VLSI Testing
積體電路測試

Testability Measure

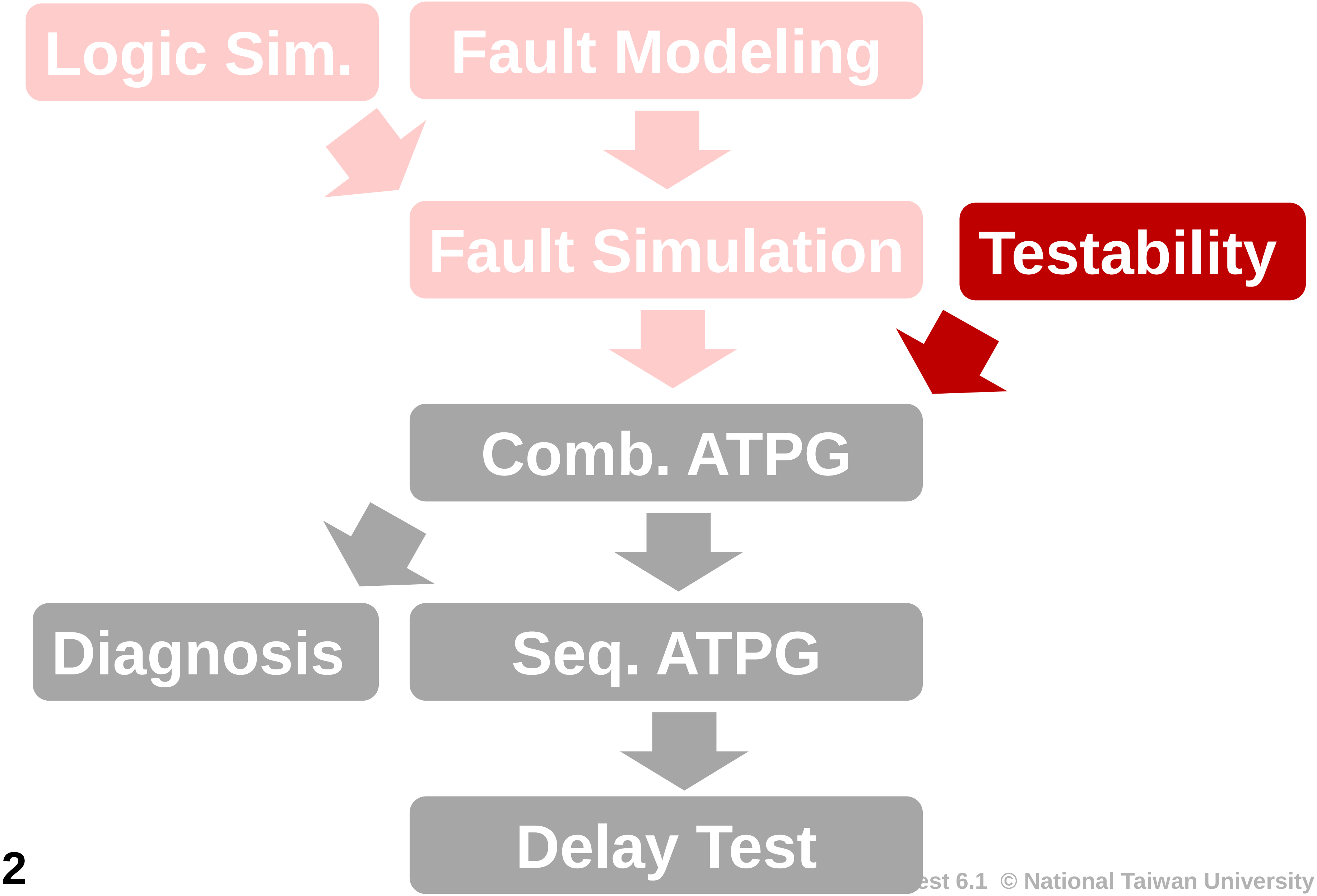
Professor James Chien-Mo Li 李建模

Lab. of Dependable Systems

Graduate Institute of Electronics Engineering
National Taiwan University

* Some pictures are courtesy of Prof. Jiun-Lang Huang, NTU

Course Roadmap (EDA Topics)



Logic Sim.

