第2讲 可测试性设计(1)

李晓维

中科院计算技术研究所

Email: lxw@ict.ac.cn

http://people.ucas.edu.cn/~lxw

Chapter 2

Design for Testability

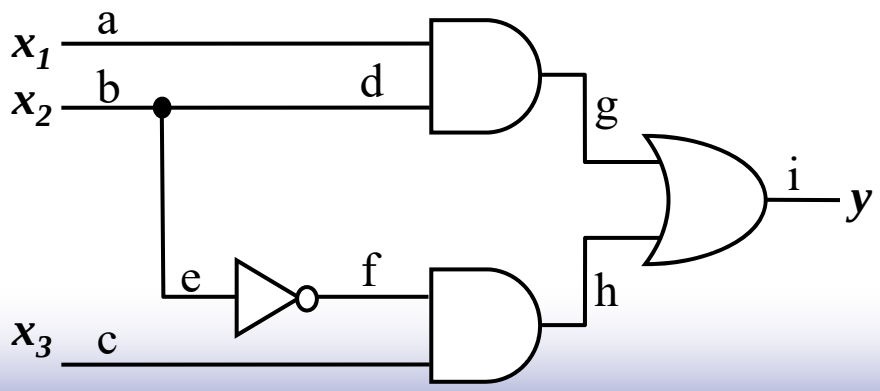
3个引导问题

- 组合电路和时序电路有什么区别？
- 如何测试一个stuck-at故障？
- 尝试分析什么是“Testability”，并对比组合电路和时序电路的Testability

Stuck-at Faults

- Any line can be
  - Stuck-at-0 (SA0)
  - Stuck-at-1 (SA1)

  # fault types: k=2
- Example circuit:
  - # fault sites: n=9
  - # single faults =2×9=18



Truth table for fault-free behavior and behavior of all possible stuck-at faults

$x_1x_2x_3$	000	001	010	011	100	101	110	111
y	0	1	0	0	0	1	1	1
a SA0	0	1	0	0	0	1	0	0
a SA1	0	1	1	1	0	1	1	1
b SA0	0	1	0	1	0	1	0	1
b SA1	0	0	0	0	1	1	1	1
c SA0	0	0	0	0	0	0	1	1
c SA1	1	1	0	0	1	1	1	1
d SA0	0	1	0	0	0	1	0	0
d SA1	0	1	0	0	1	1	1	1
e SA0	0	1	0	1	0	1	1	1
e SA1	0	0	0	0	0	0	1	1
f SA0	0	0	0	0	0	0	1	1
f SA1	0	1	0	1	0	1	1	1
g SA0	0	1	0	0	0	1	0	0
g SA1	1	1	1	1	1	1	1	1
h SA0	0	0	0	0	0	0	1	1
h SA1	1	1	1	1	1	1	1	1
i SA0	0	0	0	0	0	0	0	0
i SA1	1	1	1	1	1	1	1	1

Design For Testability - contents

- Introduction
- Testability Analysis
- Design for Testability Basics
- Scan Cell Designs
- Scan Architectures
- Scan Design Rules
- Scan Design Flow
- Special-Purpose Scan Designs
- RTL Design for Testability
- Concluding Remarks

Introduction

History

- During early years, design and test were separate
  - The final quality of the test was determined by keeping track of the number of defective parts shipped to the customer
  - Defective parts per million (PPM) shipped was a final test score
  - This approach worked well for small-scale integrated circuits
- During 1980s, fault simulation was used
  - Failed to improve the circuit's fault coverage beyond 80%
- Increased test cost and decreased test quality lead to DFT engineering

Introduction

History

- Various testability measures & *ad hoc* testability enhancement methods
  - To improve the testability of a design
  - To ease sequential ATPG (automatic test pattern generation)
  - Still quite difficult to reach more than 90% fault coverage
- Structured DFT
  - To conquer the difficulties in controlling and observing the internal states of sequential circuits
  - Scan design is the most popular structured DFT approach
- Design for testability (DFT) has migration recently
  - From gate level to register-transfer level (RTL)

Testability Analysis

- Testability:
  - A relative measure of the effort or cost of testing a logic circuit
- Testability Analysis:
  - The process of assessing the testability of a logic circuit
- Testability Analysis Techniques:
  - Topology-based Testability Analysis
    - SCOAP - *Sandia Controllability/Observability Analysis Program*
    - *Probability-based testability analysis*
  - Simulation-based Testability Analysis

Testability Analysis – SCOAP

- *Controllability*
  - Reflects the difficulty of setting a signal line to a required logic value from primary inputs
- *Observability*
  - Reflects the difficulty of propagating the logic value of the signal line to primary outputs

Testability Analysis – SCOAP

- calculates six numerical values for each signals in a logic circuit
  - CC0(*s*): combinational 0-controllability of *s*
  - CC1(*s*): combinational 1-controllability of *s*
  - CO(*s*): combinational observability of *s*
  - SC0(*s*): sequential 0-controllability of *s*
  - SC1(*s*): sequential 1-controllability of *s*
  - SO(*s*): sequential observability of *s*

Testability Analysis – SCOAP

- The value of controllability measures range between 1 to infinite
- The value of observability measures range between 0 to infinite
  - The CC0 and CC1 values of a primary input are set to 1
  - The SC0 and SC1 values of a primary input are set to 0
  - The CO and SO values of a primary output are set to 0

Testability Analysis - SCOAP

Combinational Controllability Calculation Rules

	0-controllability (Primary input, output, branch)	**1-controllability** (Primary input, output, branch)
Primary Input	1	1
AND	*min* {input 0-controllabilities} + 1	Σ (input 1-controllabilities) + 1
OR	Σ (input 0-controllabilities) + 1	*min* {input 1-controllabilities} + 1
NOT	Input 1-controllability + 1	Input 0-controllability + 1
NAND	Σ (input 1-controllabilities) + 1	*min* {input 0-controllabilities} + 1
NOR	*min* {input 1-controllabilities} + 1	Σ (input 0-controllabilities) + 1
BUFFER	Input 0-controllability + 1	Input 1-controllability + 1
XOR	*min* {CC1(a)+CC1(b), CC0(a)+CC0(b)} + 1	*min* {CC1(a)+CC0(b), CC0(a)+CC1(b)} + 1
XNOR	*min* {CC1(a)+CC0(b), CC0(a)+CC1(b)} + 1	*min* {CC1(a)+CC1(b), CC0(a)+CC0(b)} + 1
Branch	Stem 0-controllability	Stem 1-controllability

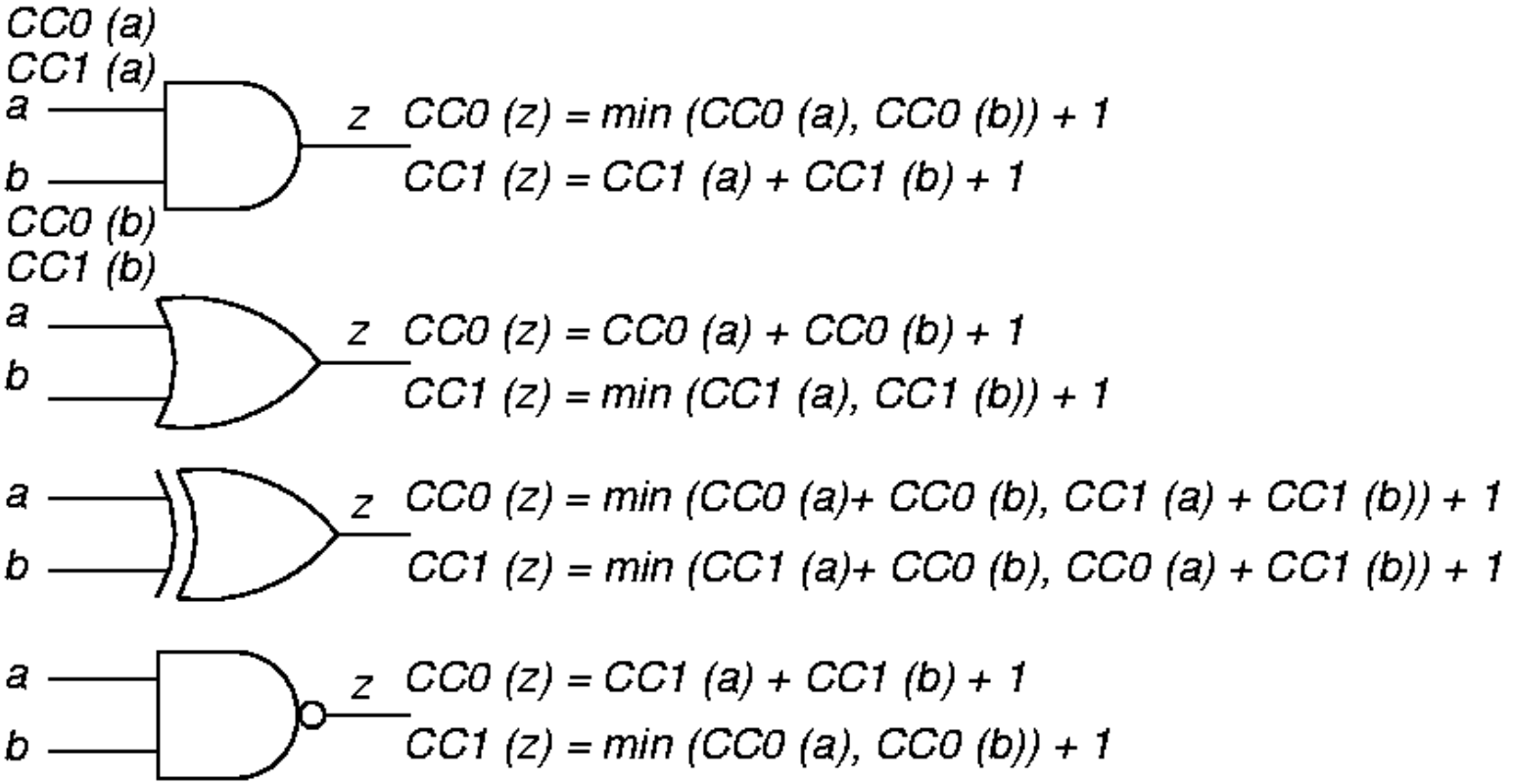
Testability Analysis - SCOAP

Combinational Controllability Observability Rules

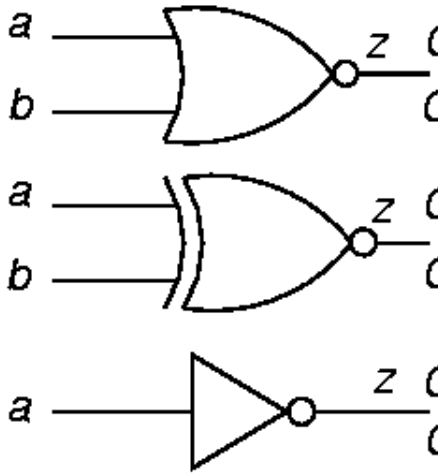
	Observability (Primary output, input, stem)
Primary Output	0
AND / NAND	Σ (output observability, 1-controllabilities of other inputs) + 1
OR / NOR	Σ (output observability, 0-controllabilities of other inputs) + 1
NOT / BUFFER	Output observability + 1
XOR / XNOR	a: Σ (output observability, *min* {CC0(b), CC1(b)}) + 1 b: Σ (output observability, *min* {CC0(a), CC1(a)}) + 1
Stem	*min* {branch observabilities}

a, b: inputs of an XOR or XNOR gate

Controllability Examples



More Controllability Examples



$CC0\ (z) = min\ (CC1\ (a), CC1\ (b)) + 1$

$CC1\ (z) = CC0\ (a) + CC0\ (b) + 1$

$CC0\ (z) = min\ (CC1\ (a) + CC0\ (b), CC0\ (a) + CC1\ (b)) + 1$

$CC1\ (z) = min\ (CC0\ (a) + CC0\ (b), CC1\ (a) + CC1\ (b)) + 1$

$CC0\ (z) = CC1\ (a) + 1$

$CC1\ (z) = CC0\ (a) + 1$

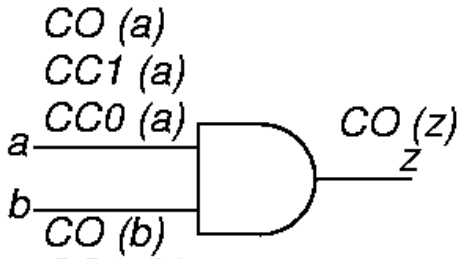
Observability Examples

To observe a gate input:

Observe output and make other input values non-controlling

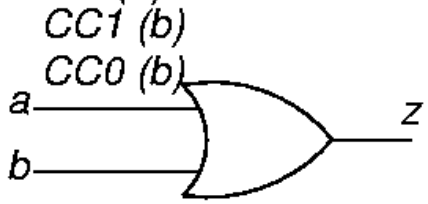
$CO(a) = CO(z) + CC1(b) + 1$

$CO(b) = CO(z) + CC1(a) + 1$



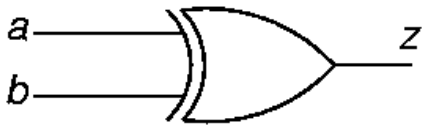
$CO(a) = CO(z) + CC0(b) + 1$

$CO(b) = CO(z) + CC0(a) + 1$



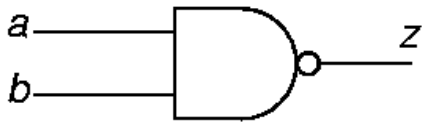
$CO(a) = CO(z) + \min(CC0(b), CC1(b)) + 1$

$CO(b) = CO(z) + \min(CC0(a), CC1(a)) + 1$



$CO(a) = CO(z) + CC1(b) + 1$

$CO(b) = CO(z) + CC1(a) + 1$



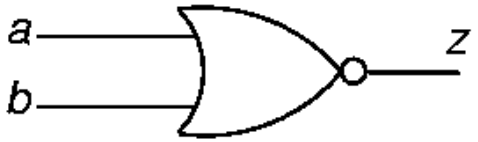
More Observability Examples

To observe a fanout stem:

