VLSI Test Technology and Reliability (ET4076)

Lecture 4 (part 2)

Testability Measurements
(Chapter 6)

Said Hamdioui

Computer Engineering Lab
Delft University of Technology
2009-2010

Learning aims of today's lecture

- Be able to describe the need and the importance of testability measurements for test generation
- Be able to perform
  - Combinational testability measures
  - Sequential testability measures

Contents

- Testability
  - Definition, purpose, …
- SCOAP measures
  - Definitions of Controllability and observability
  - Calculations of Controllability and observability
  - Sources of correlation error
- Combinational SCOAP measures
- Sequential SCOAP measures
- SCOAP for test vector length prediction
- (High-Level testability measures)
- Summary

Testability ….

- Different meanings and interpretations
  - For test/design engineer
    - Complexity of test generation as a measure of testability
  - Test engineer
    - Compatibility of design with test equipment
  - Product/quality engineer
    - Fault coverage

- **Testability** is the property of a circuit that makes it ***easy to test***

Testability …. Purpose

- Need approximate ***measure*** of:
  - Difficulty of **setting** internal circuit lines to 0 or 1 by setting **primary circuit** inputs
  - Difficulty of **observing** internal circuit lines by observing **primary outputs**

- Uses:
  - Analysis of difficulty of testing internal circuit parts – redesign or add special test hardware
  - Guidance for algorithms computing test patterns – avoid using hard-to-control lines
  - Estimation of fault coverage
  - Estimation of test vector length

Testability…….. Analysis

Two main attributes:

- Involves **Circuit Topological analysis**, but no test vectors and no search algorithm
  - Static analysis
- **Linear computational complexity**
  - Otherwise, is pointless – might as well use automatic test-pattern generation (or fault simulation) and calculate:
    - Exact fault coverage
    - Exact test vectors

SCOAP Measures……. Types

- **SCOAP** – Sandia Controllability and Observability Analysis Program [Rutman 1972]

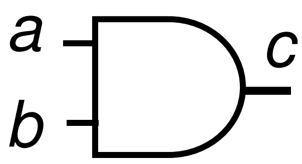
- Six numerical measures
  - Combinational measures:
    - *CC0* – Difficulty of setting circuit line to logic 0
    - *CC1* – Difficulty of setting circuit line to logic 1
    - *CO* – Difficulty of observing a circuit line
  - Sequential measures – analogous:
    - *SC0*
    - *SC1*
    - *SO*

SCOAP Measures…. Range

- Controllabilities – 1 (easiest) to infinity (hardest)
- Observabilities – 0 (easiest) to infinity (hardest)
- Combinational measures:
  - Roughly proportional to # of circuit lines that must be set to control or observe given line
- Sequential measures:
  - Roughly proportional to # of times a flip-flop must be clocked to control or observe given line

SCOAP Measures……Goldstein

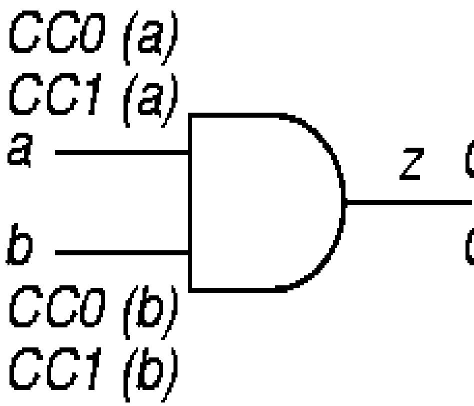
- AND gate Output 0 controllability
  - output_controllability = min (input_controllabilities) + 1



- AND gate Output 1 controllability
  - output_controllability = Σ (input_controllabilities) + 1

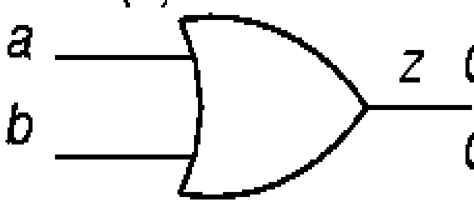
- XOR gate Output controllability
  - output_controllability = min (controllabilities of each input set) + 1

SCOAP Measures… Controllability calculations(1)



