

# Logic Synthesis and Verification

Jie-Hong Roland Jiang  
江介宏

Department of Electrical Engineering  
National Taiwan University



Fall 2012

1

# Timing Analysis & Optimization

Reading:

*Logic Synthesis in a Nutshell*

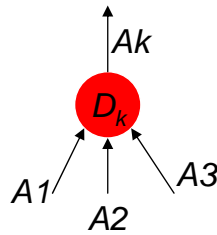
Sections 5 & 6

part of the following slides are by  
courtesy of Andreas Kuehlmann

2

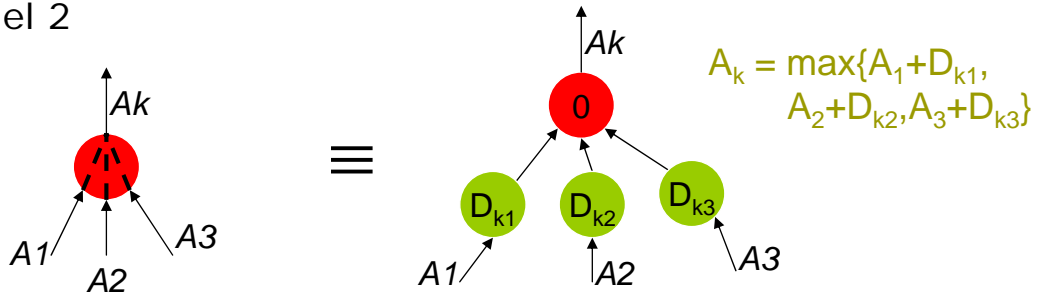
# Delay Models

## Model 1



$A_k = \text{arrival time} = \max\{A_1, A_2, A_3\} + D_k$ , where  $D_k$  is the delay at node  $k$ , parameterized according to function  $f_k$  and fanout node  $k$

## Model 2



Can also have different times for rise time and fall time

3

# Gate Delay

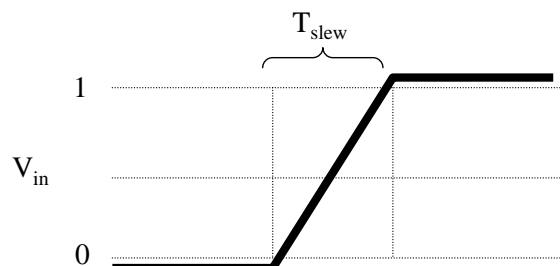
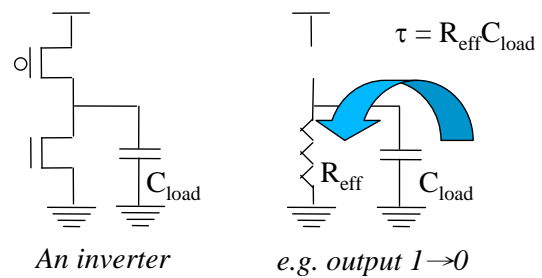
The delay of a gate depends on its circuit context, and in particular:

### 1. Output Load

- Capacitive loading  $\propto$  the charges that a gate must move to swing the output voltage
- Due to interconnect and logic fanout

### 2. Input Slew

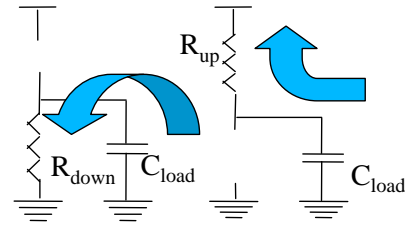
- Slew = transition time
- Slower transistor switching  $\rightarrow$  longer delay, longer output slew



4

# Rising and Falling Edges

- Driving strengths of pull-up and pull-down networks may not be equivalent
  - Rising and falling outputs may have different delays

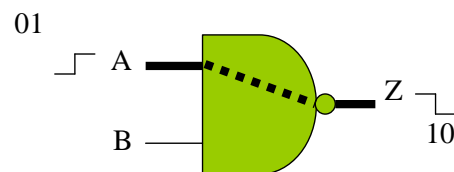


- *Idea:* maintain the latest/earliest arrival time of rising and falling transitions independently
  - Unateness of each input/output pair is encoded in the library
    - Positively unate inputs: only trigger output transitions in the same direction (e.g. an AND gate)
    - Negatively unate inputs: only trigger output transitions in the opposite direction (e.g. a NOR gate)
    - A transition on a binate input could trigger either direction on an output (e.g. an XOR gate)
  - Only considers local functionality, but allows a less conservative analysis

5

# Timing Library

- Timing library contains all relevant information about each standard cell
  - E.g., pin direction, clock, pin capacitance, etc.
- Delay (fastest, slowest, and often typical) and output slew are encoded for each input-to-output path and each pair of transition directions
- Values typically represented as 2 dimensional look-up tables (of output load and input slew)
  - Interpolation is used



```
Path(
    inputPorts(A),
    outputPorts(Z),
    inputTransition(01),
    outputTransition(10),
    "delay_table_1",
    "output_slew_table_1"
);
```

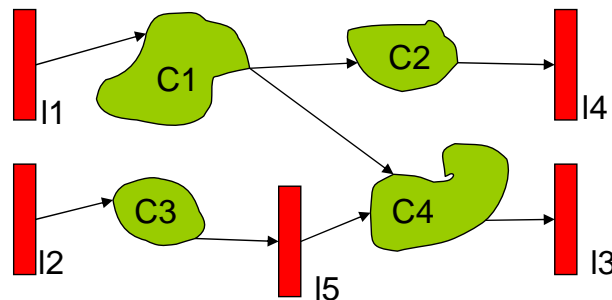
**"delay\_table\_1"**  
Output Load (nF)

Input slew (ns)	1.0	2.0	4.0	10.0
0.1	2.1	2.6	3.4	6.1
0.5	2.4	2.9	3.9	7.2
1.0	2.6	3.4	4.0	8.1
2.0	2.8	3.7	4.9	10.3

6

# Sequential Circuit

- **Arrival times** known at  $I_1$ ,  $I_2$ , and  $I_5$  (PIs and latch outputs)
- **Required times** known at  $I_3$ ,  $I_4$ , and  $I_5$  (Pos and latch inputs)
- Delay analysis gives arrival and required times (hence **slacks**) for  $C_1$ ,  $C_2$ ,  $C_3$ ,  $C_4$



7

# Arrival Time Calculation

```
// level of PI nodes initialized to 0,  
// the others are set to -1.  
// Invoke LEVEL from PO  
Algorithm LEVEL(k) { // levelize nodes  
  if( k.level != -1)  
    return(k.level)  
  else  
    k.level = 1+max{LEVEL( $k_i$ ) |  $k_i \in \text{fanin}(k)$ }  
  return(k.level)  
}  
  
// Compute arrival times:  
// Given arrival times on PI's  
Algorithm ARRIVAL() {  
  for L = 1 to MAXLEVEL  
    for {k | k.level = L}  
       $A_k = \text{MAX}\{A_{k_i}\} + D_k$   
}
```