Observe it through branch with best observability

$CO(a) = CO(z) + CC0(b) + 1$

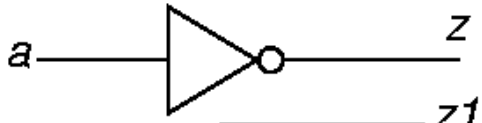
$CO(b) = CO(z) + CC0(a) + 1$



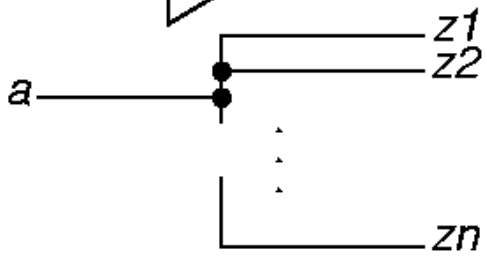
$CO(a) = CO(z) + \min(CC0(b), CC1(b)) + 1$

$CO(b) = CO(z) + \min(CC0(a), CC1(a)) + 1$

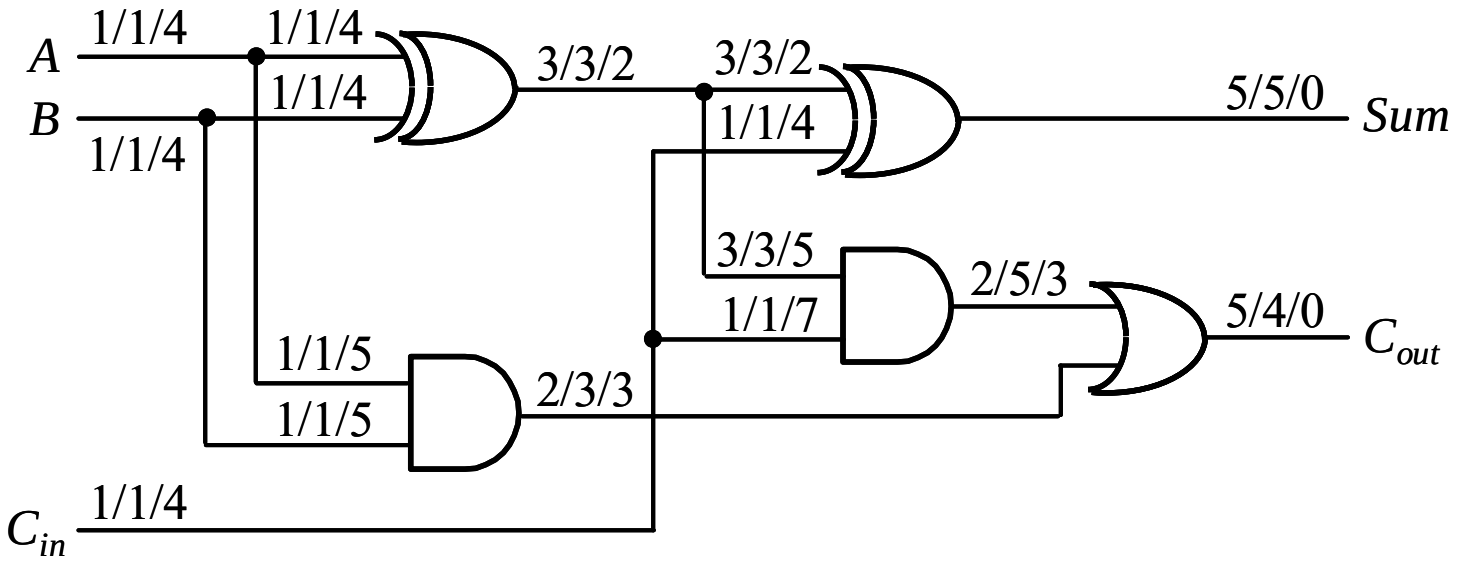
$CO(a) = CO(z) + 1$



$CO(a) = \min(CO(z1), CO(z2), \ldots, CO(zn))$



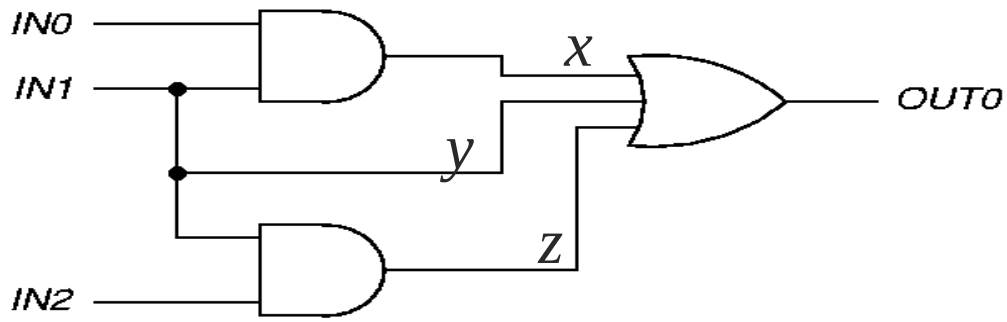
Testability Analysis – SCOAP



Example of Combinational SCOPA measures

$v1/v2/v3$ represents the signal's 0-controllability ($v1$), 1-controllability ($v2$), and observability ($v3$)

Exercise: Testability Analysis -SCOAP



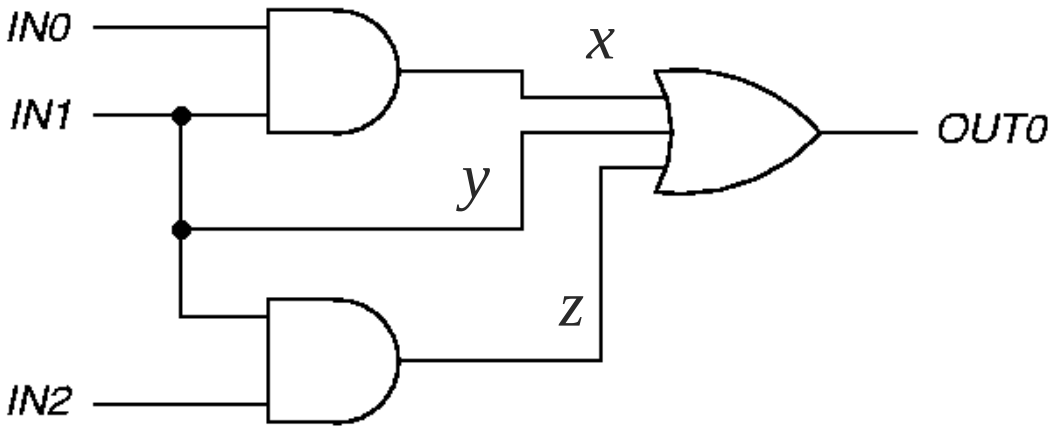
	0-controllability (Primary input, output, branch)	**1-controllability** (Primary input, output, branch)
Primary Input	1	1
AND	*min* {input 0-controllabilities} + 1	Σ (input 1-controllabilities) + 1
OR	Σ (input 0-controllabilities) + 1	*min* {input 1-controllabilities} + 1

	Observability (Primary output, input, stem)
Primary Output	0
AND / NAND	Σ (output observability, 1-controllabilities of other inputs) + 1
OR / NOR	Σ (output observability, 0-controllabilities of other inputs) + 1
Stem	*min* {branch observabilities}

Error Due to Stems & Reconvergent Fanouts

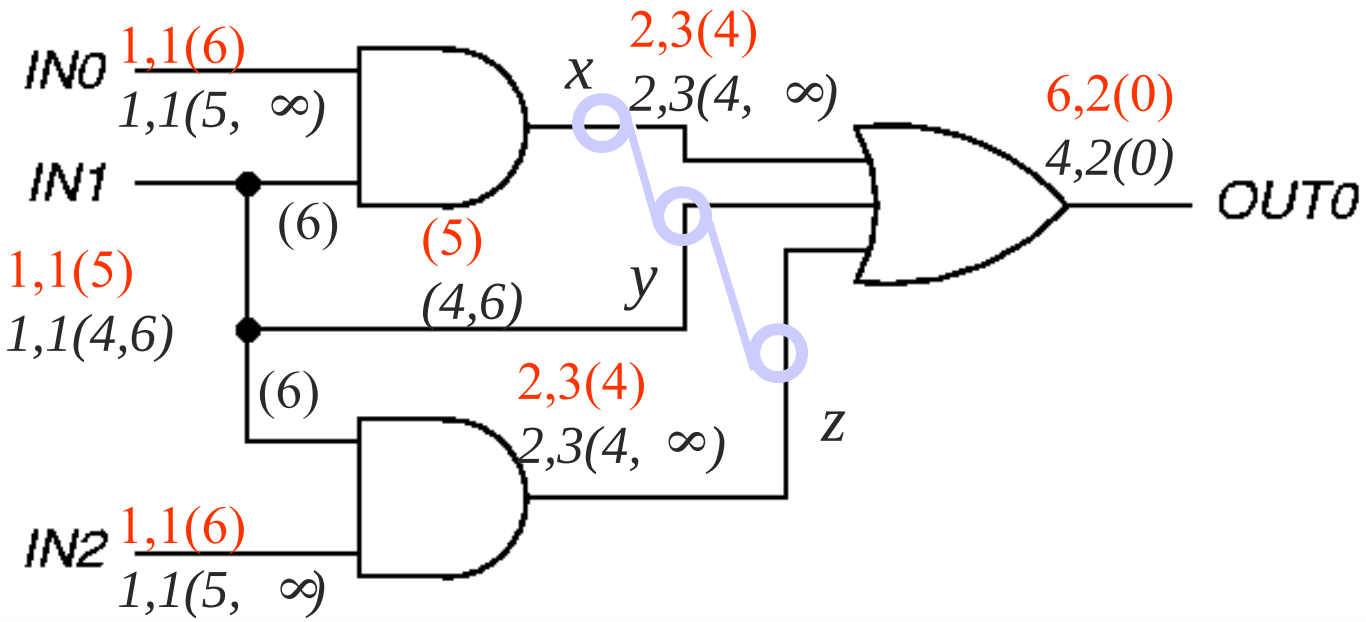
SCOAP measures wrongly assume that controlling or observing x, y, z are independent events

- *CC0 (x), CC0 (y), CC0 (z)* correlate
- *CC1 (x), CC1 (y), CC1 (z)* correlate
- *CO (x), CO (y), CO (z)* correlate



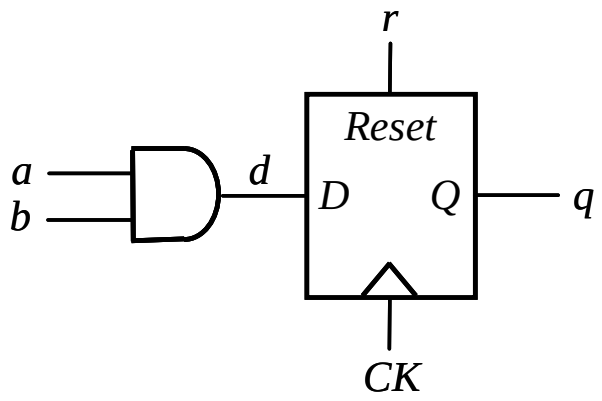
Correlation Error Example

- Exact computation of measures is NP-Complete and impractical
- Italicized measures show correct values – SCOAP measures are in red CC0,CC1 (CO)



Testability Analysis - SCOAP

Sequential Controllability and Observability Calculation



SCOAP sequential circuit example

The combinational and sequential controllability measures of signal *d*:

$$CC0(d) = \min\{CC0(a), CC0(b)\} + 1$$
$$SC0(d) = \min\{SC0(a), SC0(b)\}$$
$$CC1(d) = CC1(a) + CC1(b) + 1$$
$$SC1(d) = SC1(a) + SC1(b)$$

Testability Analysis - SCOAP

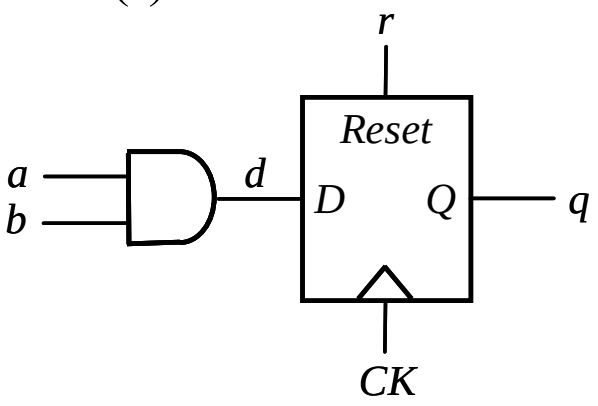
The combinational and sequential controllability and observability measures of *q:*

$$CC0(q) = \min\{CC0(d) + CC0(CK) + CC1(CK) + CC0(r), CC1(r) + CC0(CK)\}$$

$$SC0(q) = \min\{SC0(d) + SC0(CK) + SC1(CK) + SC0(r) + 1, SC1(r) + SC0(CK)\}$$

$$CC1(q) = CC1(d) + CC0(CK) + CC1(CK) + CC0(r)$$

$$SC1(q) = SC1(d) + SC0(CK) + SC1(CK) + SC0(r) + 1$$



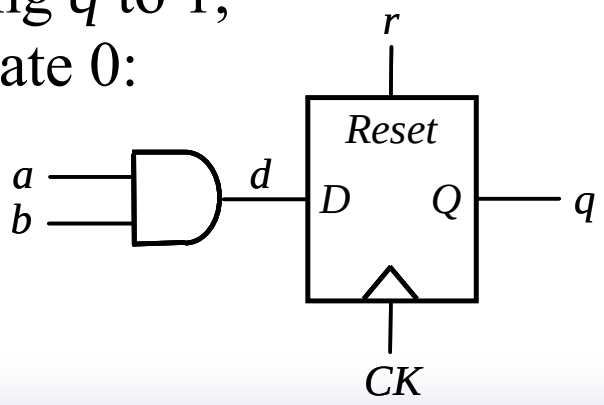
Testability Analysis - SCOAP

The data input *d* can be observed at *q* by holding the reset signal *r* at 0 and applying a rising clock edge to *CK*:

$$CO(d) = CO(q) + CC0(CK) + CC1(CK) + CC0(r)$$
$$SO(d) = SO(q) + SC0(CK) + SC1(CK) + SC0(r) + 1$$

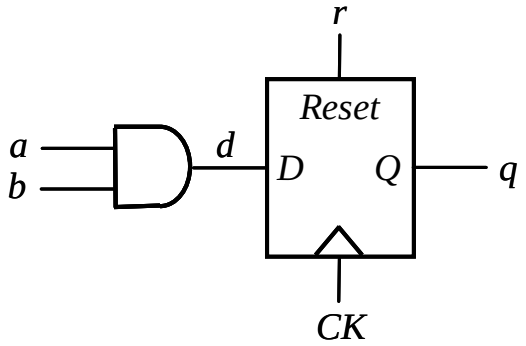
Signal *r* can be observed by first setting *q* to 1, and then holding *CK* at the inactive state 0:

$$CO(r) = CO(q) + CC1(q) + CC0(CK)$$
$$SO(r) = SO(q) + SC1(q) + SC0(CK)$$



Testability Analysis - SCOAP

- Two ways to indirectly observe the clock signal *CK* at *q*:
  - set *q* to 1, *r* to 0, *d* to 0, and apply a rising clock edge at *CK*
  - set both *q* and *r* to 0, *d* to 1, and apply a rising clock edge at *CK*