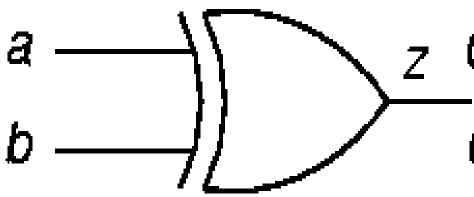
$CC0(z) = \min(CC0(a), CC0(b)) + 1$

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



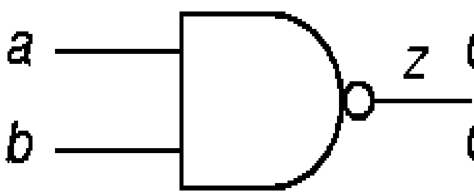
$CC0(z) = CC0(a) + CC0(b) + 1$

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



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

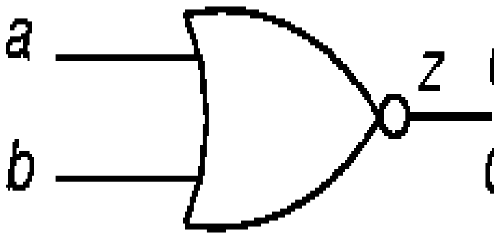
$CC1(z) = \min(CC1(a) + CC0(b), CC0(a) + CC1(b)) + 1$



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

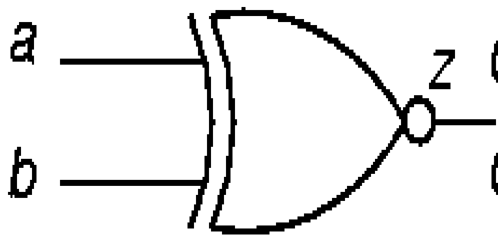
$CC1(z) = \min(CC0(a), CC0(b)) + 1$

SCOAP Measures…Controllability calculations(2)



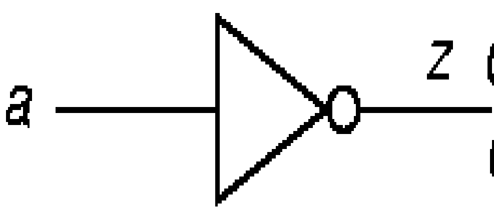
$$CC0(z) = \min(CC1(a), CC1(b)) + 1$$

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



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

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



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

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

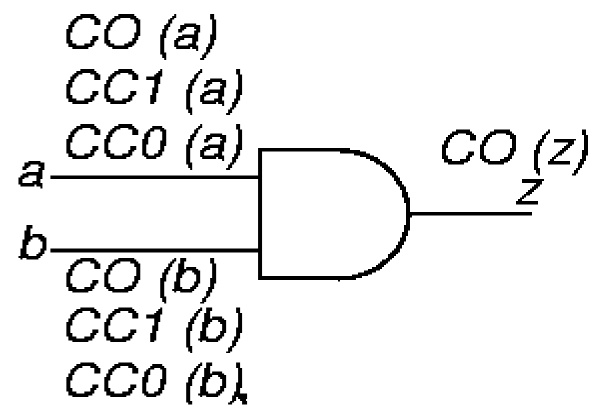
SCOAP Measures …Observability calculations(1)

- To observe a gate input:
  - Observability is computed in reverse order starting at PO and moving backwards to PI
  - Set the obervability difficulty CO of PO to **0** (for both 0 & 1)
  - The CO of an input of a gate is equal to the obervability of its output, plus the difficulty of setting all other inputs to non-controlling values, plus 1 (to account for logic depth)

- **Example**

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

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



SCOAP Measures…Observability calculations(2)