Fault Modeling

Fault Simulation

Testability

Comb. ATPG

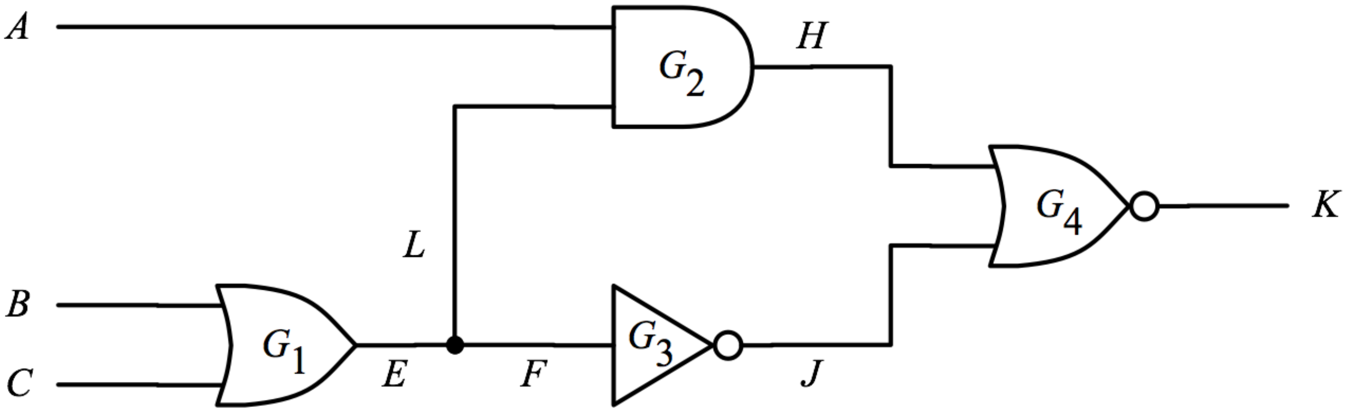
Diagnosis

Seq. ATPG

Delay Test

Motivating Problem

- **You report fault coverage of test set to your manager. Your manager is not very happy about FC number. He asked you: which faults are so difficult to detect?**



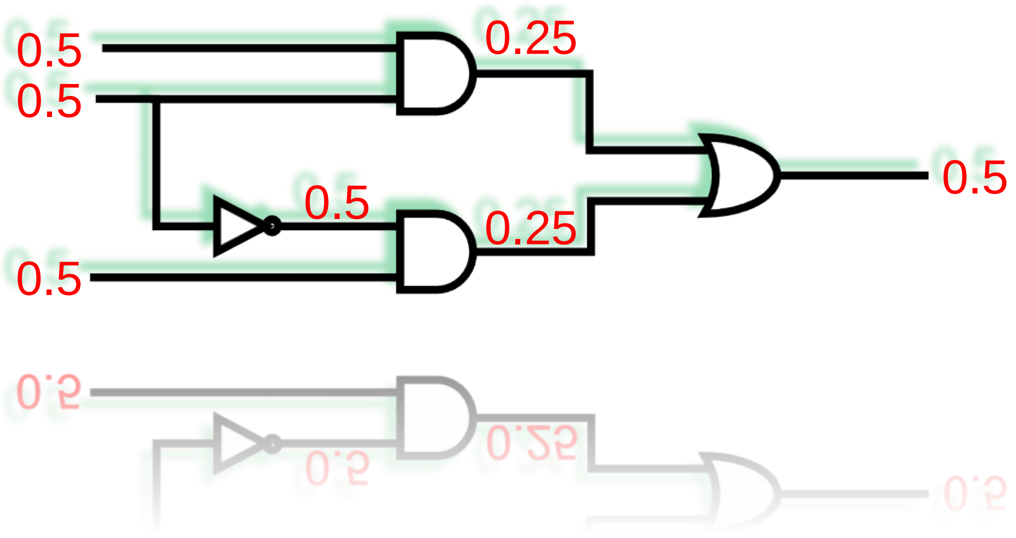
Why Am I Learning This?

- Testability measure helps to
  - Make smart decision in ATPG
  - More testable designs

> ***"Measurement is the first step that leads to control and eventually to improvement. "***
>
> ***(H. James Harrington)***

Testability Measure

- Introduction
- SCOAP (1979)
- COP (1984)
- High-level Testability Measures
- Conclusion



Testability Measures

- What is testability measure?
  - Metric to measure degree of difficulty to test a circuit

- Two important components:
  - *Controllability*
    - degree of difficulty to control a logic signal to 0 or 1
  - *Observability*
    - degree of difficulty to observe the logic value of a signal