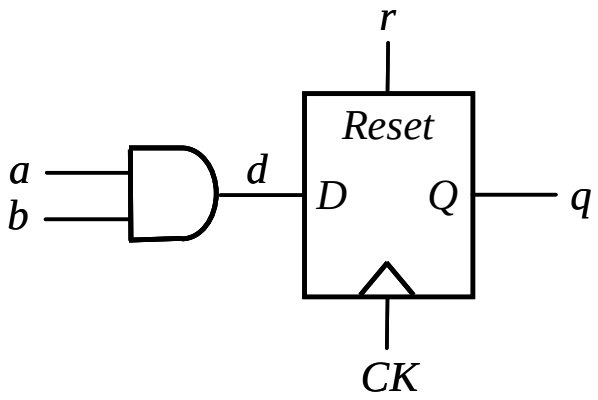
$$CO(CK) = CO(q) + CC0(CK) + CC1(CK) + CC0(r) + \min\{CC0(d) + CC1(q), CC1(d) + CC0(q)\}$$

$$SO(CK) = SO(q) + SC0(CK) + SC1(CK) + SC0(r) + \min\{SC0(d) + SC1(q), SC1(d) + SC0(q)\} + 1$$

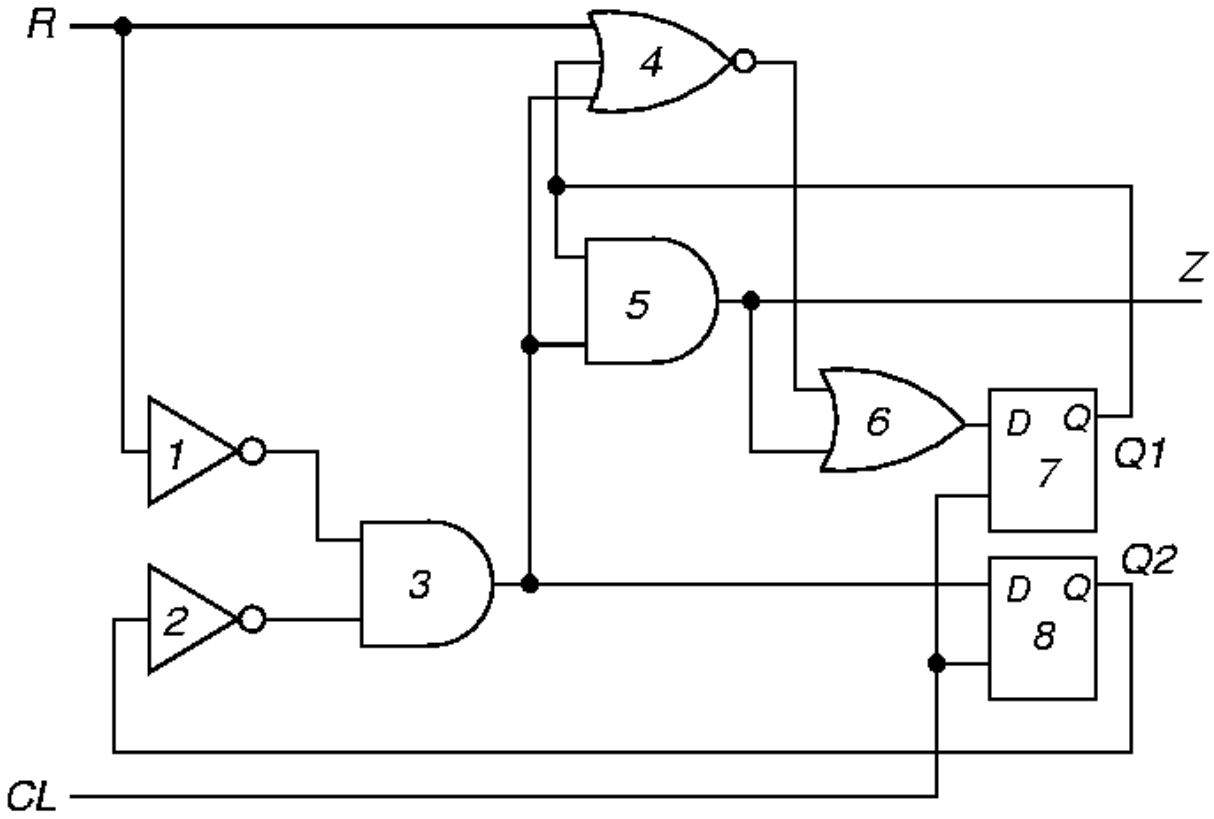
Testability Analysis - SCOAP

The combinational and sequential observability measures for both inputs *a* and *b*:

$$CO(a) = CO(d) + CC1(b) + 1$$
$$SO(a) = SO(d) + SC1(b)$$
$$CO(b) = CO(d) + CC1(a) + 1$$
$$SO(b) = SO(d) + SC1(a)$$



Sequential Example

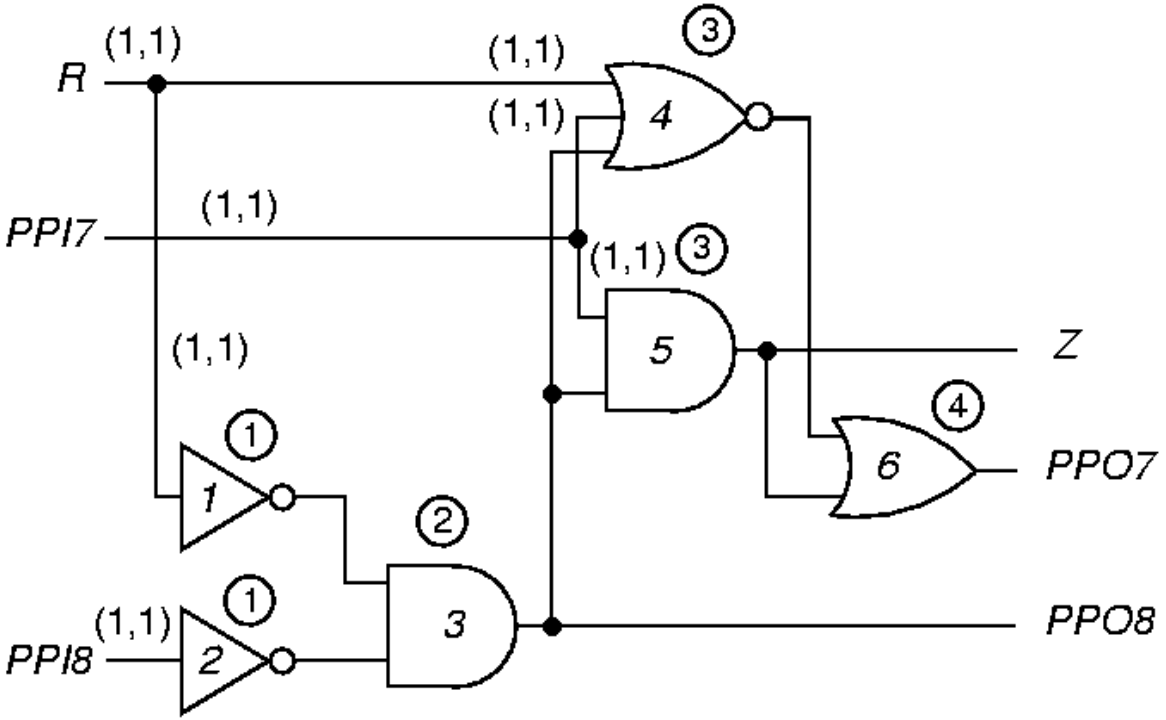


Levelization Algorithm

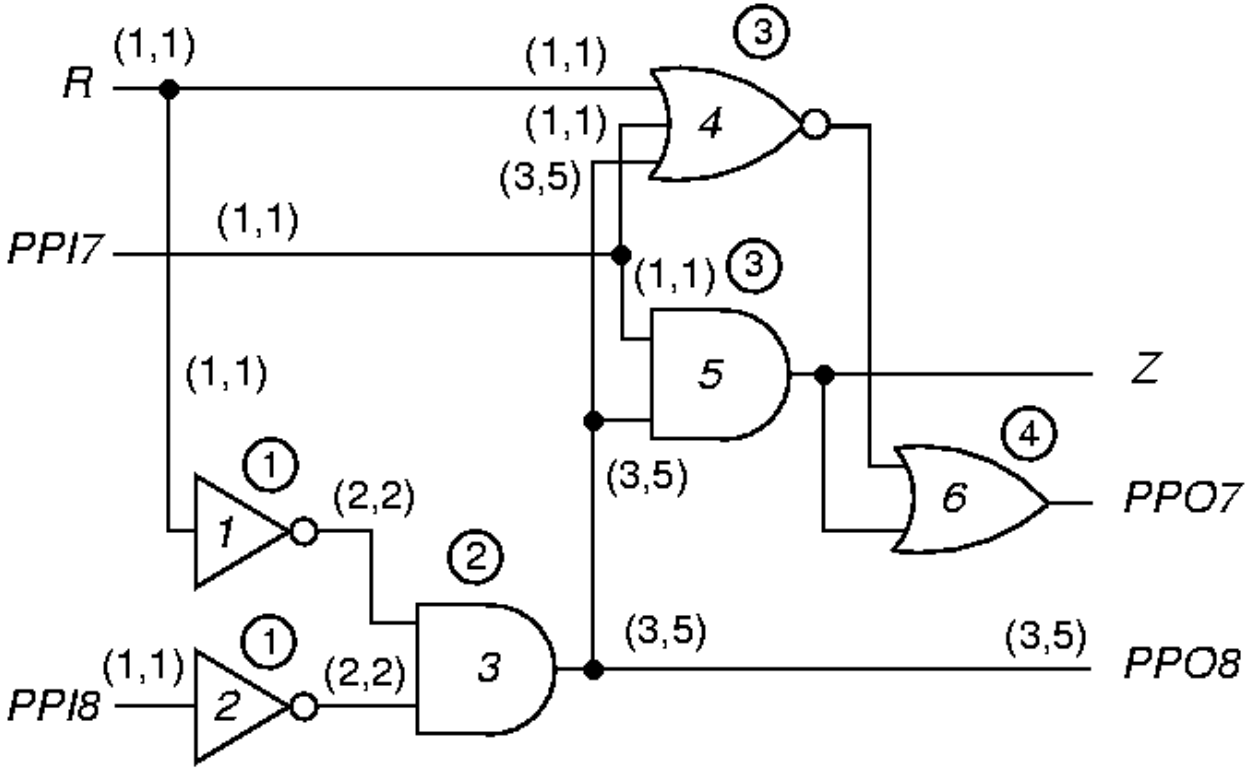
- Label each gate with max # of logic levels from primary inputs or with max # of logic levels from primary output
- Assign level # 0 to all *primary inputs* (PIs)
- For each PI fanout:
  - Label that line with the PI level number, &
  - Queue logic gate driven by that fanout
- While queue is not empty:
  - Dequeue next logic gate
  - If all gate inputs have level #'s, label the gate with the maximum of them + 1;
  - Else, requeue the gate