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

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

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

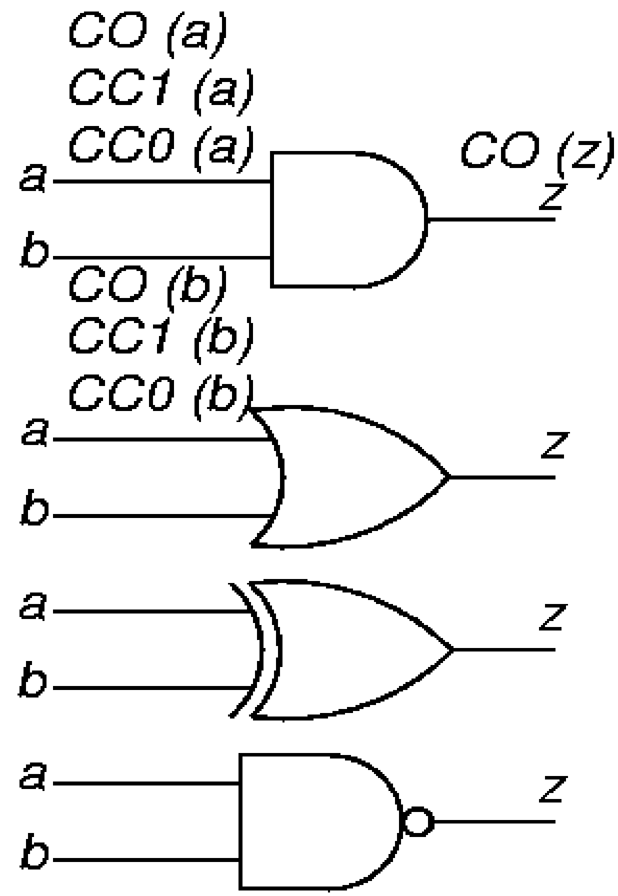
$CO(b) = CO(z) + CC0(a) + 1$

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

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

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

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

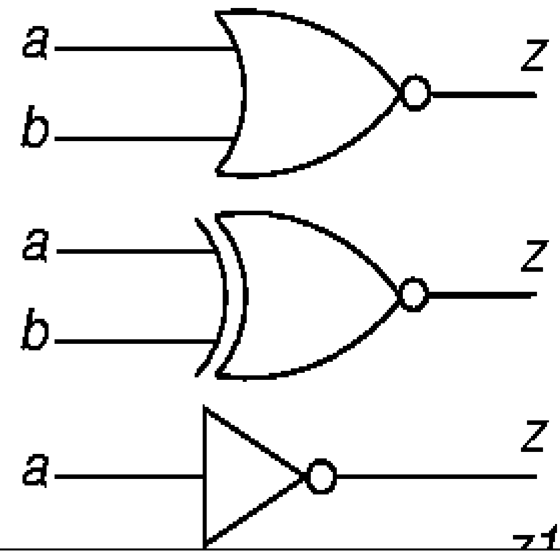


SCOAP Measures…Observability calculations(3)

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

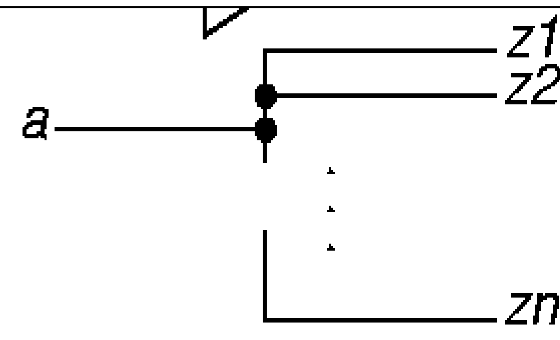
$$CO(a) = CO(z) + \min(CC0(b), CC1(b)) + 1$$
$$CO(b) = CO(z) + \min(CC0(a), CC1(a)) + 1$$

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



- To observe a fanout stem:
  - Observe it through branch with best observability

