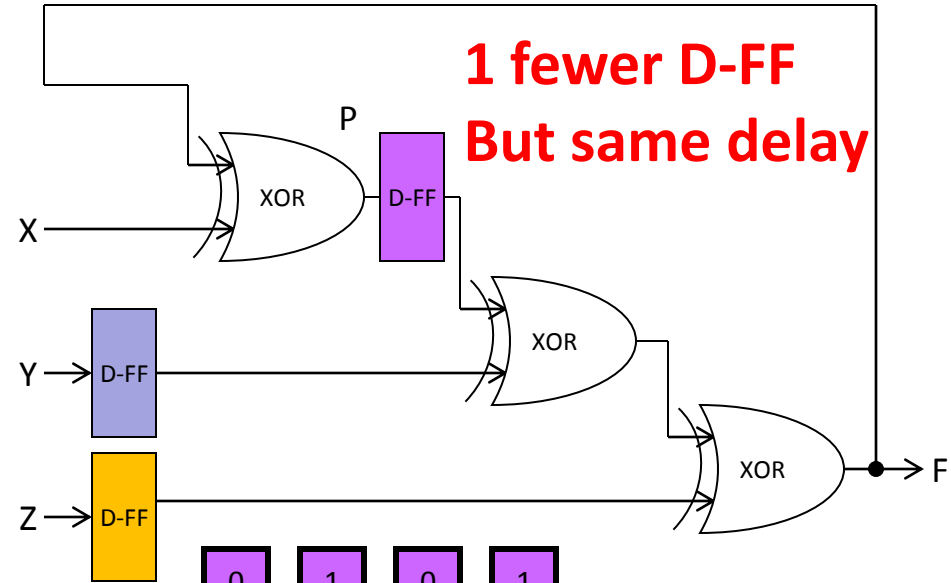
Are These Equivalent?