Controllability Through Level 1

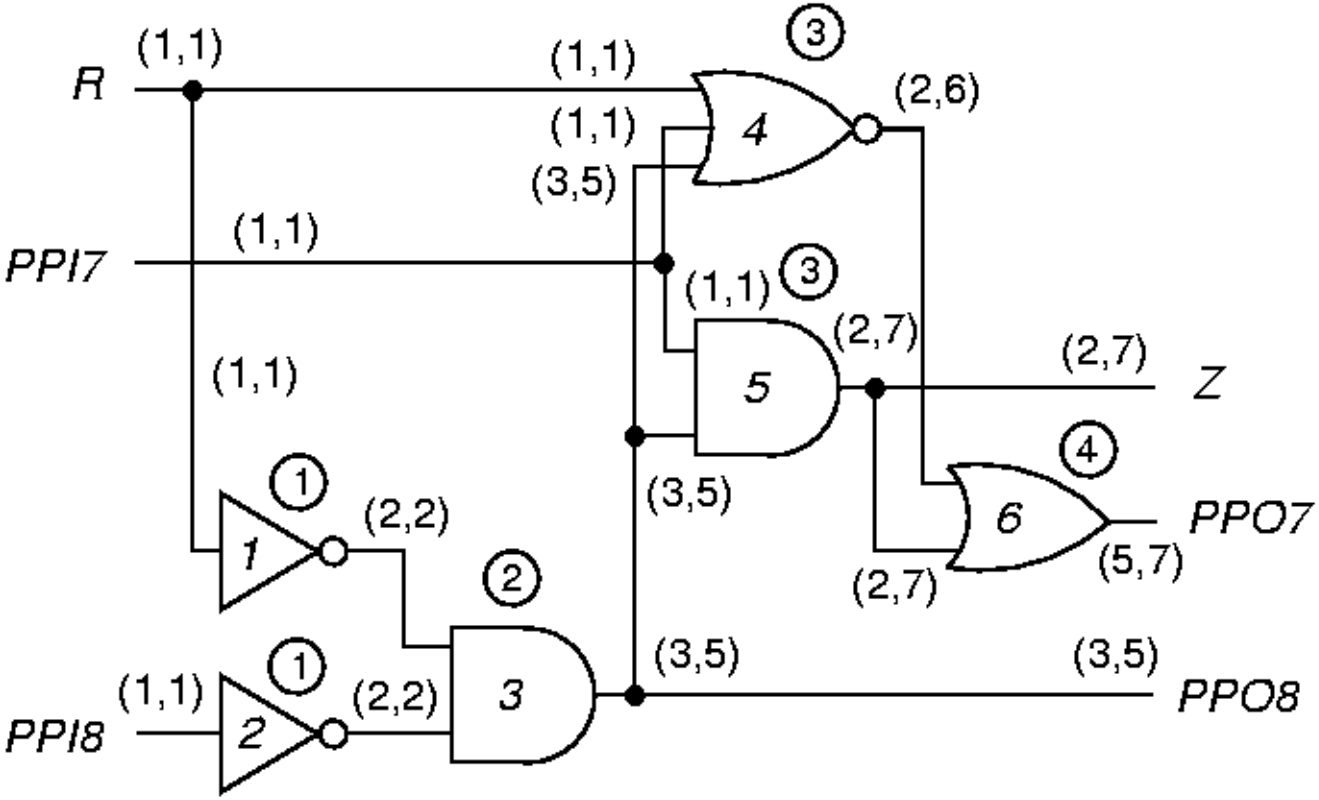
Circled numbers give level number. (CC0, CC1)



Controllability Through Level 2



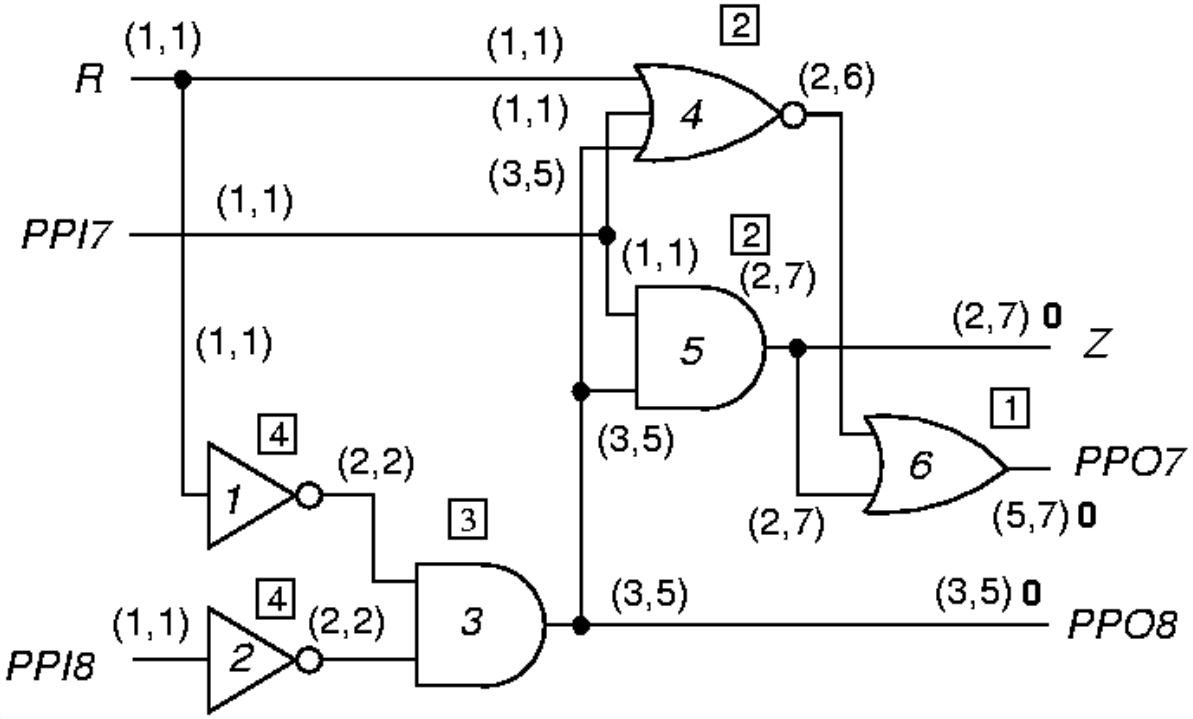
Final Combinational Controllability



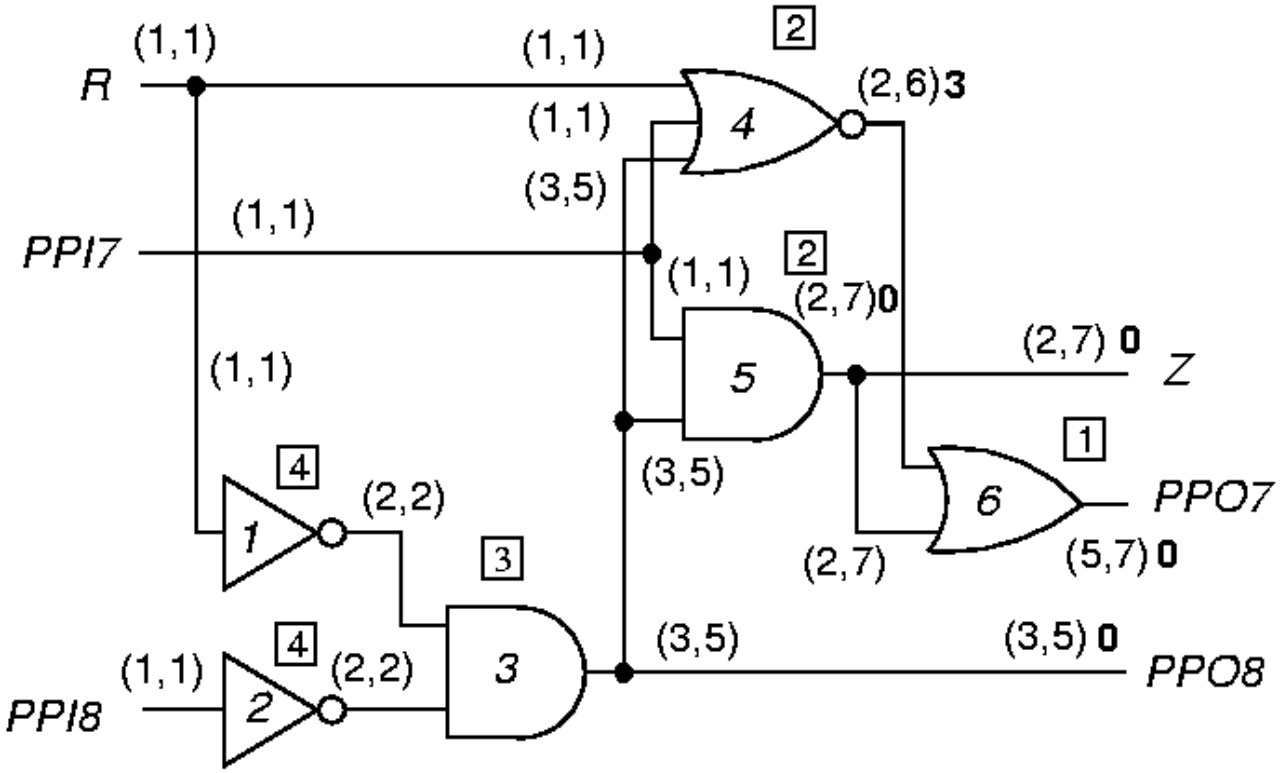
Combinational Observability for Level 1

Number in square box is level from *primary outputs* (POs).