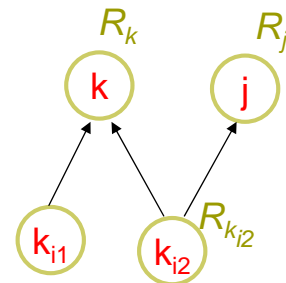
8

# Required Time Calculation

## Required time:

- Given required times on primary outputs
- Traverse in reverse topological order (i.e. from POs to PIs)
- If  $(k_i, k)$  is an edge between  $k_i$  and  $k$ , the required time of this edge is  $R_{k_i,k} = R_k - D_k$
- The required time of output of node  $k$  is  $R_k = \min \{ R_{k,k_j} \mid k_j \in \text{fanout}(k) \}$

```
// Compute required times:
// Given required times on PO's
Algorithm REQUIRED() {
  for L = MAXLEVEL-1 to 0
    for {k | k.level = L}
      R_k = MIN{R_{k,k_i}}
}
```



9

# Slack

## Slack:

- Slack at the output of node  $k$  is

$$S_k = R_k - A_k$$

$$\text{Since } R_{j,k} = R_k - D_k$$

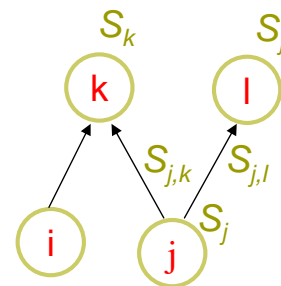
$$S_{j,k} = R_{j,k} - A_j$$

$$S_{j,k} + A_j = R_k - D_k = S_k + A_k - D_k$$

$$\text{Since } A_k = \max \{ A_i, A_j \} + D_k$$

$$S_{j,k} = S_k + \max \{ A_{\text{fanin}(k)} \} - A_j$$

$$S_j = \min \{ S_{j,\text{fanout}(j)} \}$$



## Note:

- Each edge of a circuit graph has a slack and required time
- Negative slack is bad

10

# Static Timing Analysis

□ A **static critical path** of a Boolean network is a path  $P = \{n_1, n_2, \dots, n_p\}$ , where  $S_{n_k, n_{k+1}} < 0$

■ Note:

□ If a node  $n$  is on a static critical path, then at least one of the fanin edges of  $n$  is critical. Hence, all critical paths reach from an input to an output.

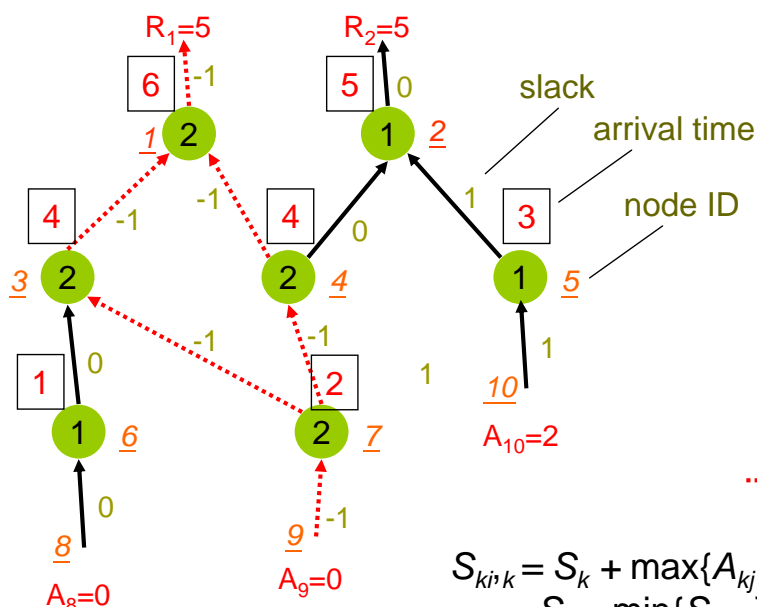
□ There may be several critical paths

□ Timing optimization is a min-max problem:  
minimize  $\max\{-S_i, 0\}$

11

# Static Timing Analysis

□ Example



$A_1 = 6$	$R_1 = 5$
$A_2 = 5$	$R_2 = 5$
$S_1 = -1$	$R_3 = 3$
$S_2 = 0$	$R_7 = 1$
$S_{3,1} = -1$	$R_9 = -1$
$S_{4,1} = -1$	
$S_{4,2} = 0$	
$S_{5,2} = 1$	
$S_{6,3} = 0$	
$S_{7,3} = -1$	
$S_{7,4} = -1$	
$S_{7,5} = 1$	
$S_{8,6} = 0$	
$S_{9,7} = -1$	

..... critical path edges

$$S_{k_i, k} = S_k + \max\{A_{k_j}\} - A_{k_i}, k_j, k_i \in \text{fanin}(k)$$

$$S_k = \min\{S_{k, k_j}\}, k_j \in \text{fanout}(k)$$

12

# Static Timing Analysis

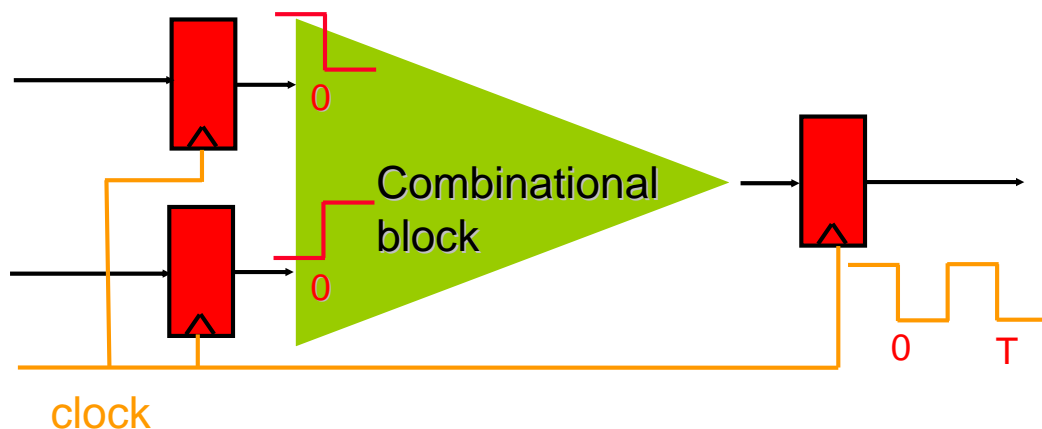
## □ Problems

- We want to determine the **true** critical paths of a circuit in order to:
  - determine the minimum cycle time that a circuit will function correctly
  - identify critical paths for performance optimization - don't want to try to optimize **wrong** (non-critical) paths
- Implications:
  - Don't want **false paths** (produced by static delay analysis)
  - Delay model is a **worst** case model
    - Need to ensure correctness for case where  $i^{\text{th}}$  gate delay  $\leq D_i^{\text{Max}}$

13

# Functional Timing Analysis

- Functional timing analysis estimates when the output of a given circuit gets stable



14

# Functional Timing Analysis

## □ Motivation

- Timing verification
  - Verifies whether a design **meets** a given timing constraint
    - E.g., cycle-time constraint
- Timing optimization
  - Needs to identify **critical** portion of a design for further optimization
  - Critical path identification
- In both applications, **accurate** analysis is desirable