Testability Measures Controllability and Observability

Testability Analysis

- What is *Testability Analysis*?
  - Calculate testability measures for a given circuit
- Why testability analysis?
  - 1. Help ATPG make smart decision
  - 2. Insert DFT circuits to improve controllability and observability
- When testability analysis?
  - In *preprocess* stage of ATPG or DFT insertion
- Requirements of testability analysis
  - Should run very fast
  - Sometimes , accuracy is not very important

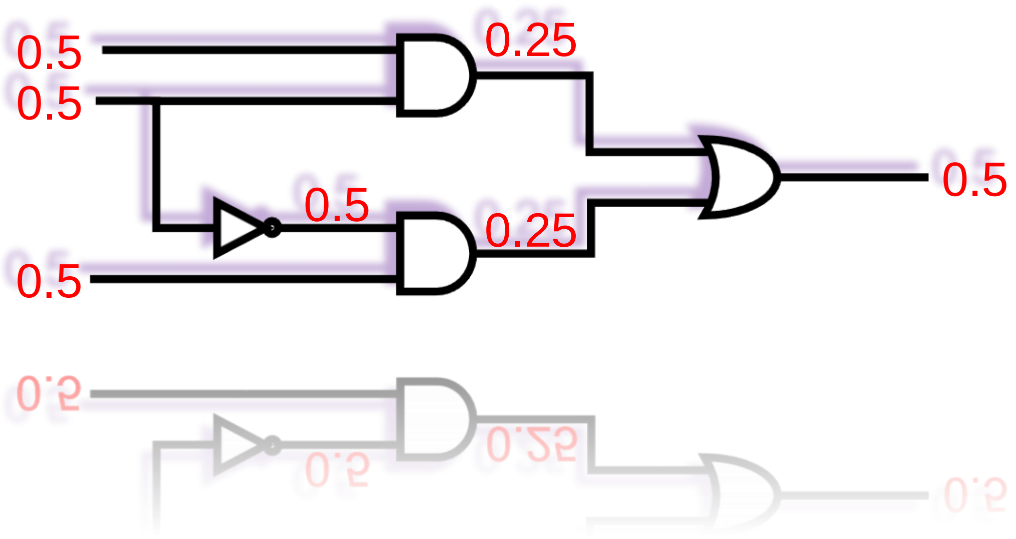
Categories of Testability Analysis

- 1. *Topology-based* analysis
  - Only analyzes structure of circuit. No test vectors are given.
  - Example: SCOAP
- 2. *Probability-based* analysis
  - Uses *signal probability* to estimate the testability
  - Example: COP
- 3. *High-level* analysis
  - Performs testability analysis before synthesis
  - Example: RTL testability analysis

- 4. *Simulation-base* analysis (not in lecture)
  - Apply input patterns,
  - Perform simulation and estimate testability

Testability Measure

- Introduction
- SCOAP (1979)
  - Combinational
  - Sequential
- COP (1984)
- High-level testability measures



SCOAP [Goldstein 1979]

- *Sandia Controllability Observability Analysis Program**
- SCOAP computes 6 numbers for each node *N*

*Sandia is name of research lab.

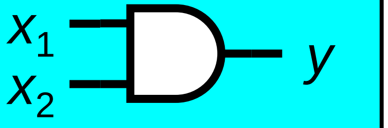
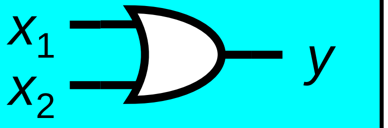
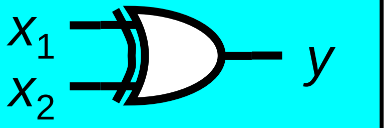
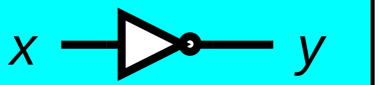
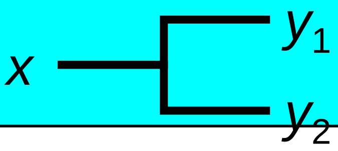
	0-controllability	1-controllability	Observability
Combinational	$CC^0(N)$	$CC^1(N)$	$CO(N)$
Sequential	$SC^0(N)$	$SC^1(N)$	$SO(N)$

Combinational Controllability

- ***$CC^0(N)$, $CC^1(N)$***
  - **minimum number of combinational PI assignments and logic levels required to control a 0 or a 1 on node *N***
  - ***Smaller* number, *easier* to control**
- **How to calculate CC?**
  - **CC(PI) = 1.**
  - **From PI to PO. Add 1 to account for logic level**
- **Gate propagation rules:**
  - **If only one input controls gate output:**
    - **CC(gate_output) = min { CC(gate_input) } + 1**
  - **If all inputs needed to set gate output:**
    - **CC(gate_output) = Σ CC(gate_input) + 1**
  - **CC(branches)=CC(stem)**