- Suppose FFs are initialized to 0



0	0	1	0
0	1	1	1
0	1	1	1
0	1	1	1

F	0	1	0	1
X	1	1	1	1
Y	1	1	1	1
Z	1	1	1	1



**1 fewer D-FF
But same delay**

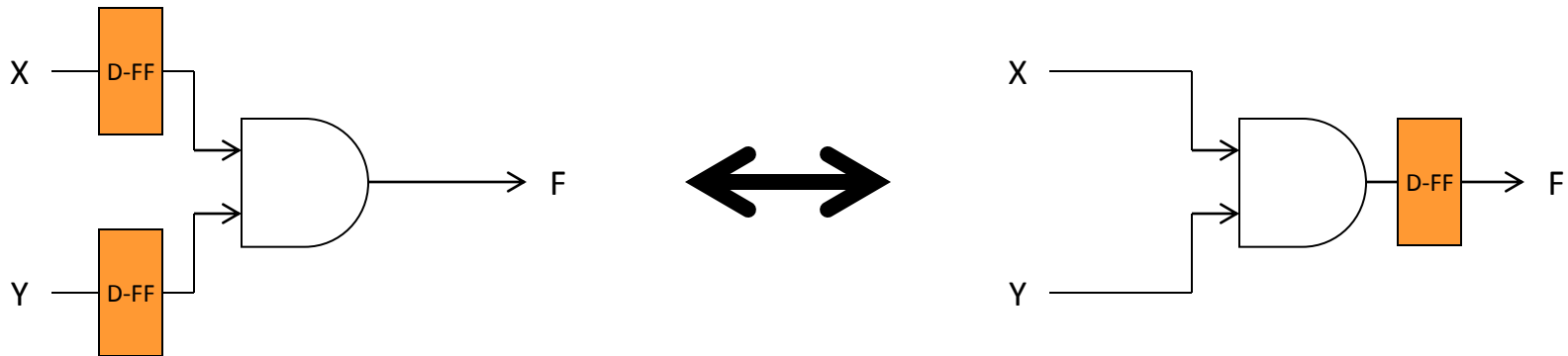
0	1	0	1
---	---	---	---

0	1	1	1
0	1	1	1

F	0	1	0	1
X	1	1	1	1
Y	1	1	1	1
Z	1	1	1	1

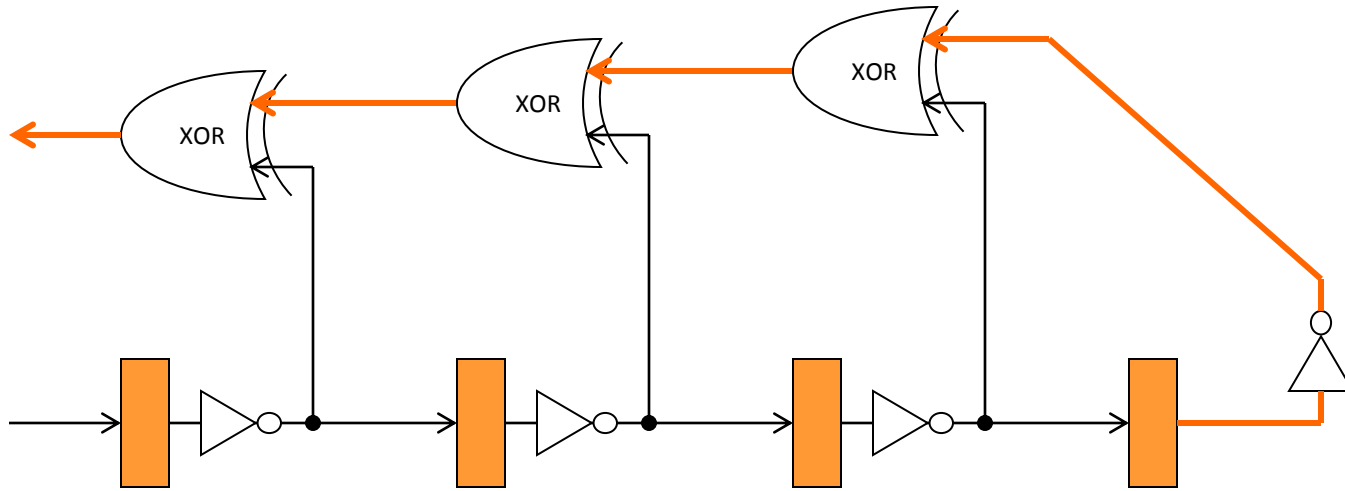
Basic Idea of Retiming

- If you have 2 FFs at the inputs, you can move it to the output
- Or if you have a FF at the output, you can move it to the inputs
- In general, can move N FFs from inputs to output, and vice versa

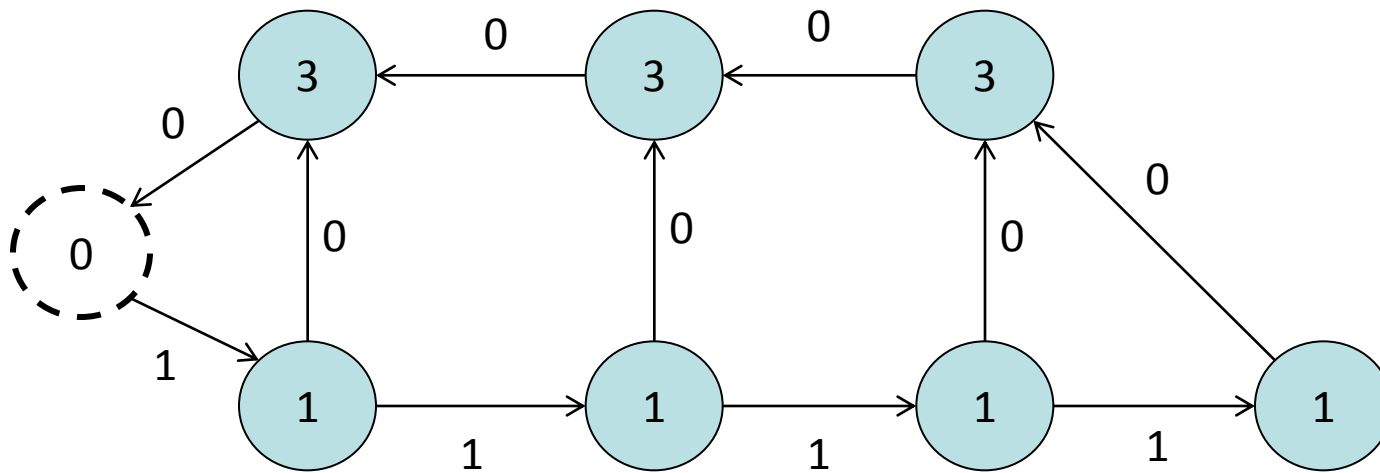


Example

- How to move FFs around to minimize clock period? (assuming $T_{pcq} = T_{setup} = 0$)