15

# Gate-Level Timing Analysis

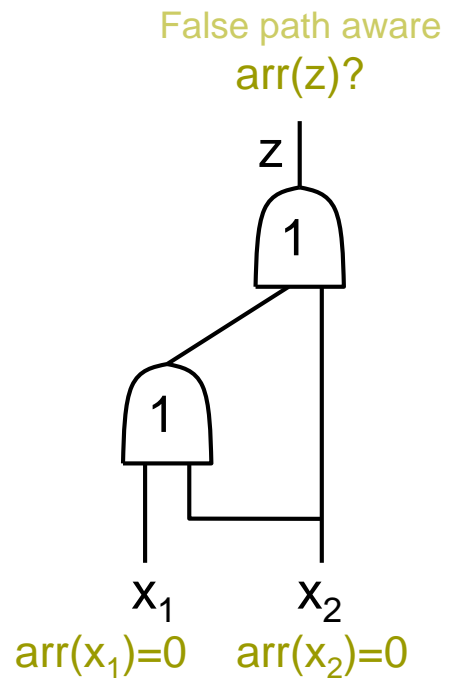
- **Naïve approach** - Simulate all input vectors with SPICE
  - Accurate, but too expensive
- **Gate-level timing analysis (our focus)**
  - Less accurate than SPICE due to the level of abstraction, but much more **efficient**
  - Scenario:
    - Gate/wire delays are **pre-characterized** (accuracy loss)
    - Perform timing analysis of a gate-level circuit assuming the **gate/wire delays**

16



# Gate-Level Timing Analysis

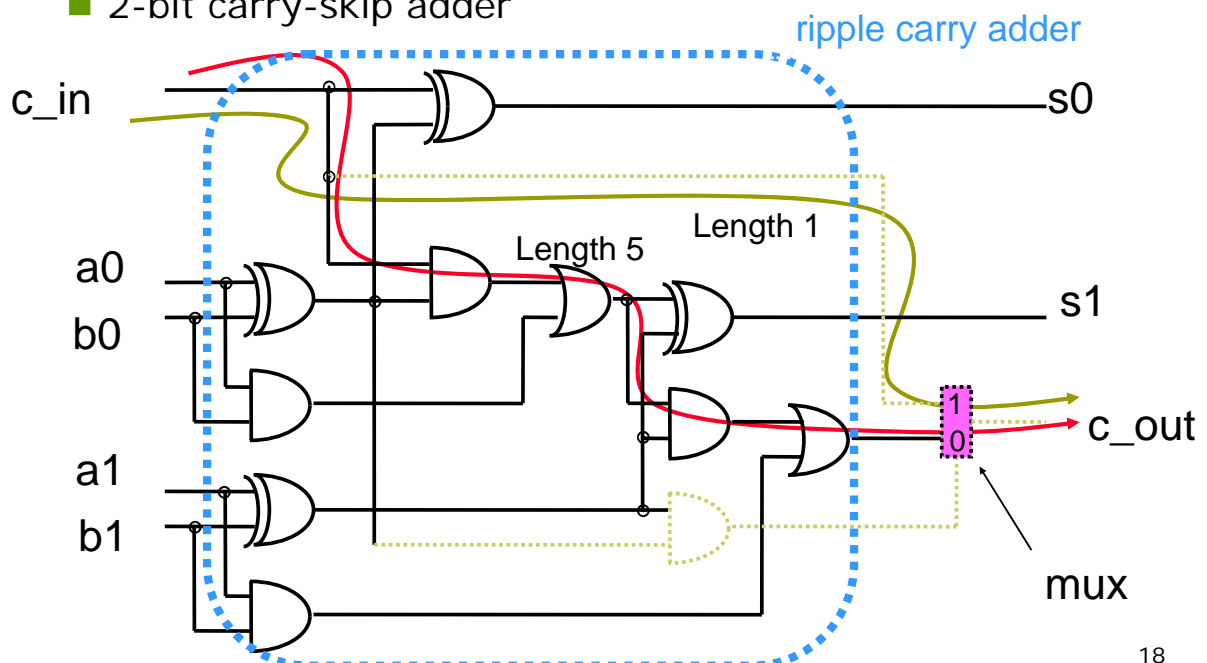
- A naive approach is topological analysis
  - Easy longest-path problem
  - Linear in the size of a network
- Not all paths can propagate signal events
  - **False paths**
  - If all longest paths are false, topological analysis gives delay **over-estimation**
- **Functional timing analysis** = **false-path-aware** timing analysis
  - Compute false-path-aware arrival time



17

# Gate-Level Timing Analysis

- Example
  - 2-bit carry-skip adder



18

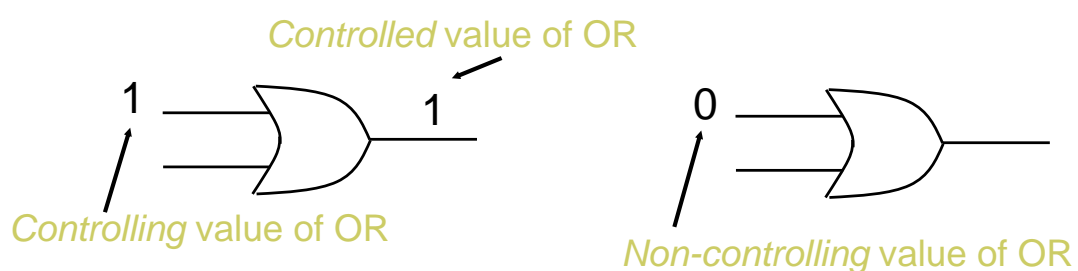
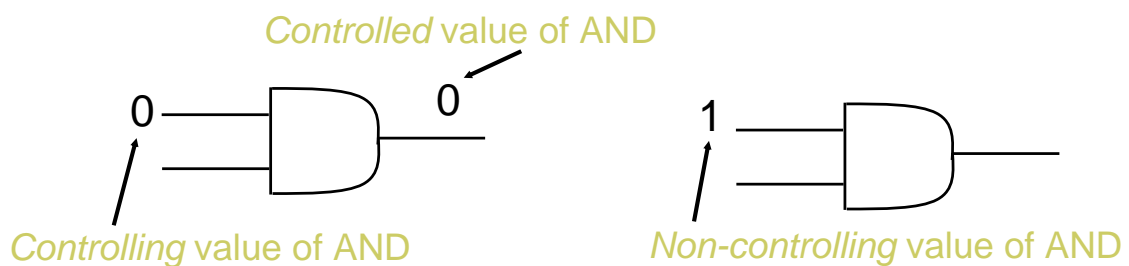
# False Path Analysis

- Is a path **responsible** for circuit delay?
  - If the answer is no, can **ignore** the path for delay computation
- Check the falsity of long paths until we find the longest true path
  - How can we determine whether a path is a false path?
- Delay under-estimation is **unacceptable**
  - Can lead to overlooking timing **violation**
- Delay over-estimation is **acceptable**, but not desirable
  - **Topological** analysis may yield over-estimation, but never under-estimation