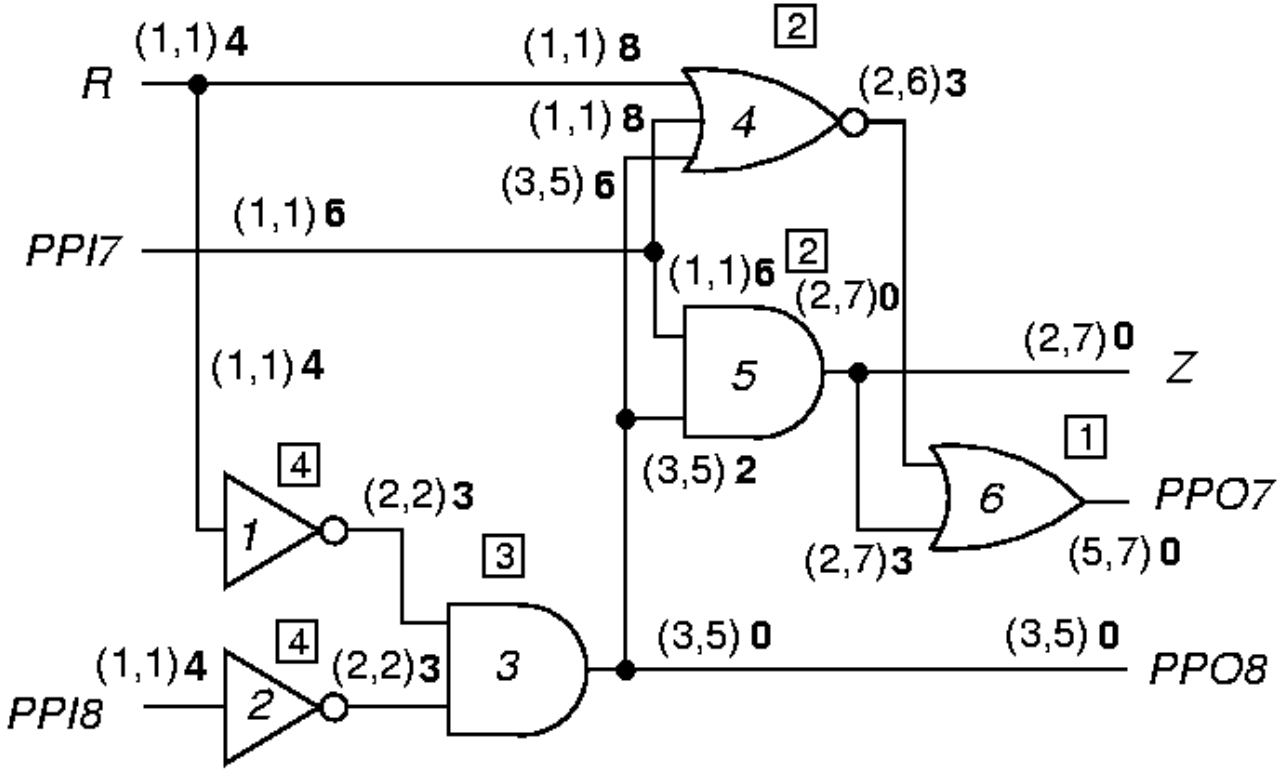
(CC0, CC1) CO



Combinational Observabilities for Level 2



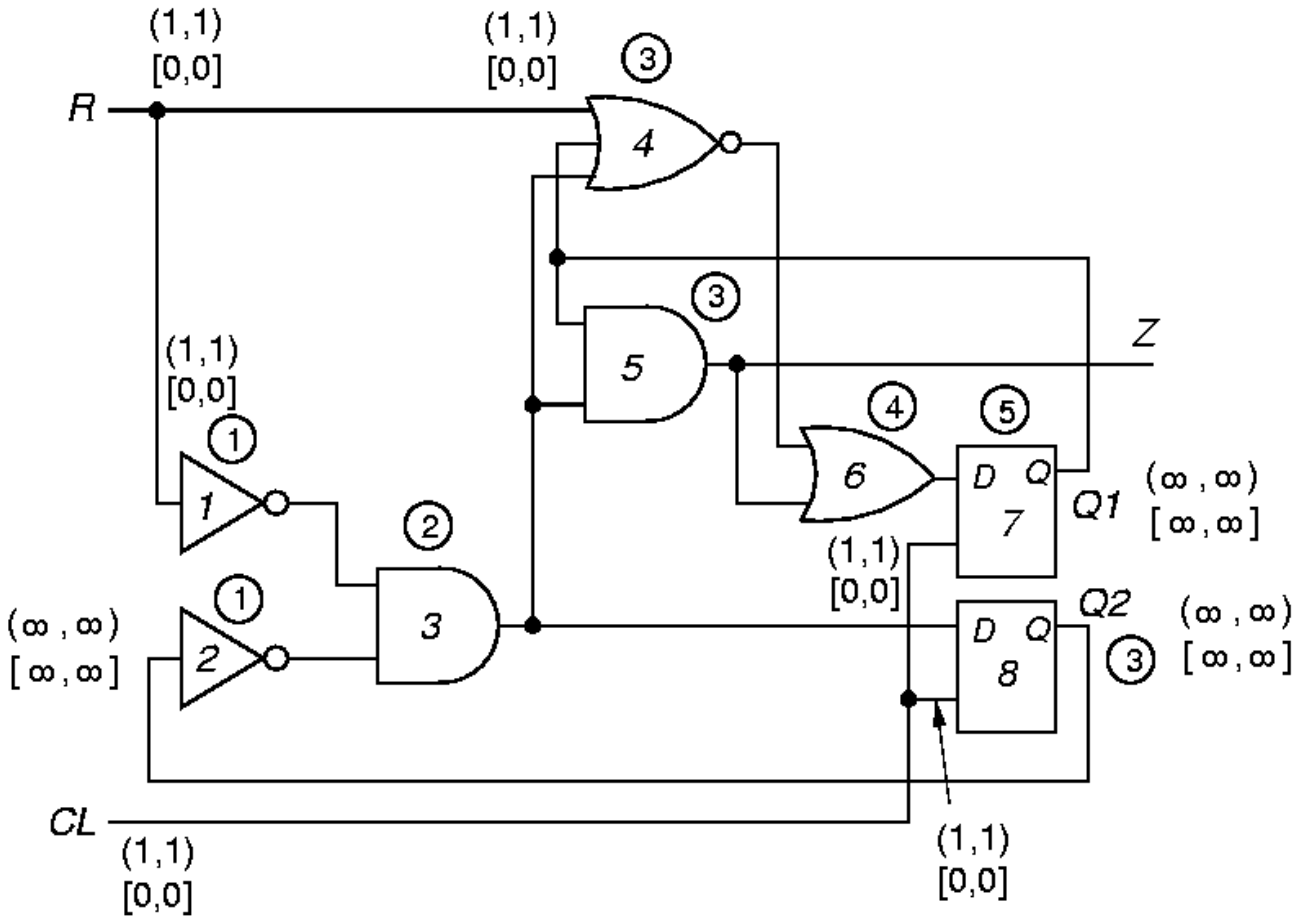
Final Combinational Observabilities



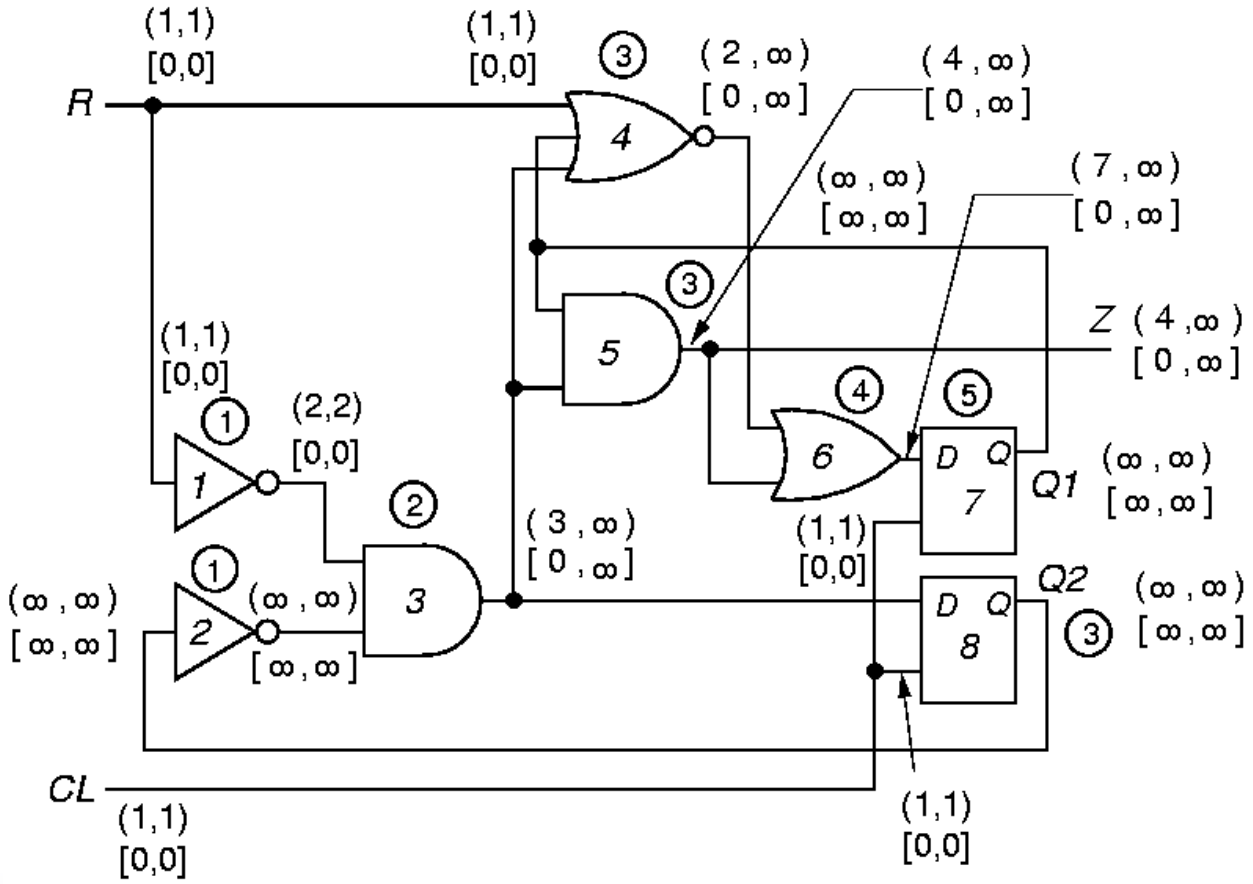
Testability Computation

- For all PIs, $CC0 = CC1 = 1$ and $SC0 = SC1 = 0$
- For all other nodes, $CC0 = CC1 = SC0 = SC1 = \infty$
- Go from PIs to POs, using *CC* and *SC* equations to get controllabilities -- Iterate on loops until *SC* stabilizes -- convergence guaranteed
- For all POs, set $CO = SO = 0$
- Work from POs to PIs, Use *CO*, *SO*, and controllabilities to get observabilities
- Fanout stem (*CO*, SO) = min branch (CO, SO)
- If a *CC* or *SC* (*CO* or *SO*) is ∞ , that node is uncontrollable (unobservable)