Graph Model

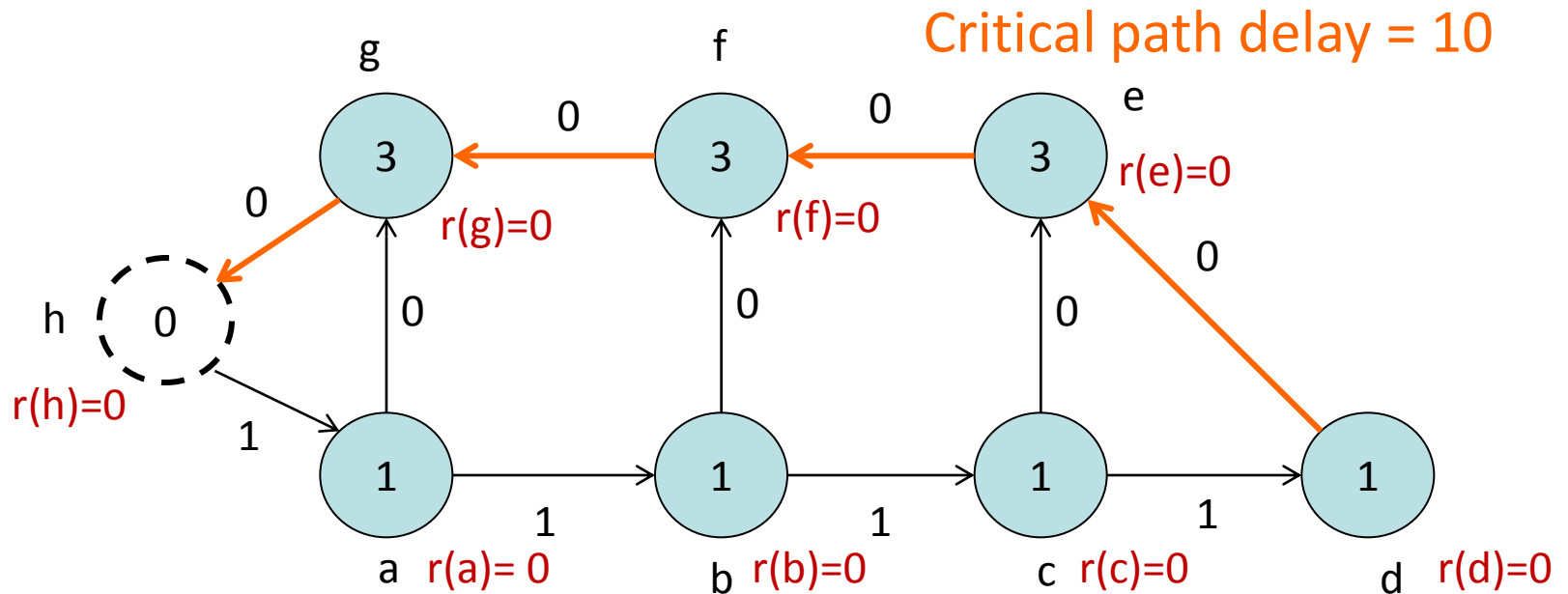


- Vertex v_i , combinational node, delay = $d(v_i)$
 - All inputs and outputs connect through a faux node “host” with $d(\text{host}) = 0$
- Edge $e(v_i, v_j)$ or e_{ij} , weight w_{ij} = number of flip-flops between v_i and v_j