19

# False Path Analysis

- Controlling and non-controlling values



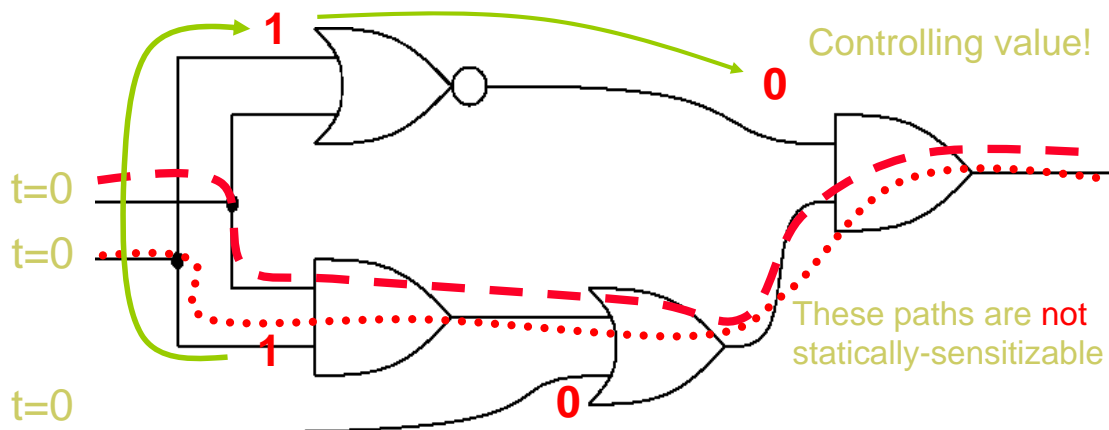
20

# False Path Analysis

## Static Sensitization

### Static sensitization

- A path is *statically-sensitizable* if there exists an input vector such that all the side-inputs to the path are of non-controlling values
  - This is *independent* of gate delays



The longest true path is of length 2?

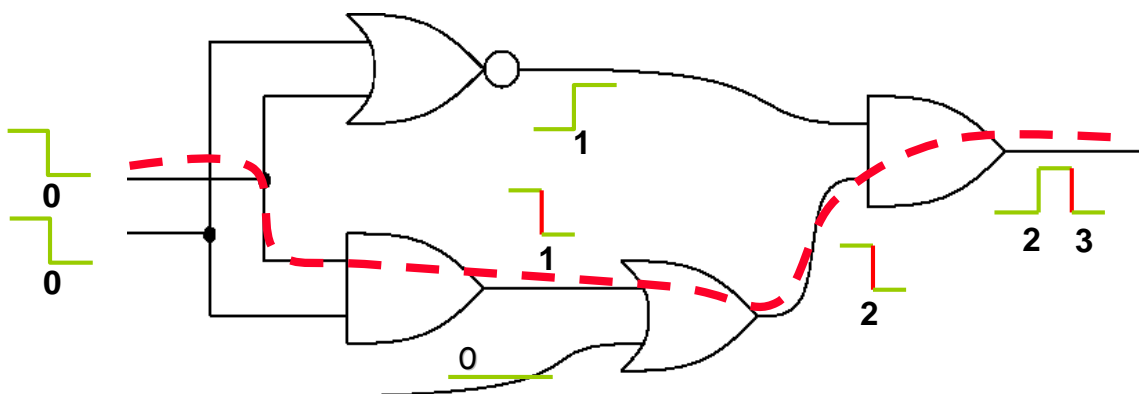
21

# False Path Analysis

## Static Sensitization

### Example

- The (dashed) path is responsible for delay!
- Delay *under-estimation* by static sensitization (delay = 2 when true delay = 3)
  - incorrect condition



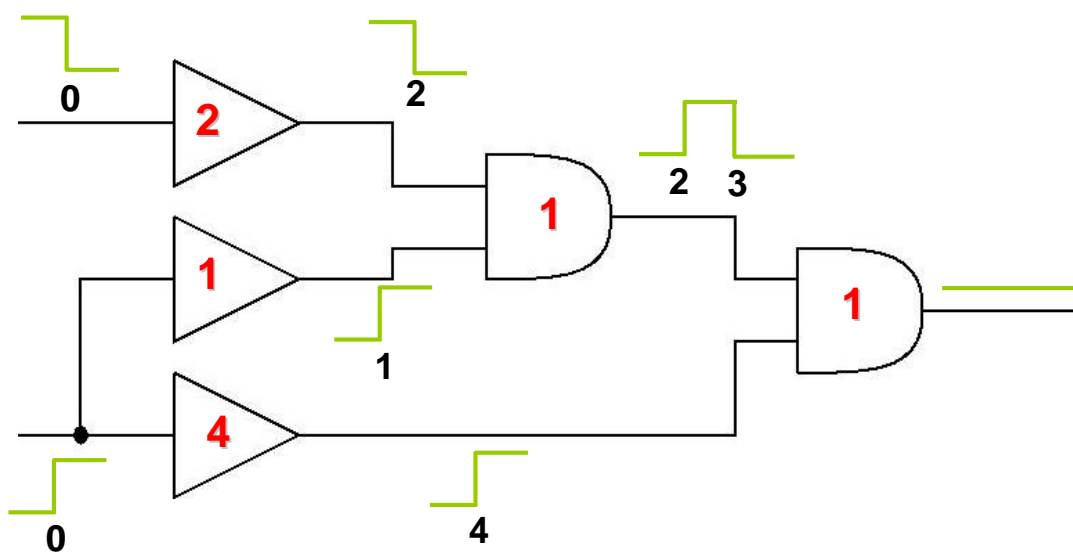
22

## False Path Analysis Static Sensitization

- ❑ Problem: The idea of forcing non-controlling values to side inputs is okay, **but** timing was ignored
  - The same signal can have a **controlling** value at one time and a **non-controlling** value at another time
- ❑ How about **timing simulation** as a correct method?

23

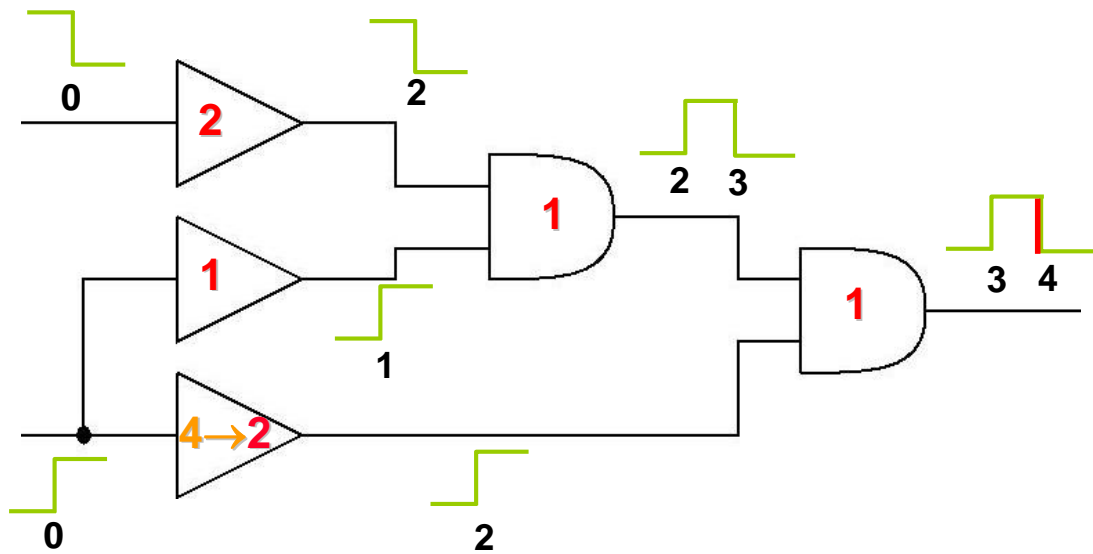
## False Path Analysis Timing Simulation