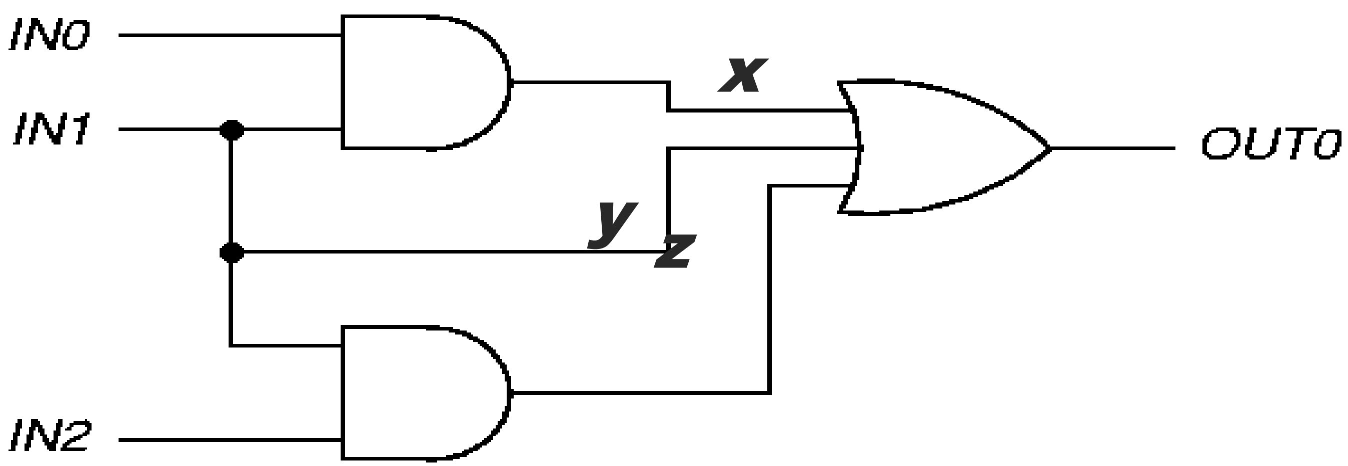
$$CO(a) = \min(CO(z1), CO(z2), ..., CO(zn))$$



SCOAP Measures: Source of correlation errors

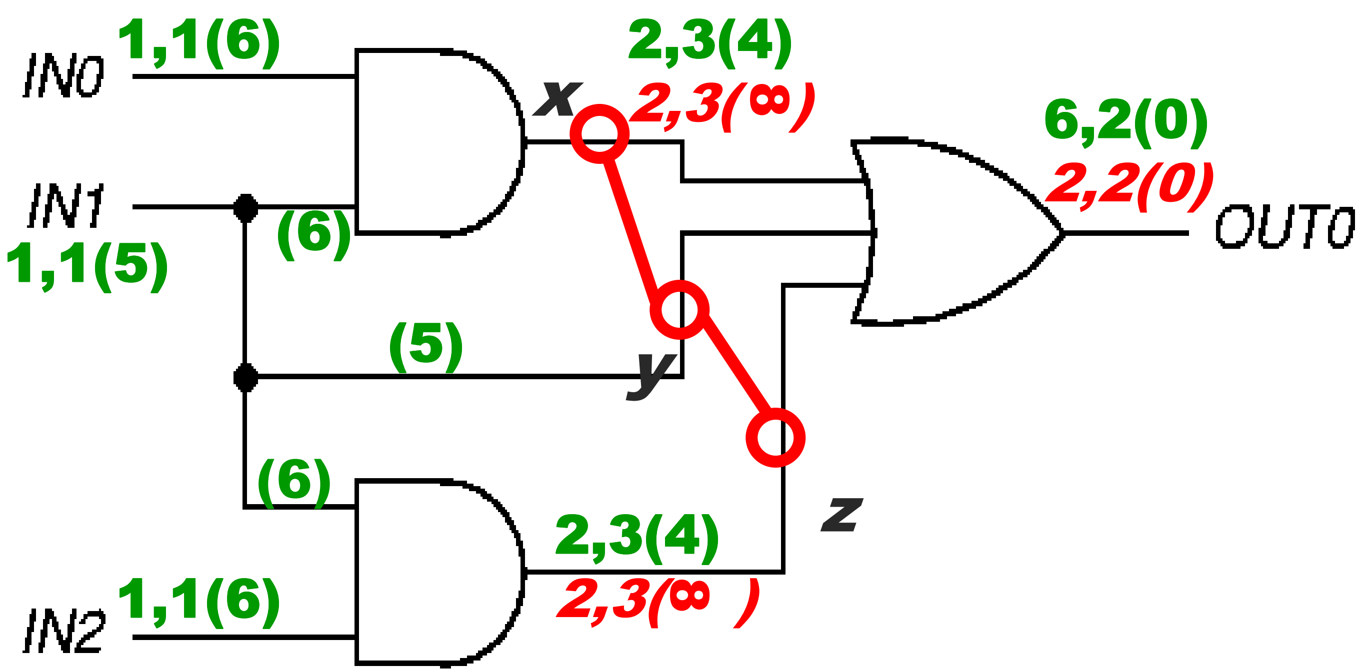
- SCOAP measures wrongly assume that controlling or observing reconverging fanout stems are **independent**