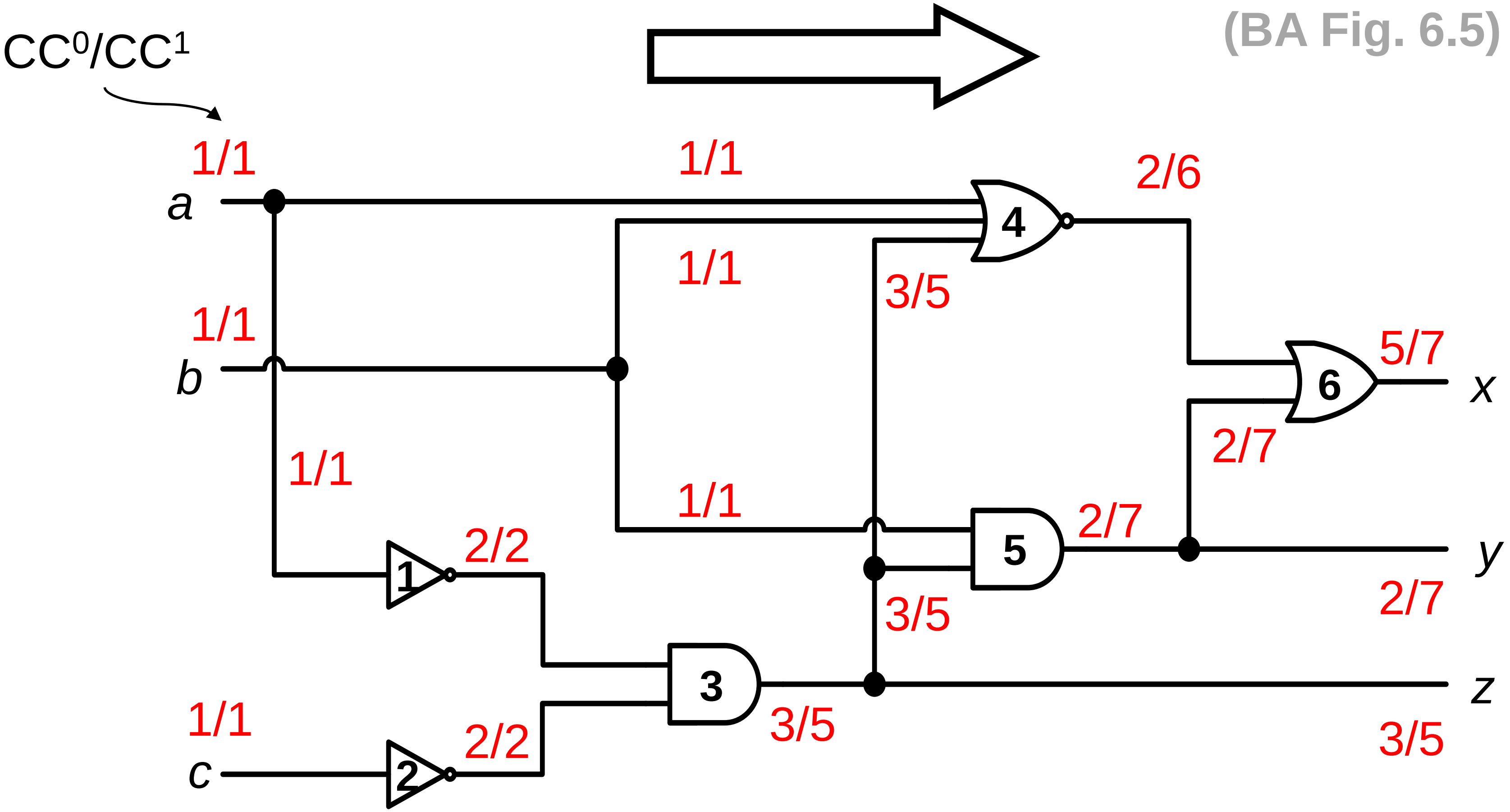
Smaller CC is Better

$CC^0(N)$ & $CC^1(N)$

	$CC^0(y)$	$CC^1(y)$
Primary inputs	1	1
	$\min[CC^0(x_1),CC^0(x_2)] + 1$	$CC^1(x_1) + CC^1(x_2) + 1$
	$CC^0(x_1) + CC^0(x_2) + 1$	$\min[CC^1(x_1),CC^1(x_2)] + 1$
	$\min[CC^0(x_1) + CC^0(x_2),$ $CC^1(x_1) + CC^1(x_2)] + 1$	$\min[CC^0(x_1) + CC^1(x_2),$ $CC^1(x_1) + CC^0(x_2)] + 1$
	$CC^1(x) + 1$	$CC^0(x) + 1$
	$CC^0(y_1) = CC^0(y_2)= CC^0(x)$	$CC^1(y_1) = CC^1(y_2)= CC^1(x)$