Implies delay = 0 under input vector (0,1)  
**BUT!**

24

## False Path Analysis Timing Simulation



Implies delay = 4 under the same input (0,1) as before

25

## False Path Analysis Timing Simulation

- ❑ Problem: If gate delays are **reduced**, delay estimates can **increase**
- ❑ **Not** acceptable since
  - Gate delays are just **upper-bounds**, actual delay is in  $[0, d]$ 
    - ❑ Delay **uncertainty** due to manufacturing
  - We are implicitly analyzing a **family** of circuits where gate delays are **within** the upper-bounds

26

# False Path Analysis

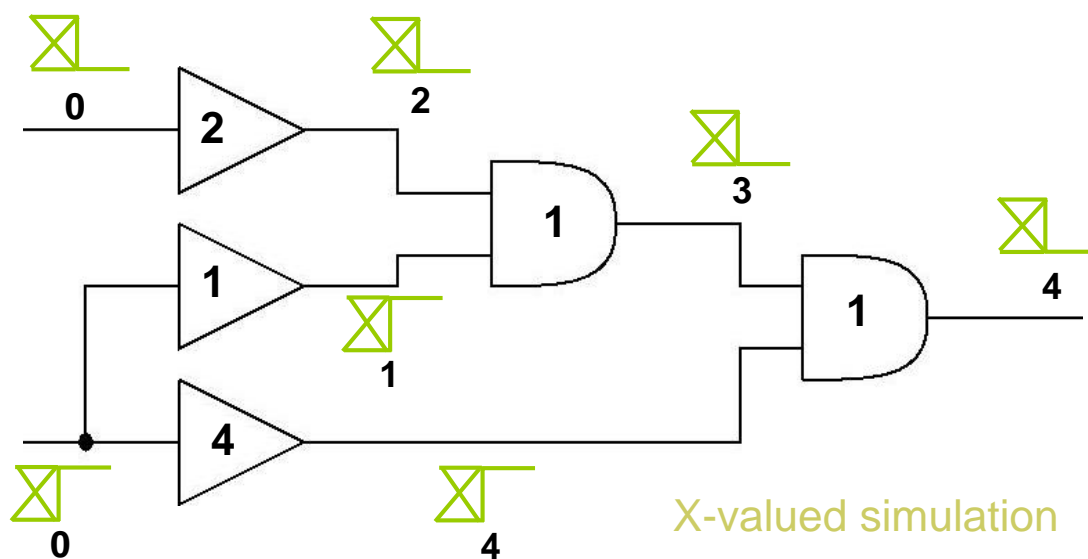
## Timing Simulation


- Definition: Given a circuit C and a delay estimation method `delay_estimate`, if
  - C\* is obtained from C by reducing some gate delays, and
  - $\text{delay\_estimate}(C^*) \leq \text{delay\_estimate}(C)$ ,
 then `delay_estimate` has *monotone speedup*
  
- Timing simulation does *not* have such a property

27

# False Path Analysis

## Timing Simulation



 means that the rising signal occurs anywhere between  $t = -\infty$  and  $t = 4$ .

28

# False Path Analysis Timing Simulation

- Timed 3-valued (0,1,X) simulation
  - called X-valued simulation
  - monotone speedup property is satisfied
  
- Underlying model of
  - floating mode condition
    - Applies to “simple gate” networks only
  - viability
    - Applies to general Boolean networks

29

# False Path Analysis Timing Simulation

- Checking the falsity of every path explicitly is too expensive due to the exponential number of paths
  - Modern approach:
    1. Start:  
 $L = L_{top}$  = topological longest path delay  
 $L_{old} = 0$
    2. Binary search:  
**if** (Delay(L)) (\*)  $L_d = |L - L_{old}|/2, L_{old} = L, L = L + L_d$   
**else**  $L_d = |L - L_{old}|/2, L_{old} = L, L = L - L_d$   
**if** ( $L > L_{top}$  or  $L_d < \text{threshold}$ )  $L = L_{old}$ , **done**
- (\*) Delay(L) = 1 if there is an input vector under which an output gets stable only at time  $t$  where  $L \leq t$  ?
- Can be reduced to a SAT or timed-ATPG problem

30

# SAT-Based False Path Analysis

## Decision problem:

Is there an input vector under which the output gets stable only after  $t = T$  ?

## Idea:

1. Characterize the set of all input vectors  $S(T)$  that make the output stable no later than  $t = T$
2. Check if  $S(T)$  contains  $S$  (all possible input vectors)

Can be solved as a SAT problem:

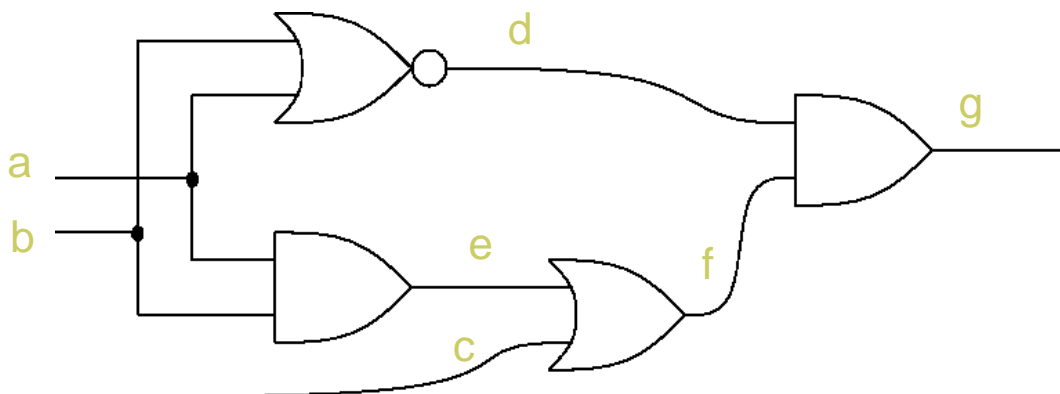
Is  $S \setminus S(T)$  empty? - set difference + emptiness checking

- Let  $F$  and  $F(T)$  be the characteristic functions of  $S$  and  $S(T)$
- Is  $F \wedge \neg F(T)$  satisfiable?