Path Delay and Path Weight

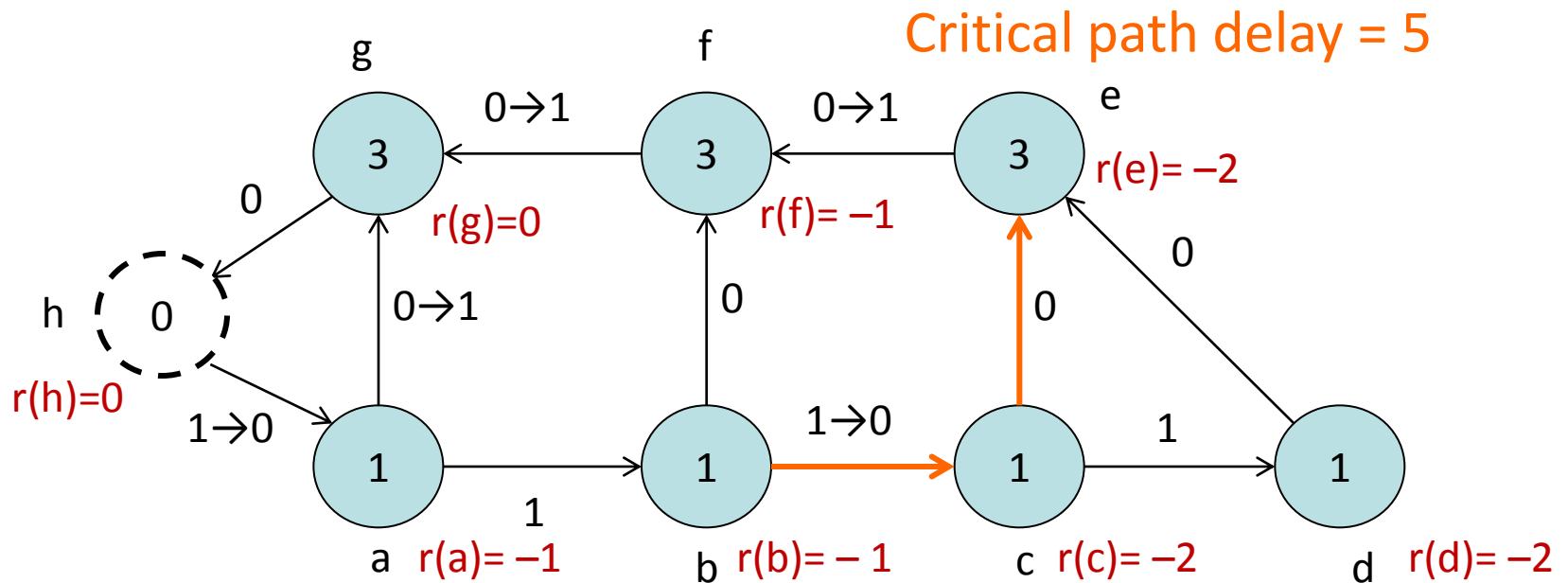
- A set of connected nodes specify a path
- Path delay = $\sum d(v_i)$ = comb. delay of path
- Path weight = $\sum w_{ij}$ = # FFs along the path
- Retiming of a node i denoted by an integer r_i
 - It represents the number of registers moved across, initially $r_i = 0$
 - Register moved from output to input, $r_i \rightarrow r_i + 1$
 - Register moved from input to output, $r_i \rightarrow r_i - 1$
 - After retiming, edge weight $w_{ij}' = w_{ij} + r_j - r_i$

Example



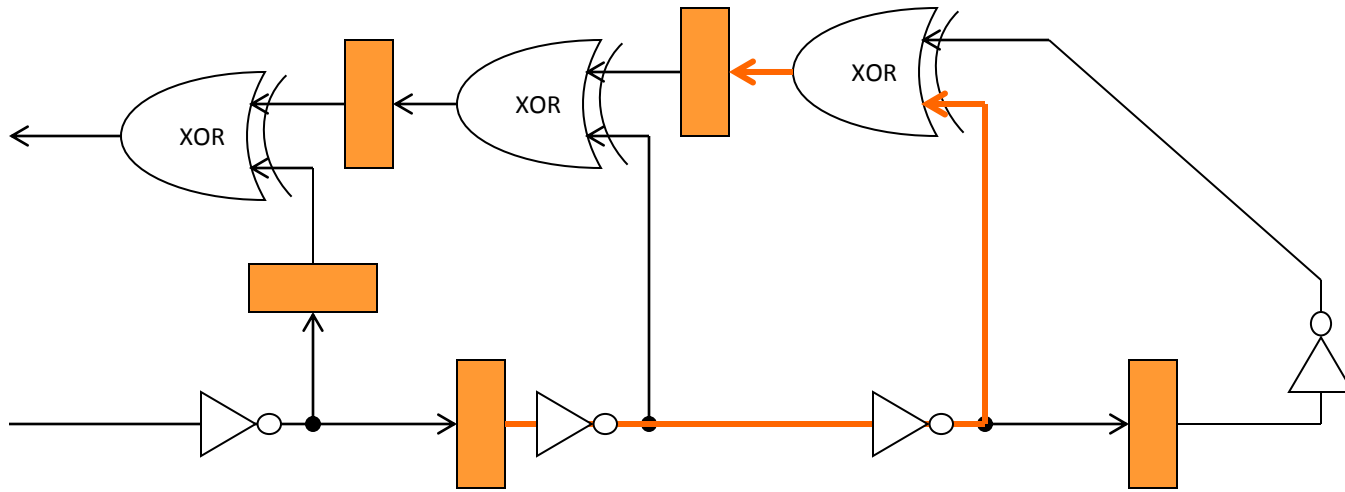
Initial retiming vector = $\{0,0,0,0,0,0,0,0\}$