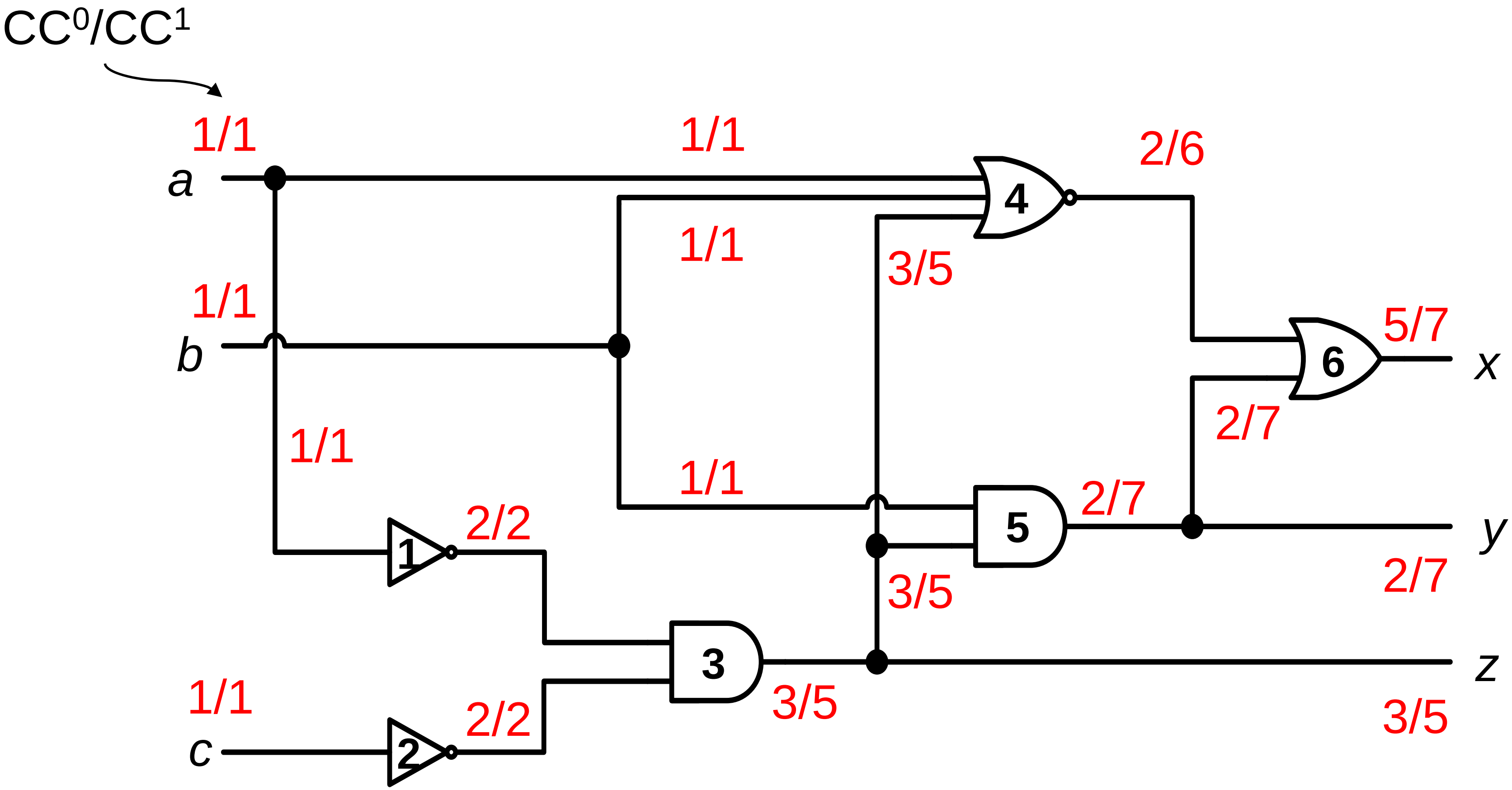
An Example – Controllability

- **Forward: From PI to PO**



Quiz

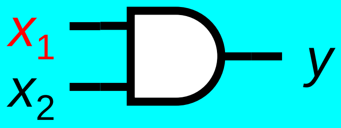
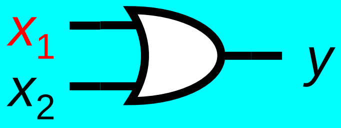
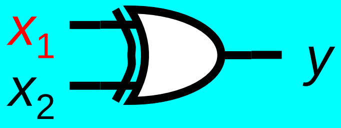
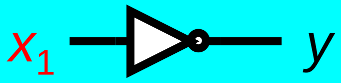
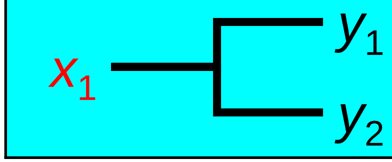
Q: how to control *y* to 0? How to control *y* to 1?



Combinational Observability

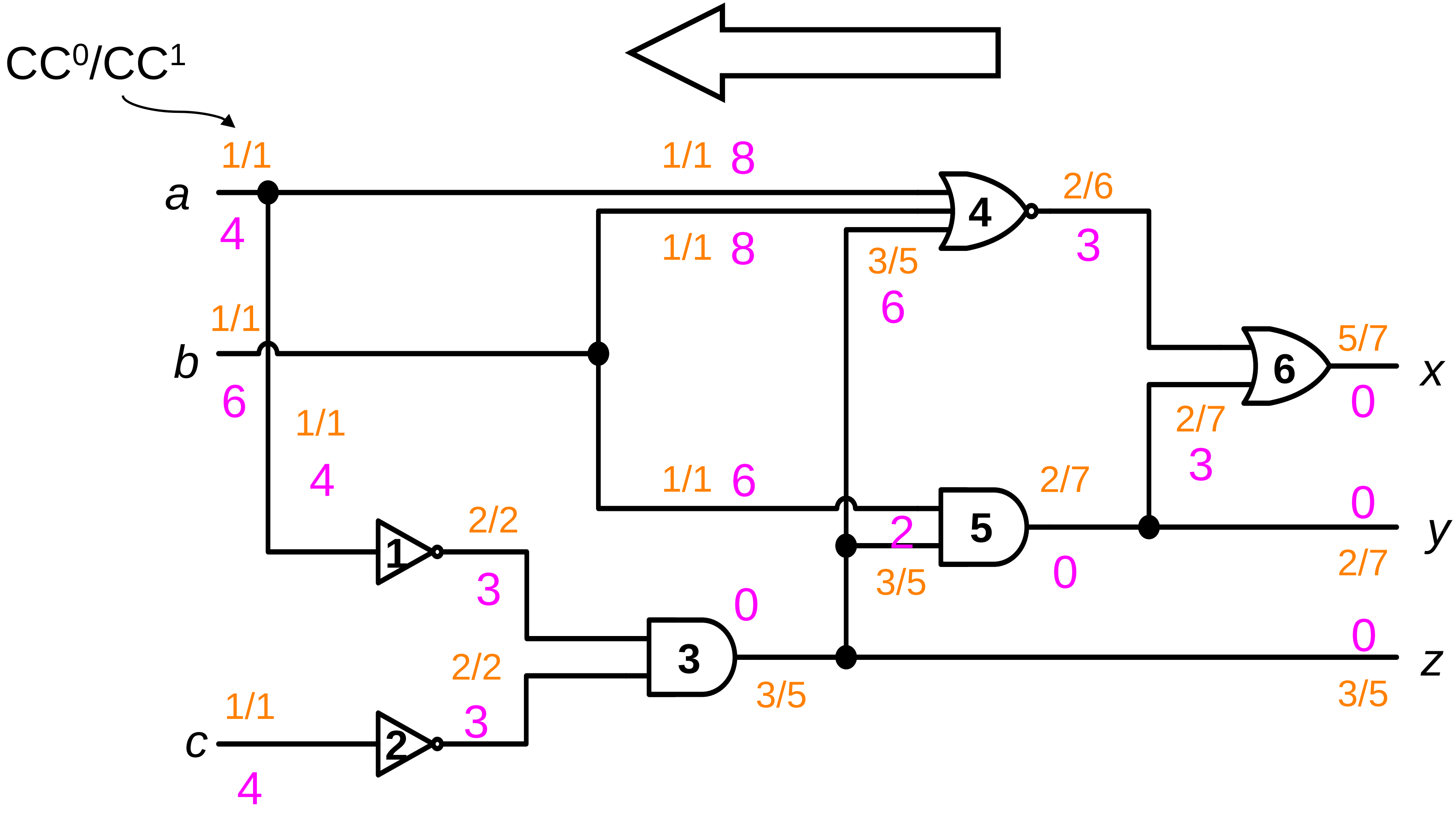
- ***CO(N)***
  - minimum number of combinational PI assignments and logic levels required to propagate logical value on node *N* to PO
  - ***Smaller*** number, ***easier*** to observe
- How to calculate CO(N)?
  - CO(PO)=0
  - From POs to PIs, add 1 to account for logic level
- CO(gate_input)= Σ
  - (1) CO(gate_output)
  - (2) CC(setting all other inputs to non-controlling value)
  - (3) + 1 for logic level
- How about fanout stem? Assume they are independent
  - CO(stem) = min { CO(branches) }