31

# SAT-Based False Path Analysis

## Example



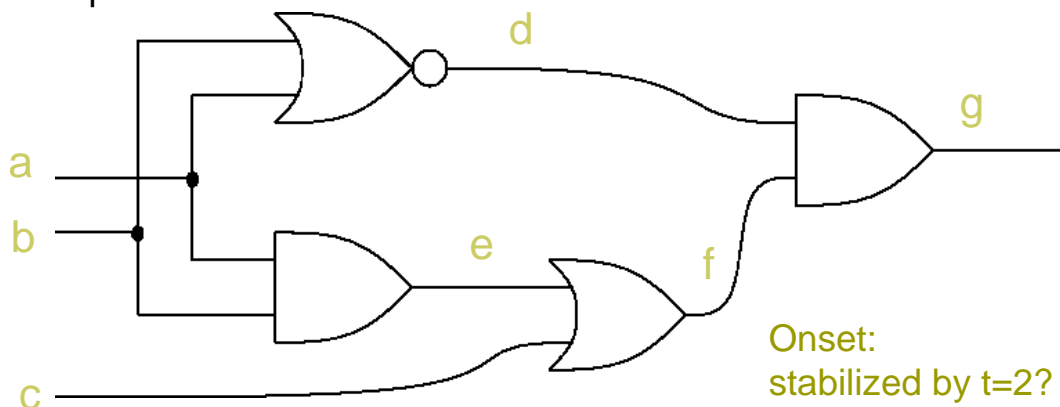
Assume all the PIs arrive at  $t = 0$ , all gate delays = 1  
Is the output stable time  $t > 2$  ?

32



# SAT-Based False Path Analysis

## Example

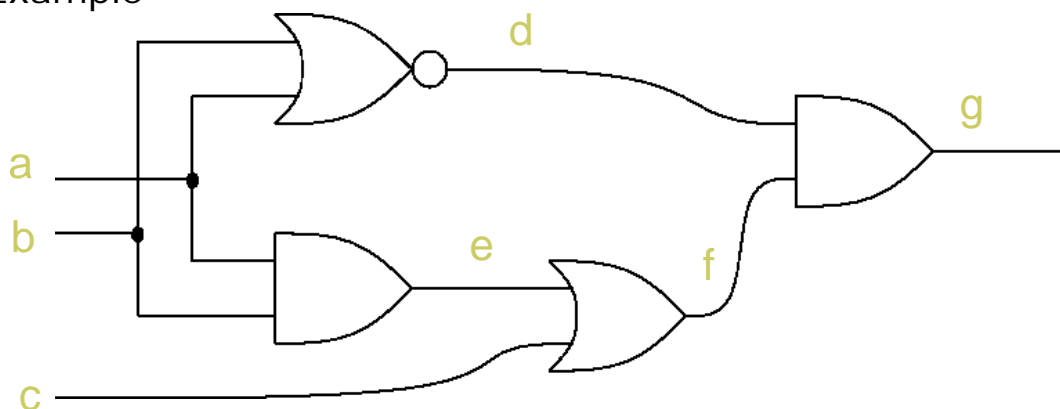


- $g(1, t=2)$  : the set of input vectors under which g gets **stable to value = 1** no later than t = 2
- $g(1, t=2) = d(1, t=1) \cap f(1, t=1) = (a(0, t=0) \cap b(0, t=0)) \cap (c(1, t=0) \cup e(1, t=0)) = \neg a \neg b (c \cup \emptyset) = \neg a \neg b c = S1(t=2)$
- $g(1, t=\infty) = \text{onset} = \neg a \neg b c = g(1, t=2) = S1$

33

# SAT-Based False Path Analysis

## Example

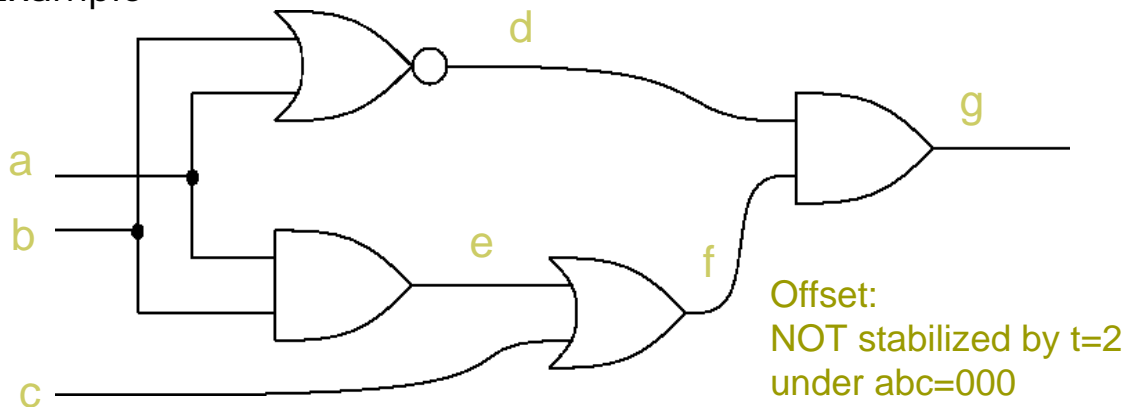


- $g(0, t=2)$  : the set of input vectors under which g gets stable to value = **0** no later than t = 2
- $g(0, t=2) = d(0, t=1) \cup f(0, t=1) = (a(1, t=0) \cup b(1, t=0)) \cup (c(0, t=0) \cap e(0, t=0)) = (a+b) + (\neg c \cap \emptyset) = a+b = S0(t=2)$
- $g(0, t=\infty) = \text{offset} = a+b+\neg c = S0$

34

# SAT-Based False Path Analysis

## Example



- $g(0, t=2)$  : the set of input vectors under which  $g$  gets stable to 0 no later than  $t=2$
- $g(0, t=2) = a+b$
- $g(0, t=\infty) = \text{offset} = a+b+\neg c$
- $g(0, t=\infty) \setminus g(0, t=2) = (a+b+\neg c) \neg(a+b) = \neg a \neg b \neg c$  **satisfiable**

35

# Timing Analysis

## Summary

- False-path-aware arrival time analysis is **well-understood**
  - **Practical** algorithms exist
- Remaining problems
  - **Incremental** analysis (make it so that a small change in the circuit does not make the analysis start all over)
  - Integration with logic **optimization**
  - Clock domain crossing issues

36

# Timing Optimization

Several factors affecting circuit delay:

- Technology
  - Design type (e.g. domino, static CMOS, etc.)
  - Gate type
  - Gate size
- Circuit structure
  - Length of computation paths
  - False paths
  - Buffering
- Electrical parasitics
  - Wire loads
  - Layout

37

# Timing Optimization

## □ Problem statement

Given:

- Initial circuit function description
- Library of primitive functions
- Performance constraints (arrival/required times)

Generate:

- An implementation of the circuit using the primitive functions, such that:
  - performance constraints are met
  - circuit area is minimized

38

# Timing Optimization

## □ Design flow

Behavioral description

Logic and latches

Logic equations

Gate netlist

Layout

Behavior optimization  
(scheduling)