- Example*:*
  - In SCOAP: x, y, z are assumed independent events
  - *CC0 (x), CC0 (y), CC0 (z)* correlate
  - *CC1 (x), CC1 (y), CC1 (z)* correlate
  - *CO (x), CO (y), CO (z)* correlate



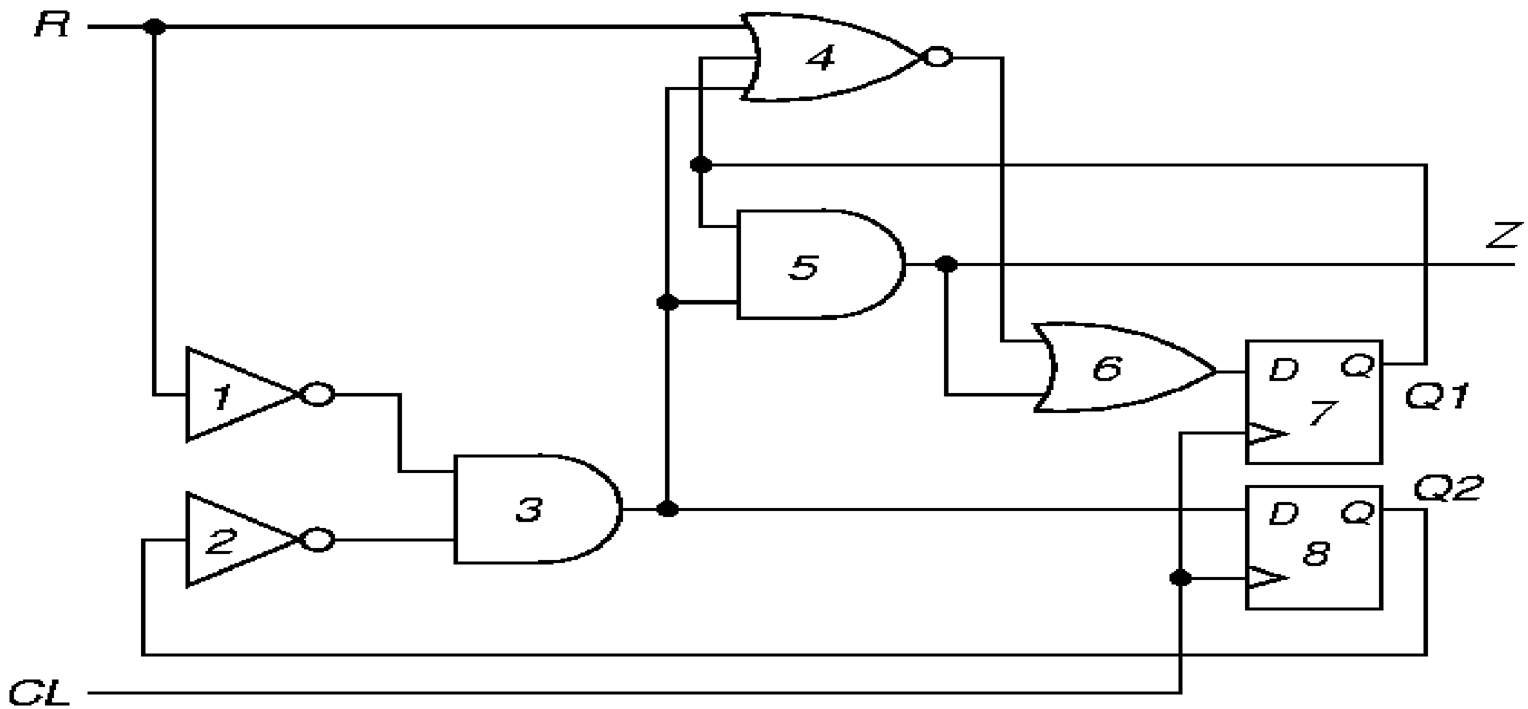
SCOAP Measures: Example of correlation errors

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



Combinational SCOAP…. Example(1)