Retimed Example



Optimal retiming vector = $\{-1, -1, -2, -2, -2, -1, 0, 0\}$

Optimized Circuit



Retiming Theorem

- Given a network $G(V, E)$ and a cycle time T , $(r_1, r_2, \dots)$ is a feasible retiming if and only if:

$$r_i - r_j \leq w_{ij} \quad \text{for all edges}$$

$$r_i - r_j \leq W(v_i, v_j) - 1 \quad \text{for all node-pairs } (v_i, v_j) \text{ such that } D(v_i, v_j) > T$$

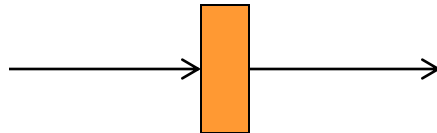
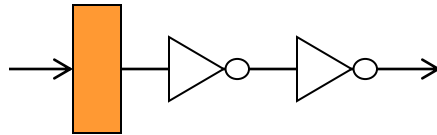
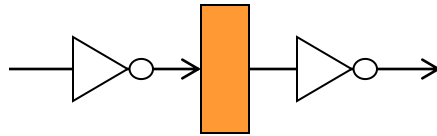
where

$W(v_i, v_j)$ is the minimum weight path between v_i and v_j

$D(v_i, v_j)$ is the maximum delay among all minimum weight paths between v_i and v_j .

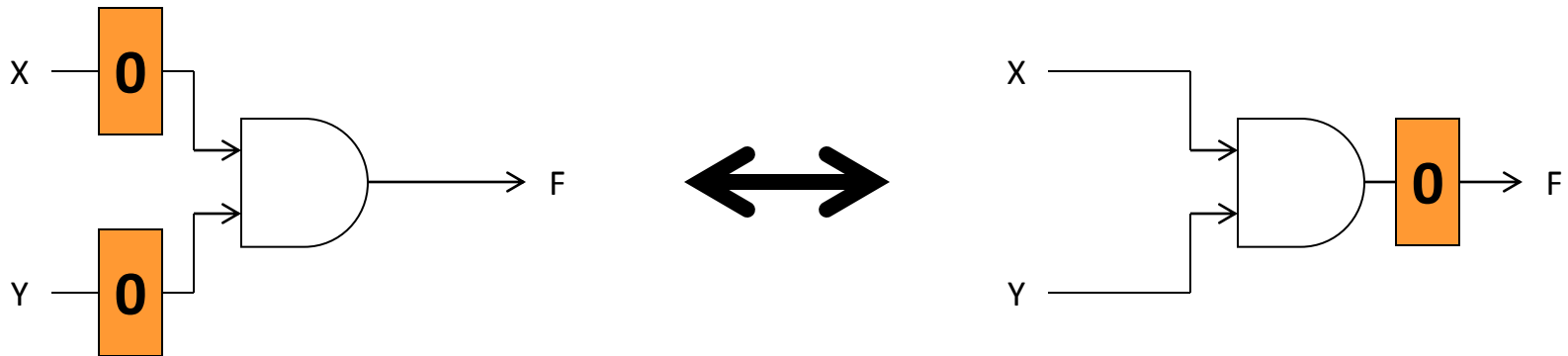
- Above is a Linear Program, which can be tested for feasibility for a given T .
- Binary search over possible cycle time T .

Retiming & Resynthesis



Initial State

- What should be the new initial state?



Cannot Always Recover Initial State

- Cannot always get exactly the same initial state behavior on the retimed circuit
- For some applications, a startup transient may not be a problem