RTL synthesis

Logic synthesis  
•Technology independent  
•Technology mapping

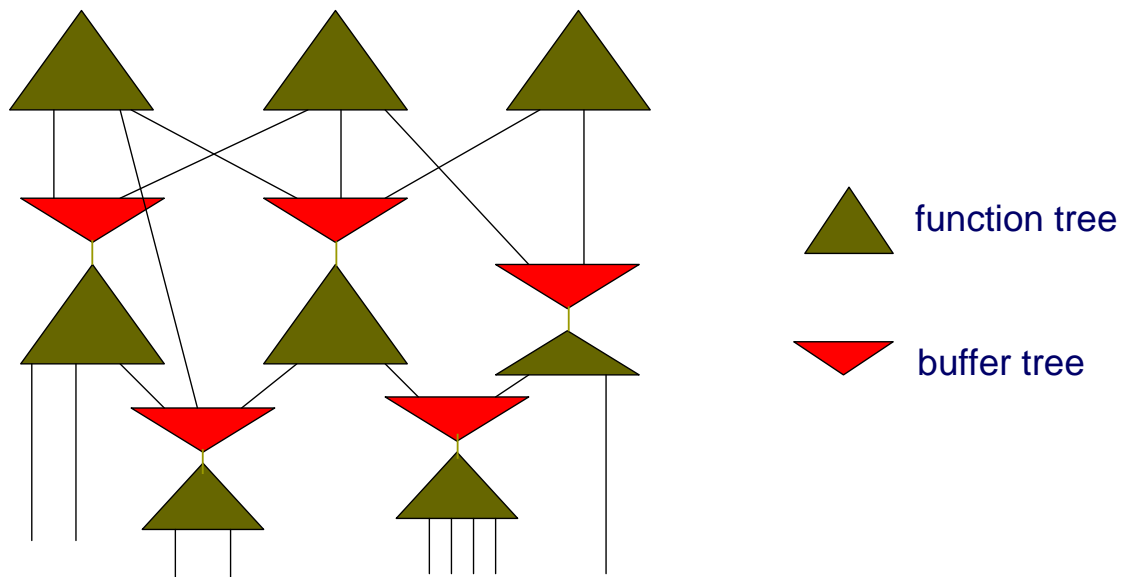
•Gate library  
•Timing constraints  
•Delay models

Timing-driven place and route

39

# Timing Optimization

## □ Buffered circuit structure



40

# Timing Optimization

- Circuit restructuring
  - Reschedule operations to reduce computation time
- Timing-driven technology mapping
  - Selection of gates from library
    - Minimum delay
      - Similar to area minimization under the **load independent model**
    - Minimize delay and area
- Implementation of **buffer trees**
- Gate/wire sizing
- Constant delay synthesis

41

# Timing Optimization

- Circuit restructuring
  - **Global**: Reduce depth of entire circuit
    - Partial collapsing
    - Boolean simplification
  - **Local**: Mimic optimization techniques in **adders**
    - Carry lookahead (**THR** tree-height reduction)
    - Conditional sum (**GST** transformation)
    - Carry bypass (**GBX** transformation)

42

# Timing Optimization

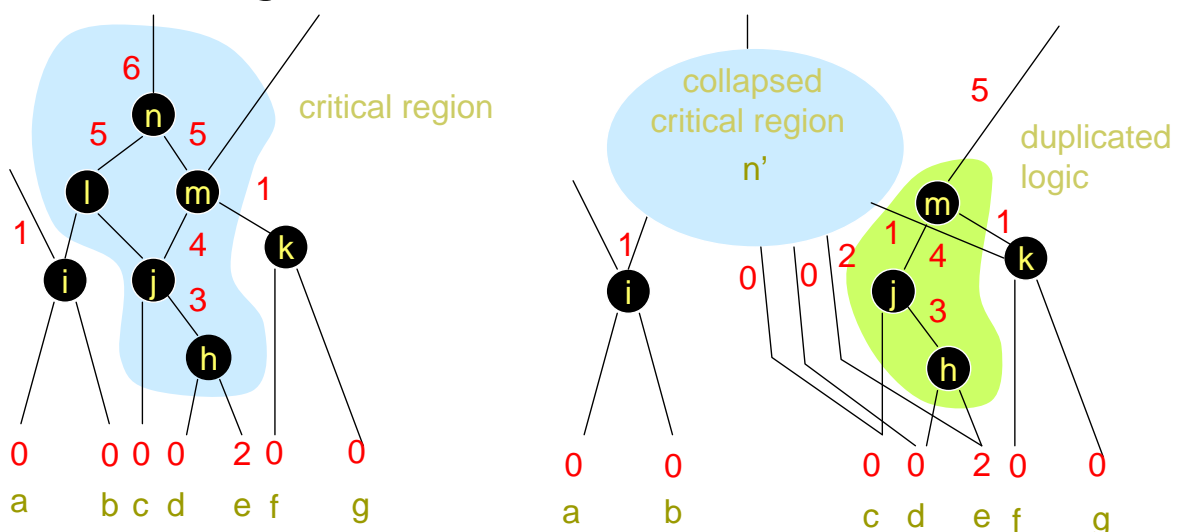
## □ Circuit restructuring

- Performance measured by logic levels, sensitisable paths, technology-dependent delays
- Level-based optimization
  - Tree height reduction
  - Partial collapsing and simplification
  - Generalized select transform
- Sensitisable path based optimization
  - Generalized bypass transform