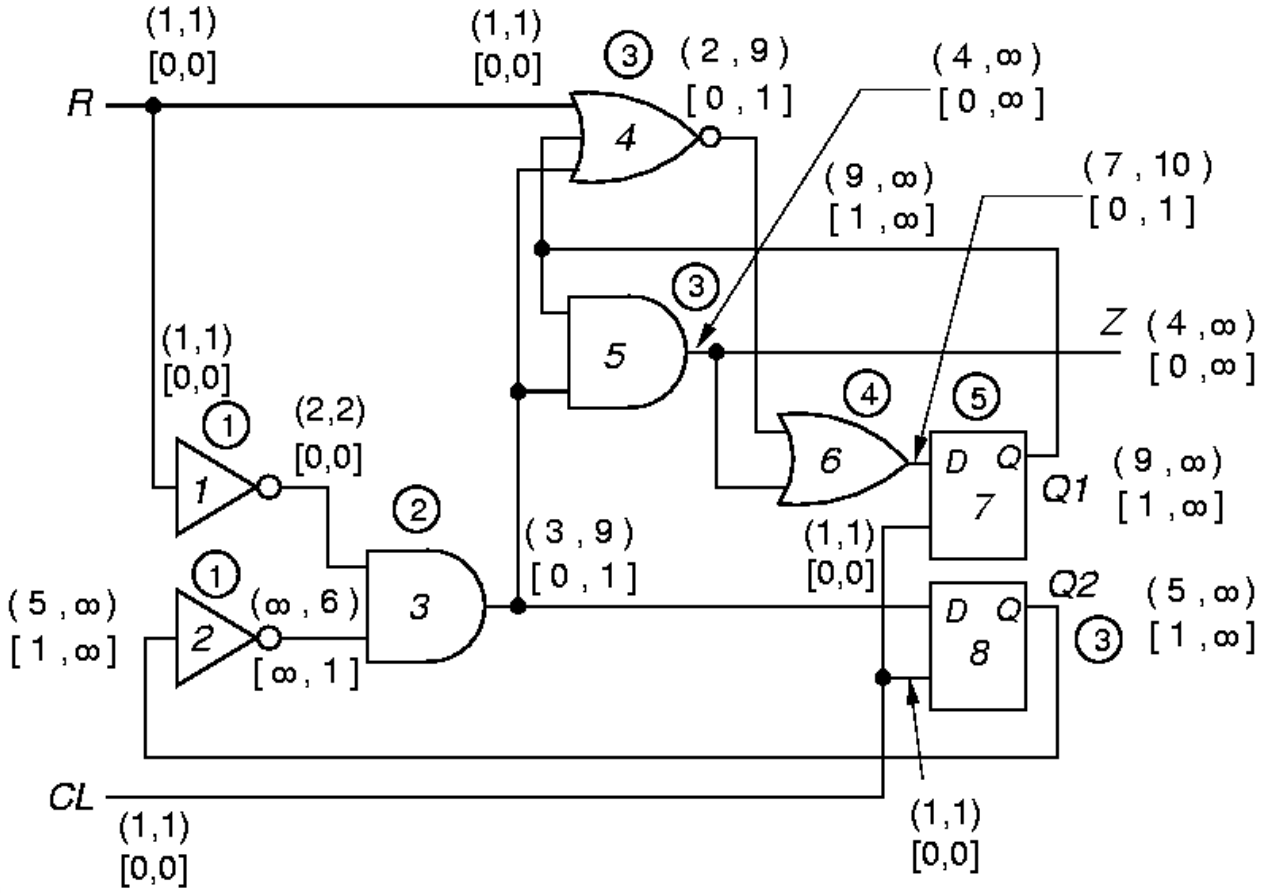
Sequential Example Initialization



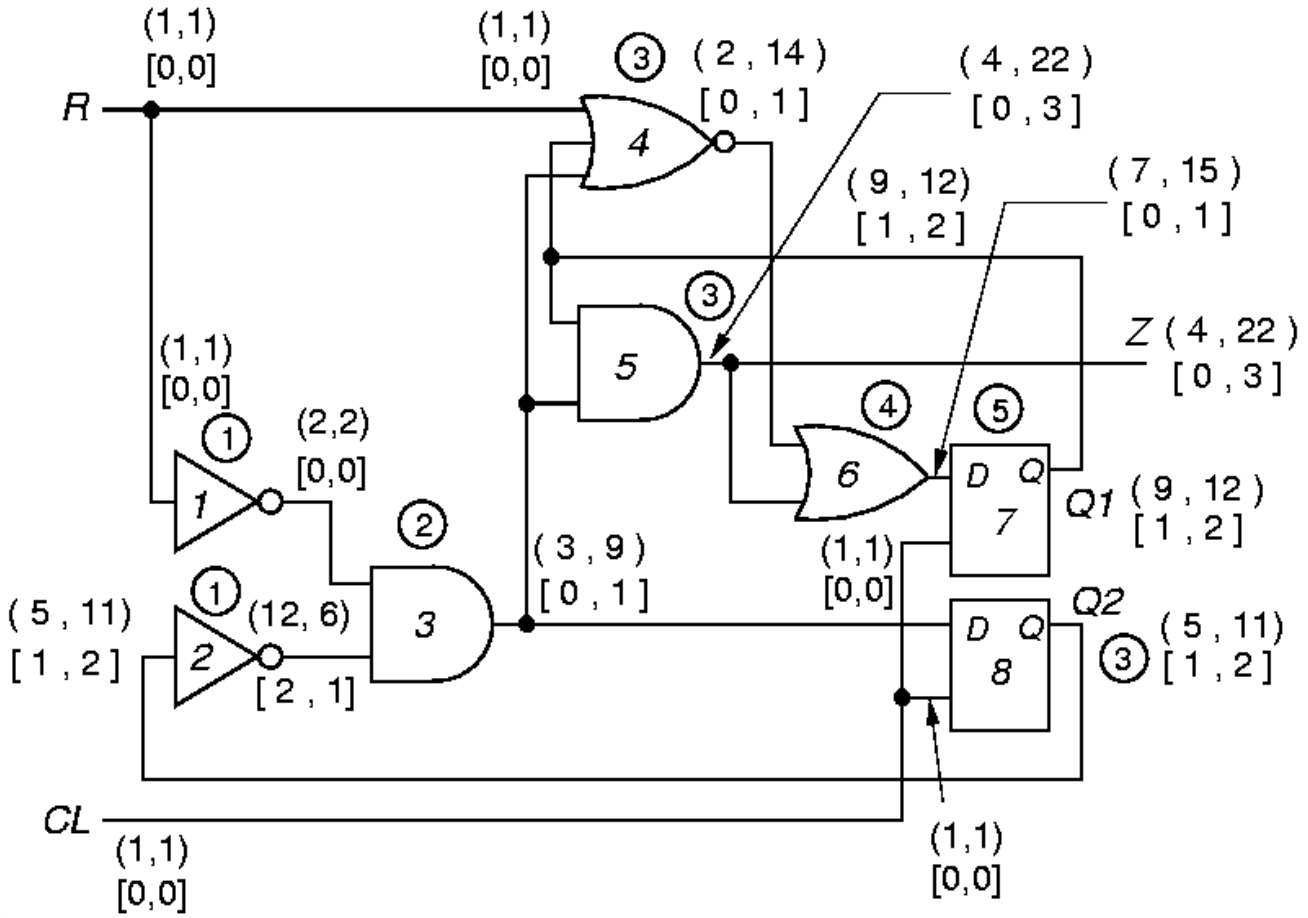
After 1 Iteration



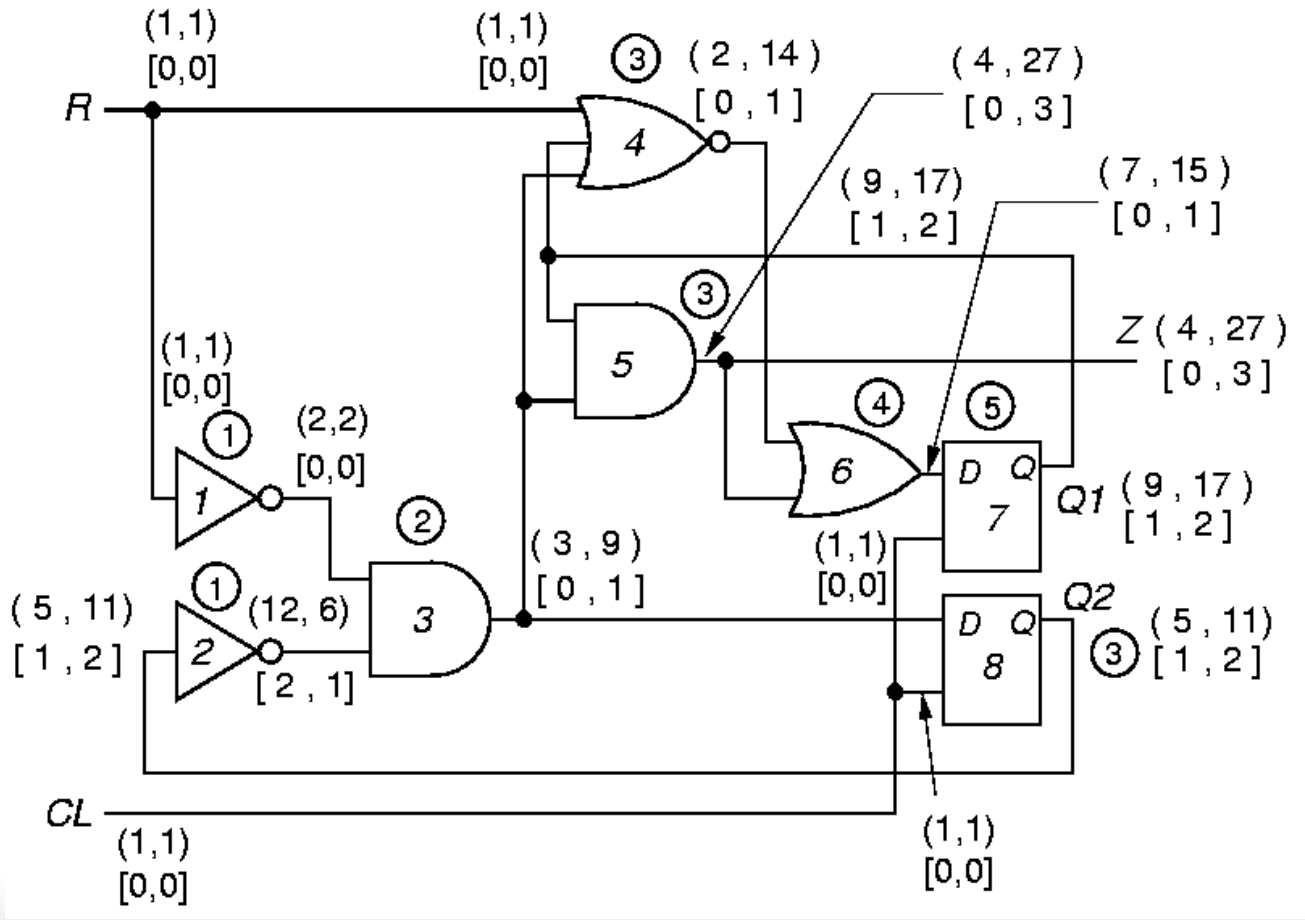
After 2 Iterations



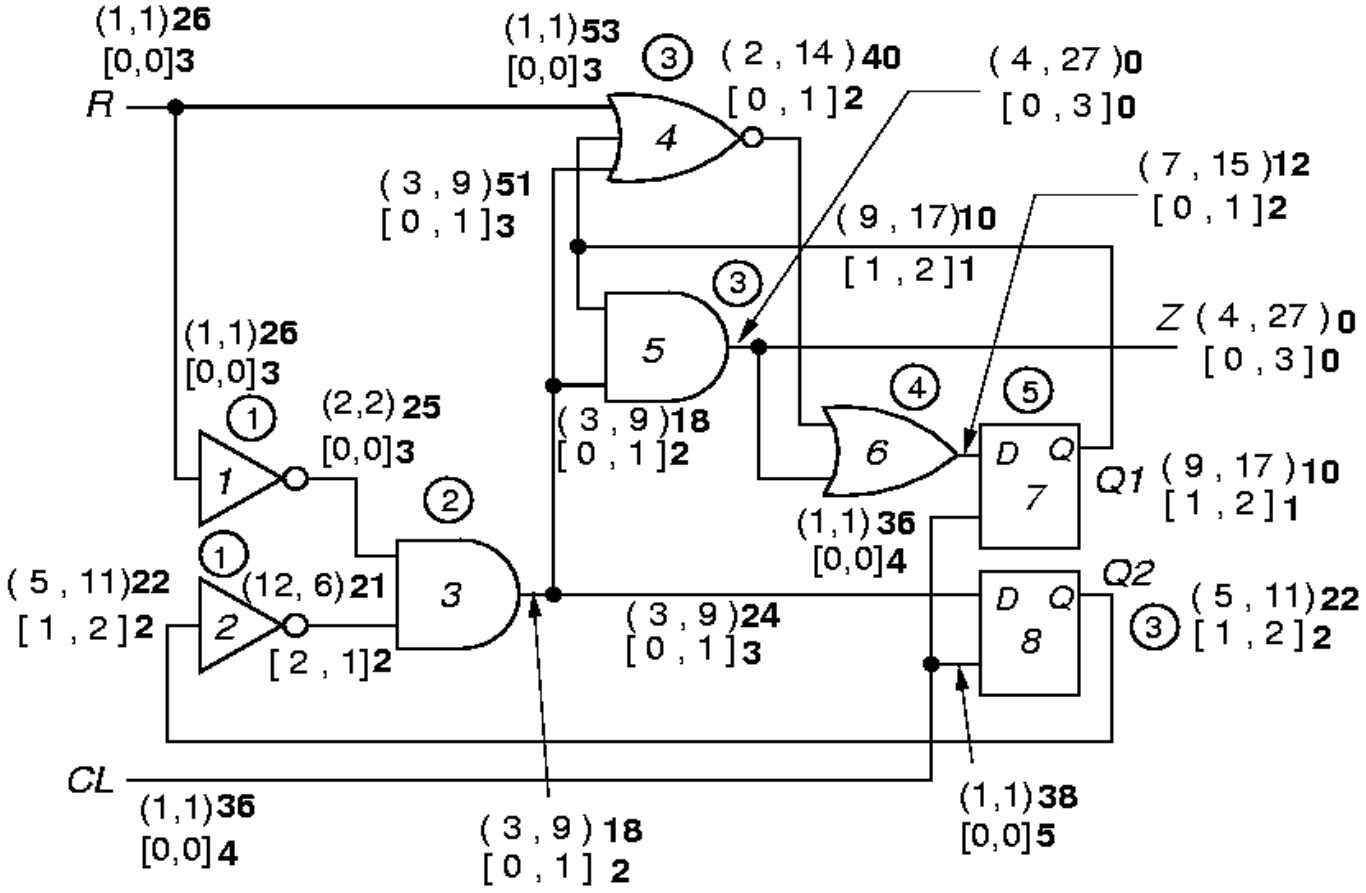
After 3 Iterations



Stable Sequential Measures



Final Sequential Observabilities



Probability-Based *Testability Analysis*

- Used to analyze the **random testability** of the circuit
  - C0(*s*): probability-based 0-controllability of *s*
  - C1(*s*): probability-based 1-controllability of *s*
  - O(*s*): probability-based observability of *s*
- Range between 0 and 1
- C0(*s*) + C1(*s*) = 1

Probability-based controllability calculation rules

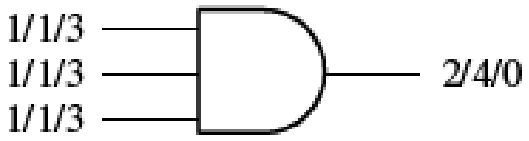
	0-controllability (Primary input, output, branch)	**1-controllability** (Primary input, output, branch)
Primary Input	p_0	$p_1 = 1 - p_0$
AND	1 – (output 1-controllability)	Π (input 1-controllabilities)
OR	Π (input 0-controllabilities)	1 – (output 0-controllability)
NOT	Input 1-controllability	Input 0-controllability
NAND	Π (input 1-controllabilities)	1 – (output 0-controllability)
NOR	1 – (output 1-controllability)	Π (input 0-controllabilities)
BUFFER	Input 0-controllability	Input 1-controllability
XOR	1 – 1-controllabilty	$\Sigma(C1(a) \times C0(b), C0(a) \times C1(b))$
XNOR	1 – 1-controllability	$\Sigma(C0(a) \times C0(b), C1(a) \times C1(b))$
Branch	Stem 0-controllability	Stem 1-controllability

Probability-based observability calculation rules

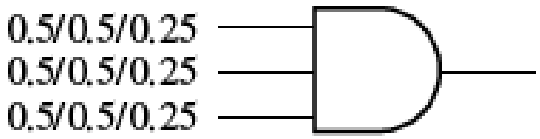
	Observability (Primary output, input, stem)
Primary Output	1
AND / NAND	Π (output observability, 1-controllabilities of other inputs)
OR / NOR	Π (output observability, 0-controllabilities of other inputs)
NOT / BUFFER	Output observability
XOR / XNOR	*a*: Π (output observability, *max* {0-controllability of *b*, 1-controllability of *b*}) *b*: Π (output observability, *max* {0-controllability of *a*, 1-controllability of *a*})
Stem	*max* {branch observabilities}

a, *b*: inputs of an XOR or XNOR gate

Difference between SCOAP testability measures and probability-based testability measures of a 3-input AND gate



(a) SCOAP combinational measures



(b) Probability-based measures

*v*1/*v*2/*v*3 represents the signal's 0-controllability (*v*1), 1-controllability (*v*2), and observability (*v*3)

Simulation-Based *Testability Analysis*

- Supplement to static or topology-based testability analysis
- Performed through statistical sampling
- Guide testability enhancement in test generation or logic BIST
- Generate more accurate estimates
- Require a long simulation time

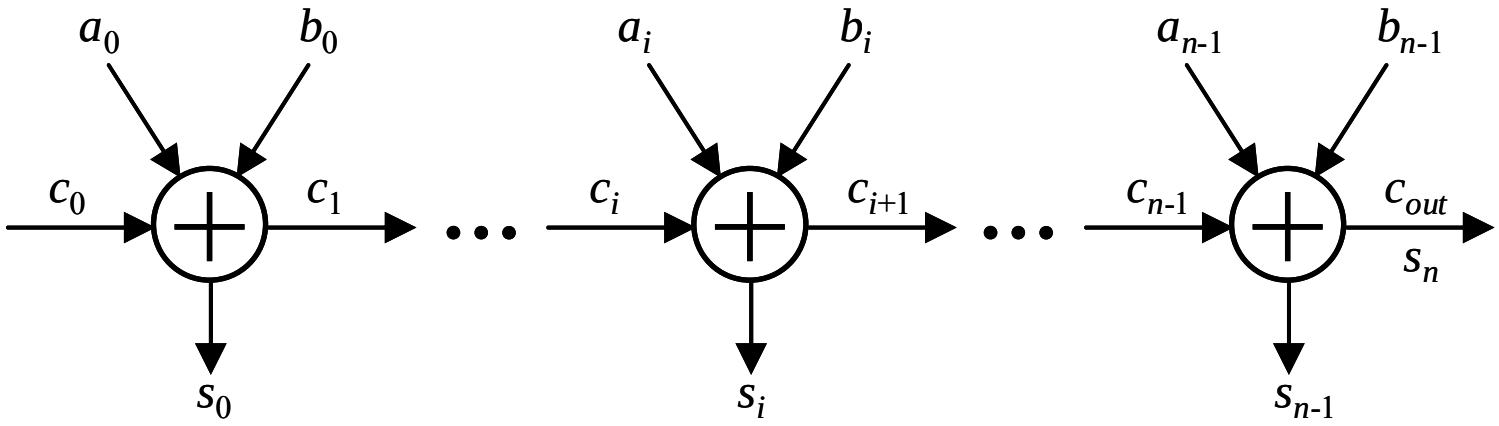
RTL *Testability Analysis*

- Disadvantages of Gate-Level Testability Enhancement
  - Costly in term of area overhead
  - Possible performance degradation
  - Require many DFT iterations
  - Long test development time

RTL *Testability Analysis*

- Advantages of RTL Testability Analysis
  - Improve data path testability
  - Improve the random pattern testability of a scan-based logic BIST circuit
  - Lead to more accurate results
    - The number of reconvergent fanouts is much less
  - Become more time efficient
    - Much simpler than an equivalent gate-level model

RTL *Testability Analysis* - Example



Ripple-carry adder composed of n full-adders

RTL *Testability Analysis* - Example

The probability-based 1-controllability measures of s_i and c_{i+1}, denoted by $C1(s_i)$ and $C1(c_{i+1})$, are calculated as follows:

$$C1(s_i) = \alpha + C1(c_i) - 2 \times (\alpha \times C1(c_i))$$
$$C1(c_{i+1}) = \alpha \times C1(c_i) + C1(a_i) \times C1(b_i)$$

$$\alpha = C1(a_i) + C1(b_i) - 2 \times C1(a_i) \times C1(b_i)$$

α is the probability that $(a_i \oplus b_i) = 1$

$C1(s_i)$ is the probability that $(a_i \oplus b_i \oplus c_i) = 1$

RTL *Testability Analysis* - Example

The probability-based 0-controllability of each output *l*, denoted by C0(*l*), in the *n*-bit ripple-carry adder is 1- C1(*l*).