CO(N)

	CO(x_1)
Primary outputs	0
	CO(y) + CC^1(x_2) + 1
	CO(y) + CC^0(x_2) + 1
	CO(y) + min[CC^0(x_2) ,CC^1(x_2)] + 1
	CO(y) + 1
	min[CO(y_1),CO(y_2)]

An Example – Observability

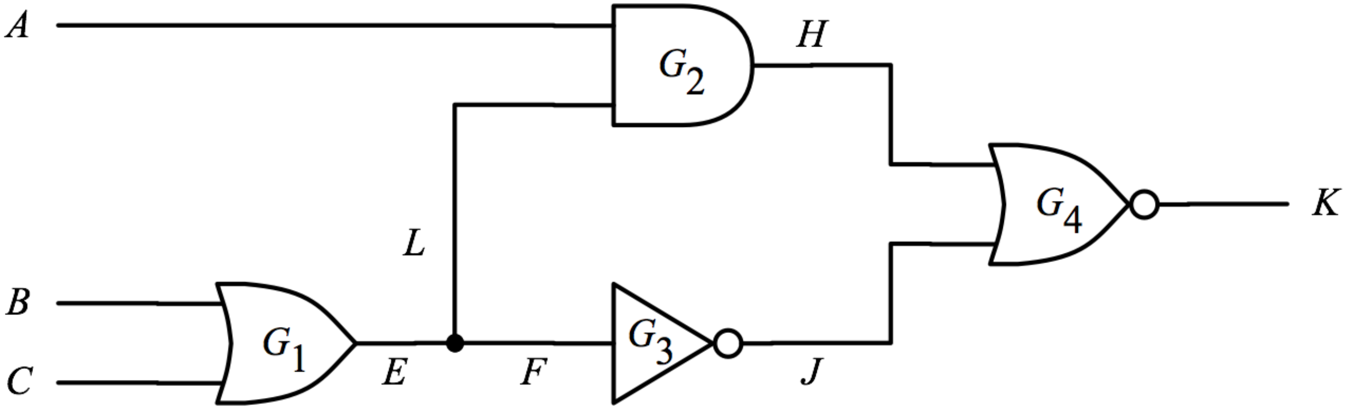
- **Backward: From PO to PI**



Quiz

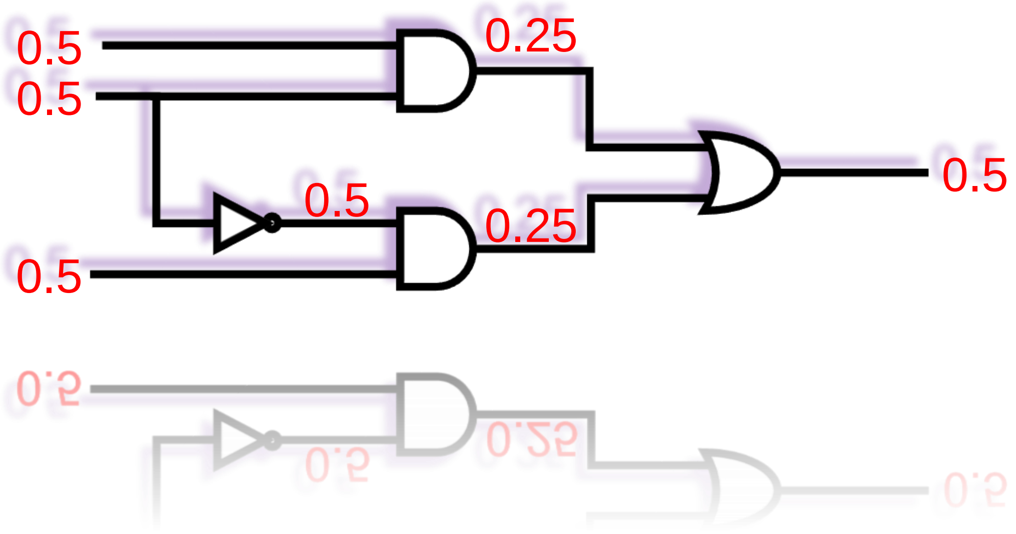
Q: Please analyze combinational SCOAP.
A:

	A	B	C	E	F	L	H	J	K
CC^0									
CC^1									
CO									



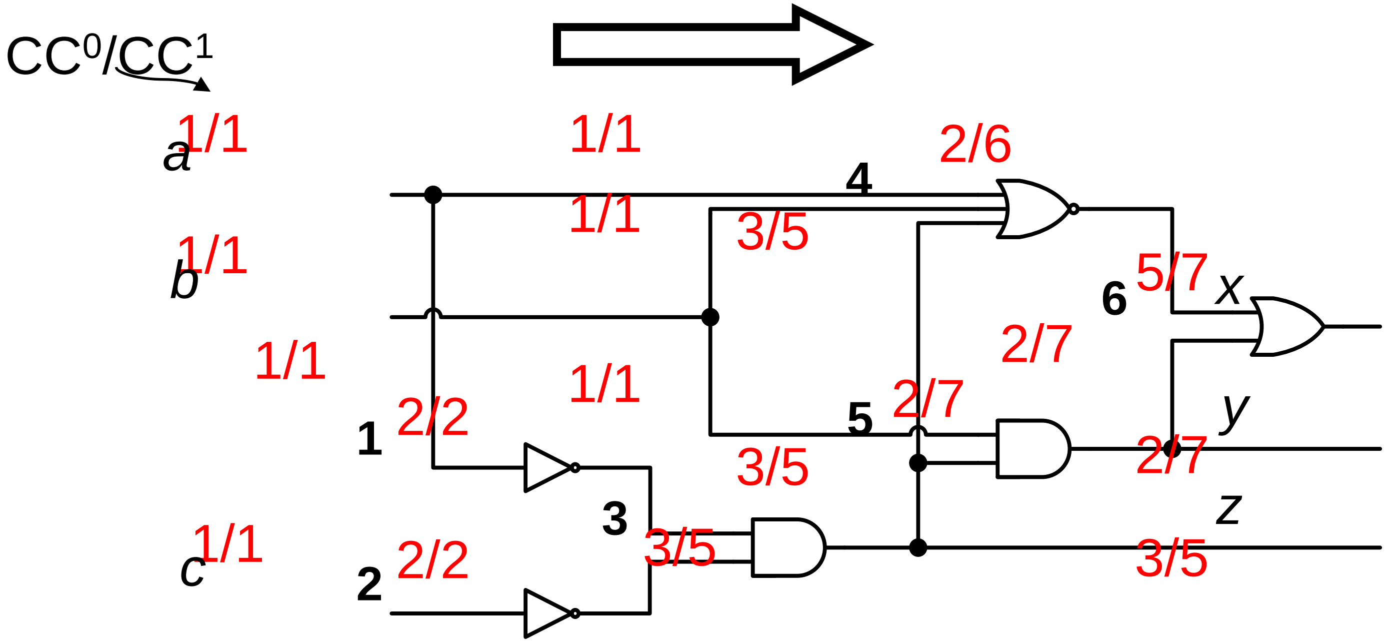
Summary

- **Introduction**
  - **Metric to measure degree of difficulty to test a circuit**
  - **Helps 1. ATPG 2. DfT**
- **SCOAP: Combinational**
  - **CC^0, CC^1, CO**



FFT

- **Q: Testability should be done very quickly. What is time complexity to calculate CC and CO?**



CC0/CC1
a 1/1
b 1/1
c 1/1
1/1
1/1
1/1
1/1
1 2/2
2 2/2
3 3/5
4 2/6
3/5
5 2/7
3/5
6 5/7
x
2/7
y 2/7
z 3/5