43

# Timing Optimization

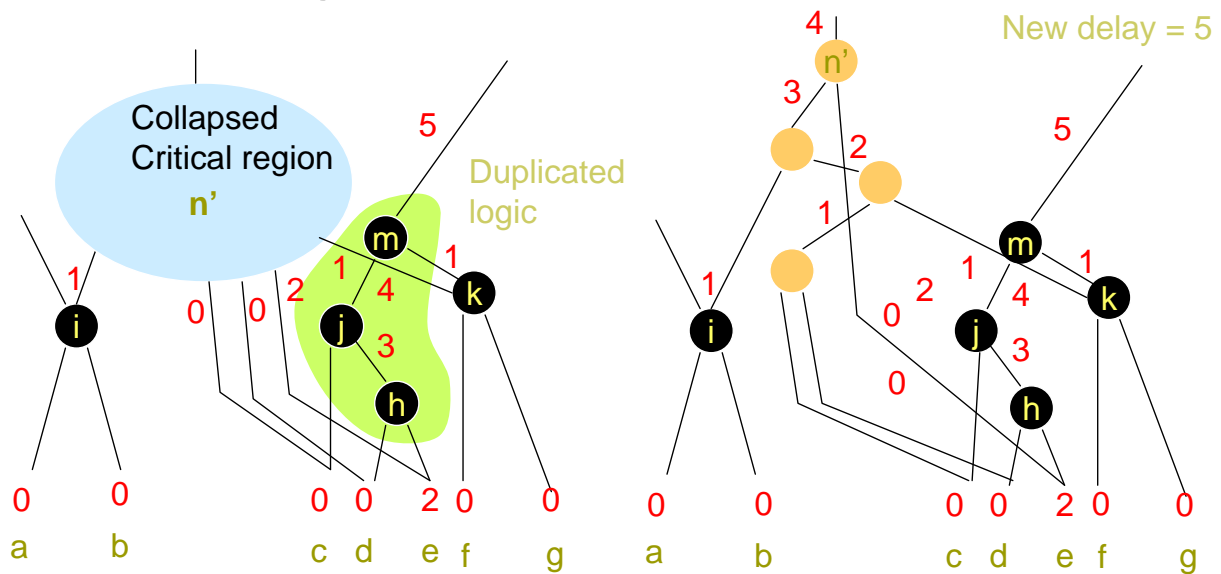
## □ Tree height reduction



44

# Timing Optimization

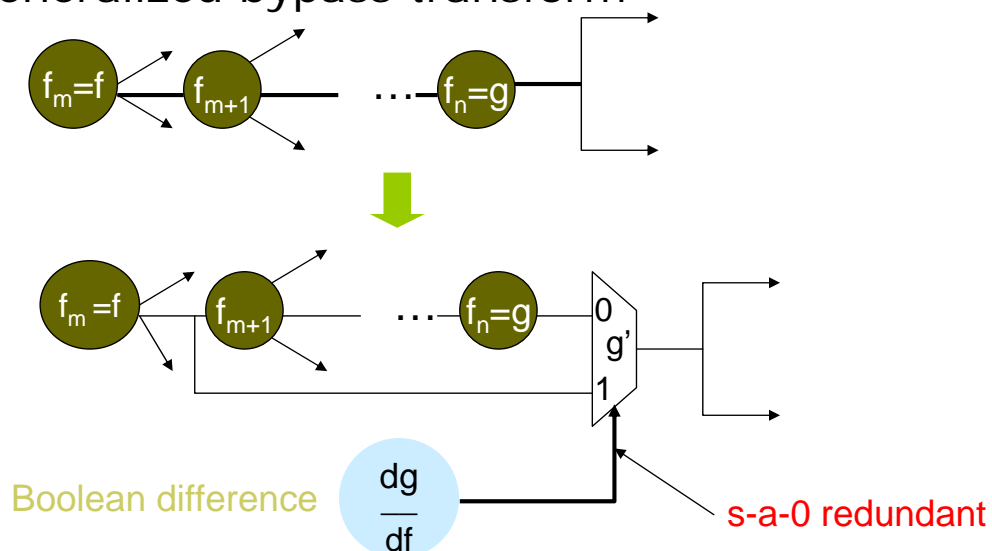
## Tree height reduction



45

# Timing Optimization

## Generalized bypass transform

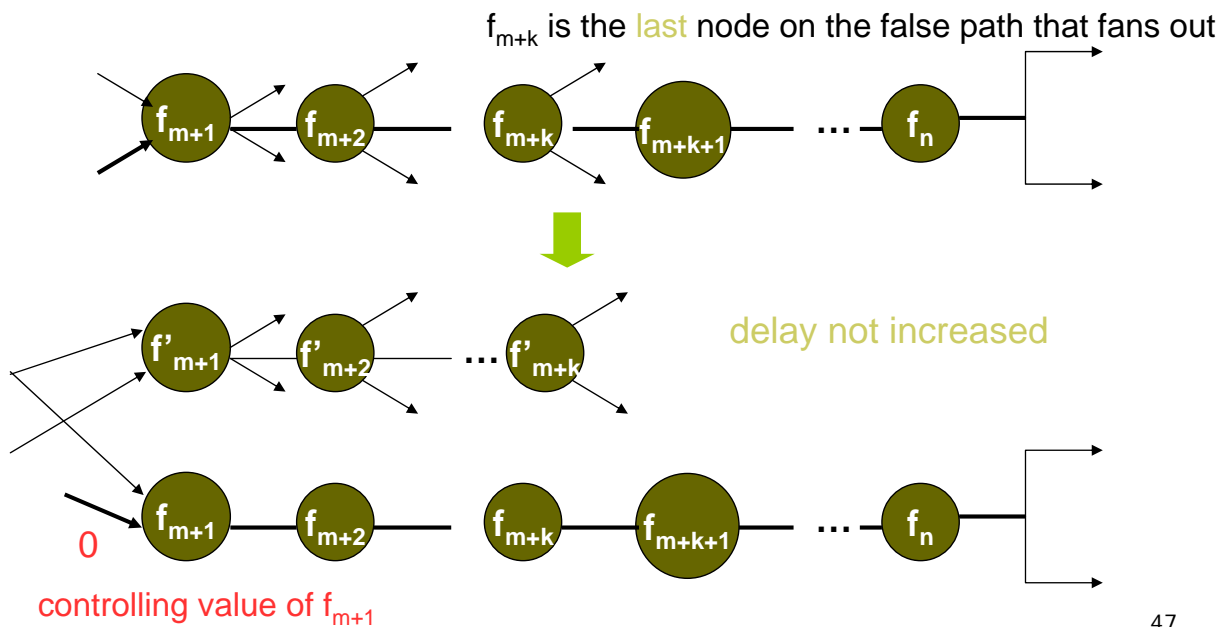


- Make critical path false to bypass critical logic and speed up circuit

46

# Timing Optimization

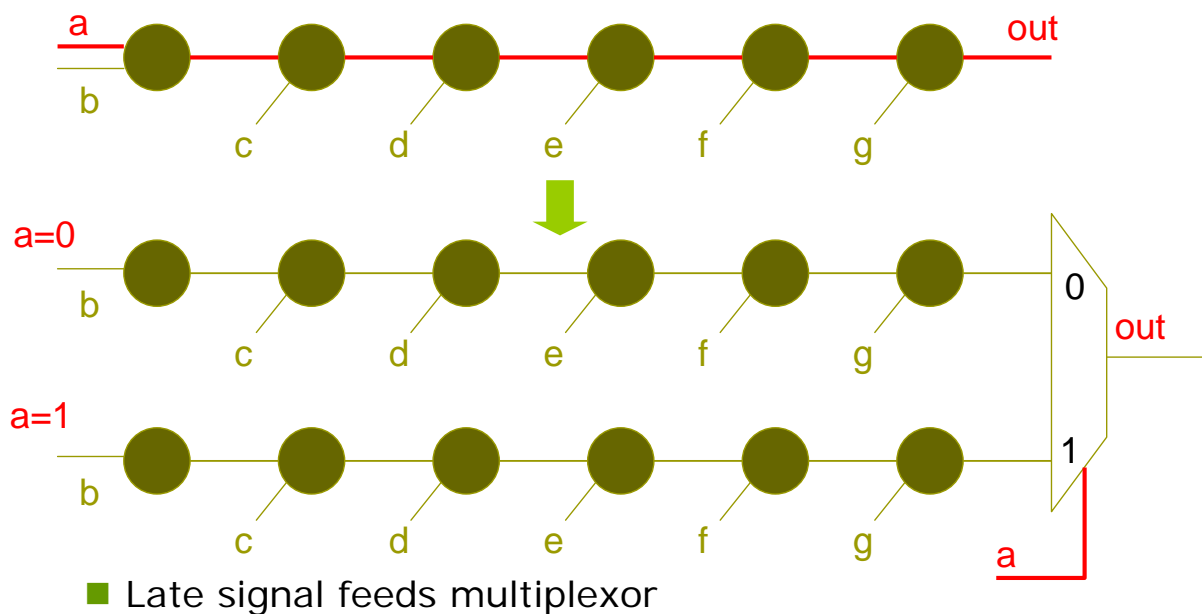
## False path removal by logic duplication



47

# Timing Optimization

## Generalized select transform



48



# Timing Optimization

---

## □ Summary

- There are various methods for delay optimization at different synthesis stages
  - No single technique dominates
  - Most techniques (except false path removal by logic duplication) ignore **false paths** when assessing the delay and critical regions