O(*l*, s_i) is defined as the probability that a signal change on *l* will result in a signal change on s_i.

$$S_i = C_i + A_i + B_i, \qquad C_{i+1} = C_i(A_i + B_i) + A_iB_i$$

Since $O(a_i, s_i) = O(b_i, s_i) = O(c_i, s_i) = O(s_i)$

where $i = 0, 1, \ldots, n-1$

$$O(a_i, s_k) = \left[C_1(c_i \oplus b_i) \times \prod_{j=i+1}^{k} C_1(a_j \oplus b_j) \right] O(s_k),$$

$$i = 0, \ldots, n-1.$$

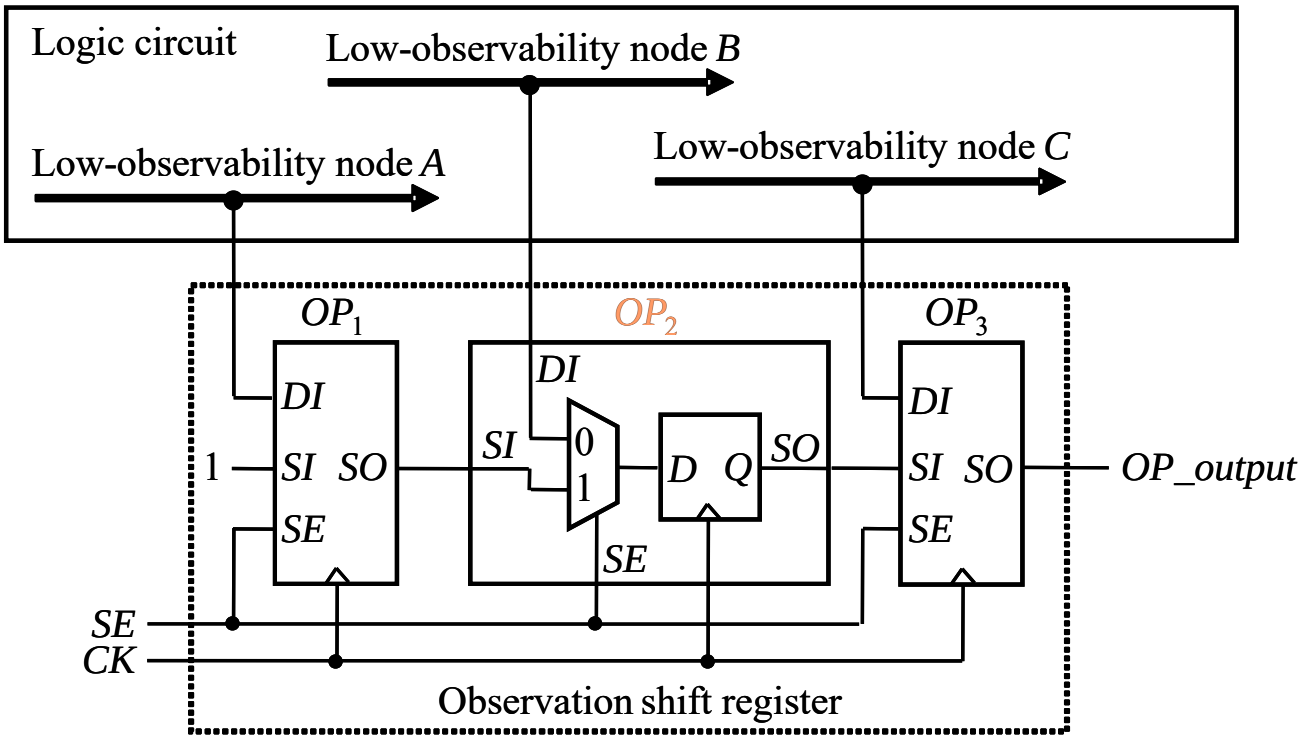
Design for Testability Basics

- *Ad hoc* DFT
  - Effects are local and not systematic
  - Not methodical
  - Difficult to predict
- A structured DFT
  - Easily incorporated and budgeted
  - Yield the desired results
  - Easy to automate

Ad Hoc Approach

- Typical *ad hoc* DFT techniques
  - Insert test points
  - Avoid asynchronous set/reset for storage elements
  - Avoid combinational feedback loops
  - Avoid redundant logic
  - Avoid asynchronous logic
  - Partition a large circuit into small blocks

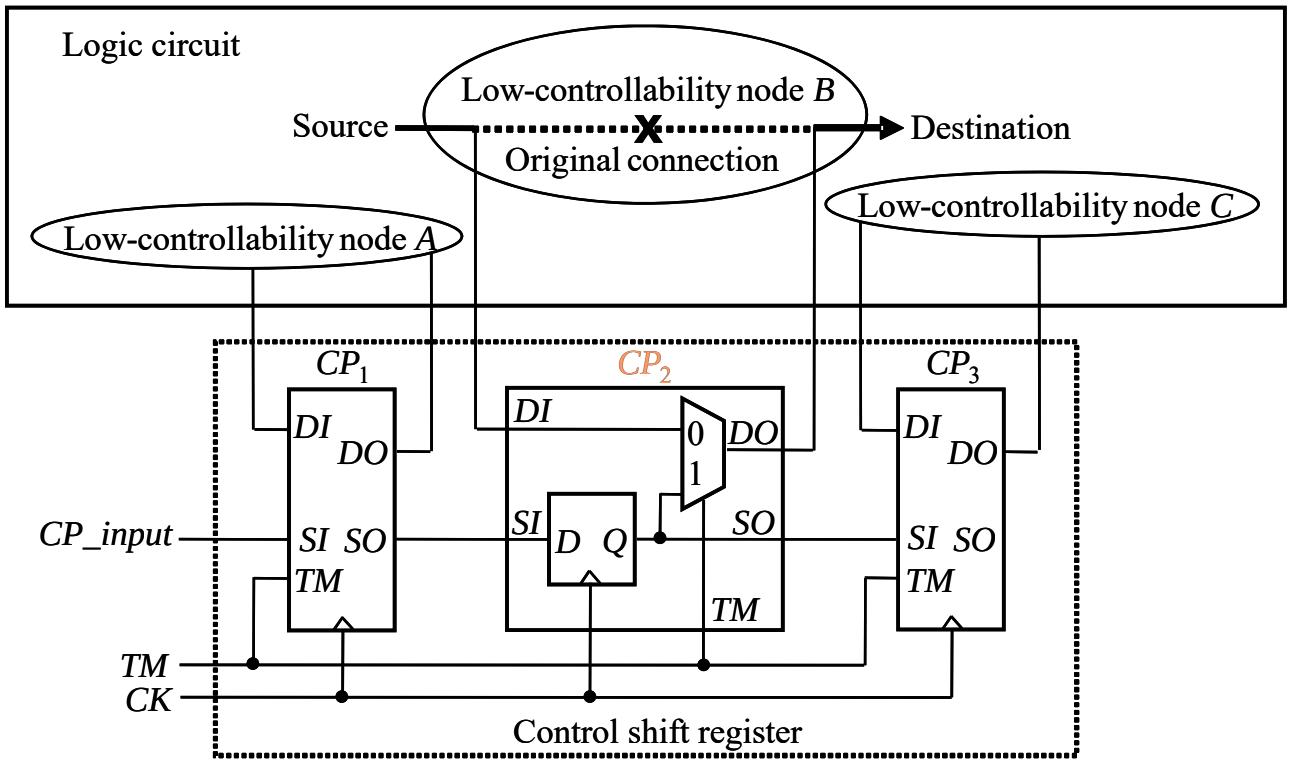
Ad Hoc Approach – Test Point Insertion



Observation point insertion

OP_2 shows the structure of an observation, which is composed of a multiplexer (MUX) and a D flip-flop.

Ad Hoc Approach – Test Point Insertion



Control point insertion

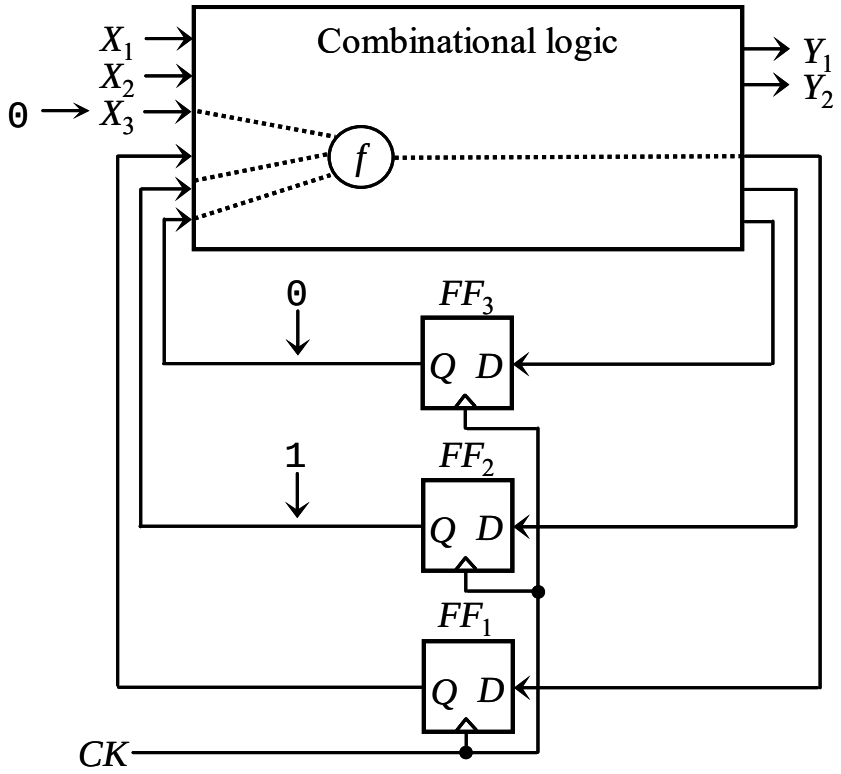
A MUX is inserted between the source and destination ends. During normal operation, $TM = 0$, such that the value from the source end drives the destination end through the 0 port of the MUX.

During test, $TM = 1$ such that the value from the D flip-flop drives the destination end through the 1 port of the MUX.

Structured Approach

- Scan design
  - Convert the sequential design into a scan design
  - Three modes of operation
    - Normal mode
      - All test signals are turned off
      - The scan design operates in the original functional configuration
    - Shift mode
    - Capture mode
      - In both shift and capture modes, a test mode signal *TM* is often used to turn on all test-related fixes

Structured Approach - Scan Design

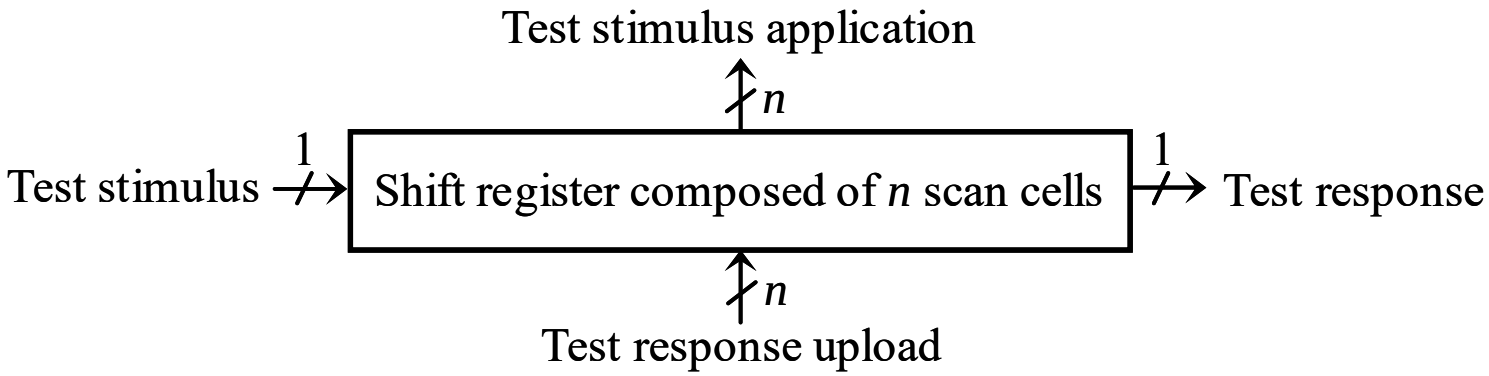


Assume that a stuck-at fault *f* in the combinational logic requires the primary input *X*3, flip-flop FF_2, and flip-flop FF_3, to be set to 0, 1, and 0.

The main difficulty in testing a sequential circuit stems from the fact that it is difficult to control and observe the internal state of the circuit.

Difficulty in testing a sequential circuit

Structured Approach - Scan Design



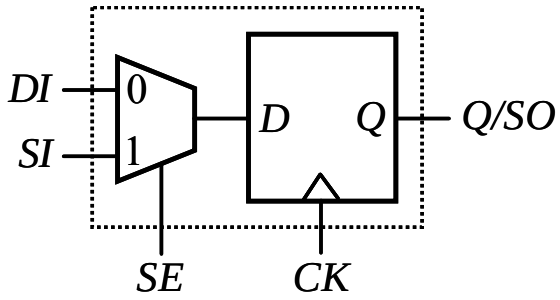
- Converting selected storage elements in the design into **scan cells**.
- Stitching them together to form **scan chains**.

How to detect stuck-at fault *f* :
(1) switching to shift mode and shifting in the desired test stimulus, 1 and 0, to FF_2 and FF_3, respectively
(2) driving a 0 onto primary input X_3
(3) switching to capture mode and applying one clock pulse to capture the fault effect into FF_1
(4) switching back to shift mode and shifting out the test response stored in FF_1, FF_2, and FF_3 for comparison with the expected response.

Scan Cell Design

- A scan cell has two inputs: data input and scan input
  - In normal/capture mode, data input is selected to update the output
  - In shift mode, scan input is selected to update the output
- Three widely used scan cell designs
  - Muxed-D Scan Cell
  - Clocked-Scan Cell
  - LSSD Scan Cell

Muxed-D Scan Cell

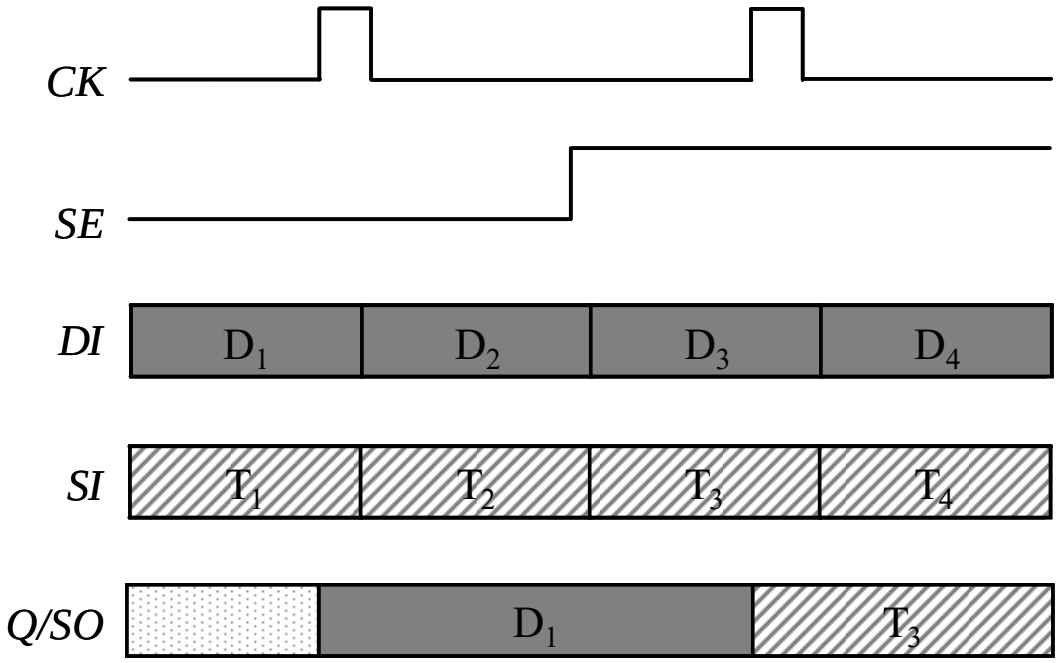


Edge-triggered muxed-D scan cell

This scan cell is composed of a D flip-flop and a multiplexer.

The multiplexer uses an additional scan enable input *SE* to select between the data input *DI* and the scan input *SI*.

Muxed-D Scan Cell

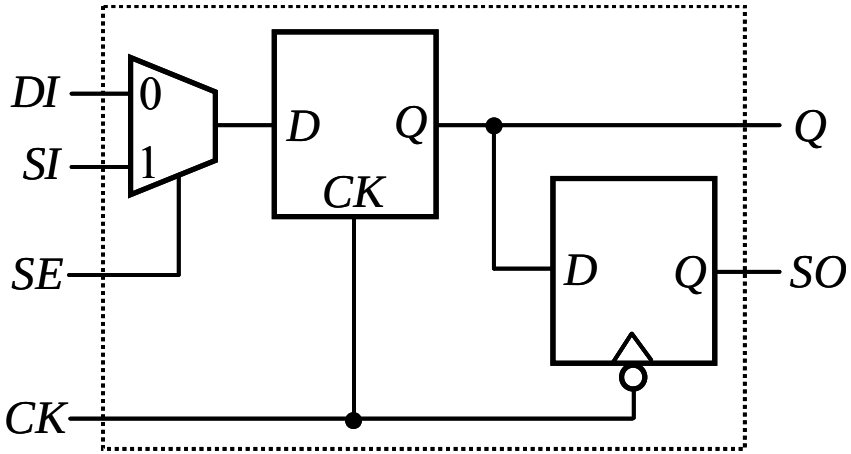


Edge-triggered muxed-D scan cell design and operation

In normal/capture mode, *SE* is set to 0. The value present at the data input *DI* is captured into the internal D flip-flop when a rising clock edge is applied.

In shift mode, *SE* is set to 1. The scan input *SI* is used to shift in new data to the D flip-flop, while the content of the D flip-flop is being shifted out.

Muxed-D Scan Cell

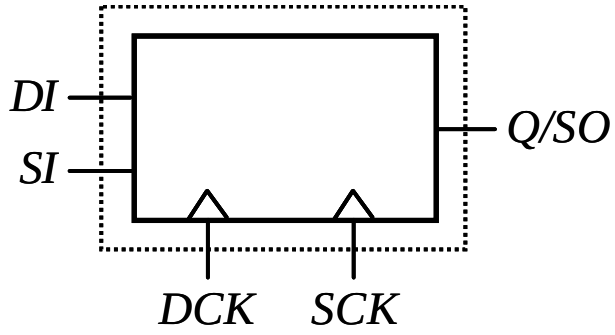


Level-sensitive/edge-triggered muxed-D scan cell design

This scan cell is composed of a multiplexer, a D latch, and a D flip-flop.

In this case, shift operation is conducted in an edge-triggered manner, while normal operation and capture operation is conducted in a level-sensitive manner.

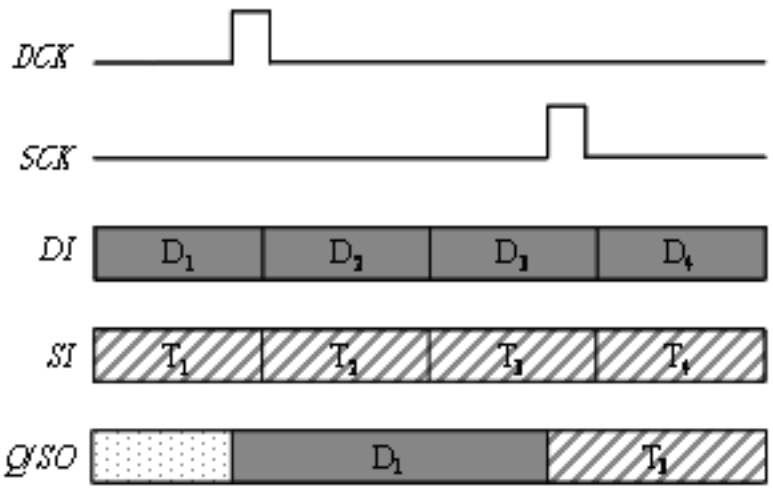
Clocked-Scan Cell



Clocked-scan cell

In the clocked-scan cell, input selection is conducted using two independent clocks, *DCK* and *SCK*.

Clocked-Scan Cell

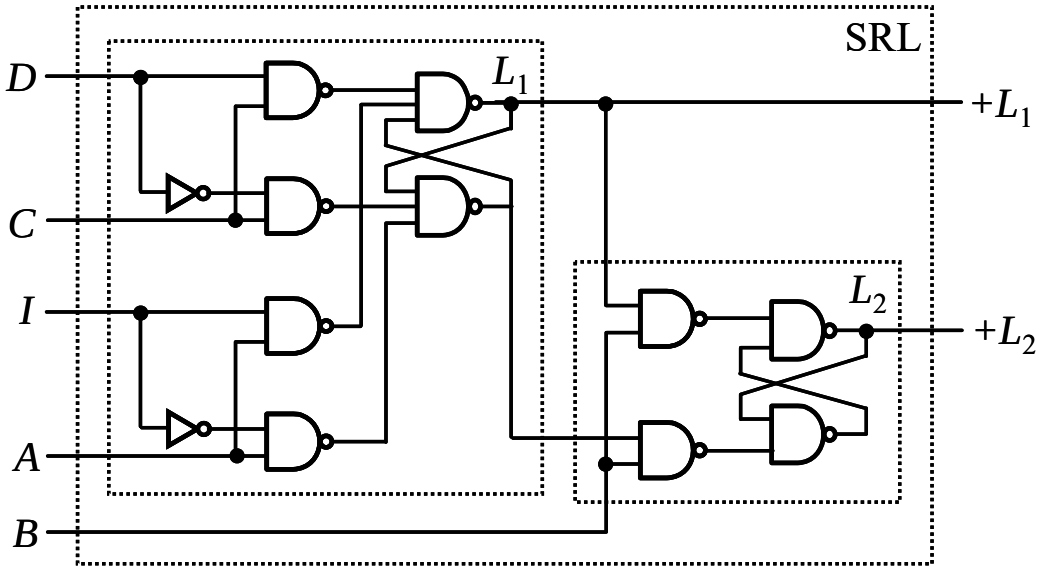


Clocked-scan cell design and operation

In normal/capture mode, the data clock *DCK* is used to capture the contents present at the data input *DI* into the clocked-scan cell.

In shift mode, the shift clock *SCK* is used to shift in new data from the scan input *SI* into the clocked - scan cell, while the content of the clocked-scan cell is being shifted out.

LSSD Scan Cell

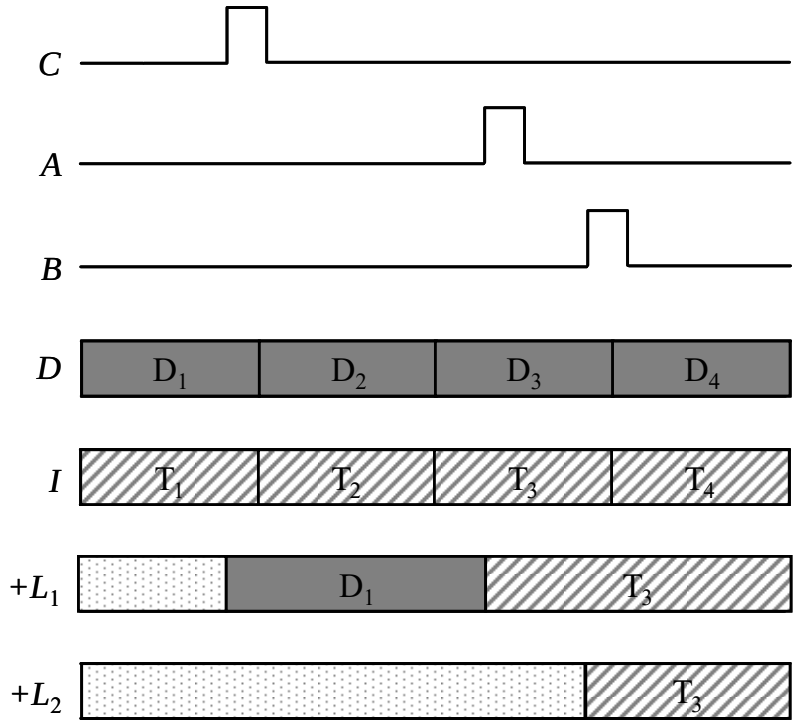


Polarity-hold SRL
(shift register latch)

An LSSD scan cell is used for level-sensitive latch based designs.

This scan cell contains two latches, a master 2-port D latch L_1 and a slave D latch L_2. Clocks C, A and B are used to select between the data input D and the scan input I to drive $+L_1$ and $+L_2$. In an LSSD design, either $+L_1$ or $+L_2$ can be used to drive the combinational logic of the design.

LSSD Scan Cell



Polarity-hold SRL design and operation

In order to guarantee race-free operation, clocks *A*, *B*, and *C* are applied in a non-overlapping manner.

The master latch L_1 uses the system clock *C* to latch system data from the data input *D* and to output this data onto $+L_1$. Clock *B* is used after clock *A* to latch the system data from latch L_1 and to output this data onto $+L_2$.

Capture mode uses both clocks C and B to output system data onto +L2. Shift mode uses clocks A and B to latch scan data from the scan input I to +L1 and then from latch L1 to +L2.

Comparing three scan cell designs

	Advantages	Disadvantages
Muxed-D Scan Cell	Compatibility to modern designs Comprehensive support provided by existing design automation tools	Add a multiplexer delay
Clocked-Scan Cell	No performance degradation	Require additional shift clock routing
LSSD Scan Cell	Insert scan into a latch-based design Guarantee to be race-free	Increase routing complexity

Exercises

- 2.1 (Testability Analysis) Calculate the SCOAP controllability and observability measures for a three-input XOR gate and for its NAND–NOR implementation.

- 2.2 (Testability Analysis) Use the rules given in Tables 2.3 and 2.4 to calculate the probability-based testability measures for a three-input XNOR gate and for its NAND–NOR implementation. Assume that the probability-based controllability values at all primary inputs and the probability-based observability value at the primary output are 0.5 and 1, respectively.

TABLE 2.3 ■ Probability-Based Controllability Calculation Rules

	0-Controllability (Primary Input, Output, Branch)	1-Controllability (Primary Input, Output, Branch)
Primary Input	p_0	$p_1 = 1 - p_0$
AND	1 – (output 1-controllability)	∏ (input 1-controllabilities)
OR	∏ (input 0-controllabilities)	1 – (output 0-controllability)
NOT	Input 1-controllability	Input 0-controllability
NAND	∏ (input 1-controllabilities)	1 – (output 0-controllability)
NOR	1 – (output 1-controllability)	∏ (input 0-controllabilities)
BUFFER	Input 0-controllability	Input 1-controllability
XOR	1 – 1-controllability	$\sum (C1(a) \times C0(b), C0(a) \times C1(b))$
XNOR	1 – 1-controllability	$\sum (C0(a) \times C0(b), C1(a) \times C1(b))$
Branch	Stem 0-controllability	Stem 1-controllability

Note: *a* and *b* are inputs of an XOR or XNOR gate.

TABLE 2.4 ■ Probability-Based Observability Calculation Rules

	Observability (Primary Output, Input, Stem)
Primary Output	1
AND/NAND	∏ (output observability, 1-controllabilities of other inputs)
OR/NOR	∏ (output observability, 0-controllabilities of other inputs)
NOT/BUFFER	Output observability
XOR/XNOR	*a*: ∏ (output observability, *max* {0-controllability of *b*, 1-controllability of *b*})
	b: ∏ (output observability, *max* {0-controllability of *a*, 1-controllability of *a*})
Stem	*max* {branch observabilities}

Note: *a* and *b* are inputs of an XOR or XNOR gate.

下次课预告

时间：2021年09月22日（周三6:10pm）

地点：教1-109

内容：可测试性设计(II)

教材：VLSI TEST PRINCIPLES AND ARCHITECTURES

Chapter 2 Design for Testability