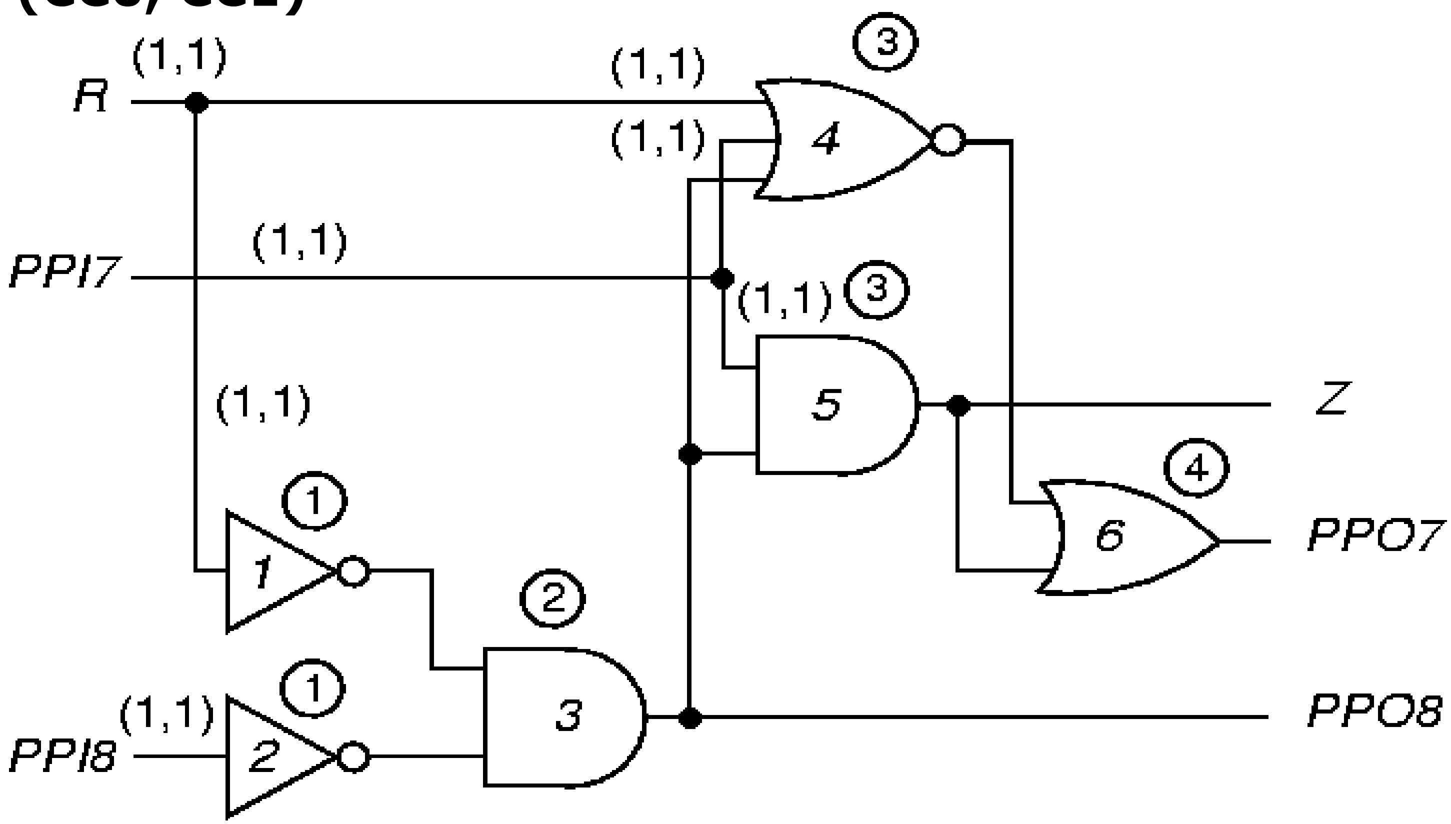
- Assume FFs have special test hardware to:
  - Read the current state and
  - Set the presence state
- D lines are the pseudo-primary outputs PPOs
- Q lines are pseudo-primary inputs PPIs



Combinational SCOAP…. Example(2)

- Replace all FFs with PPI-PPO pairs
  - CL is removed
- Gates are labeled in **level order** (1, 2, 3, 4)
  - (see algorithm 6.1 / next slide)

(CC0, CC1)



Combinational SCOAP…. Example(3)

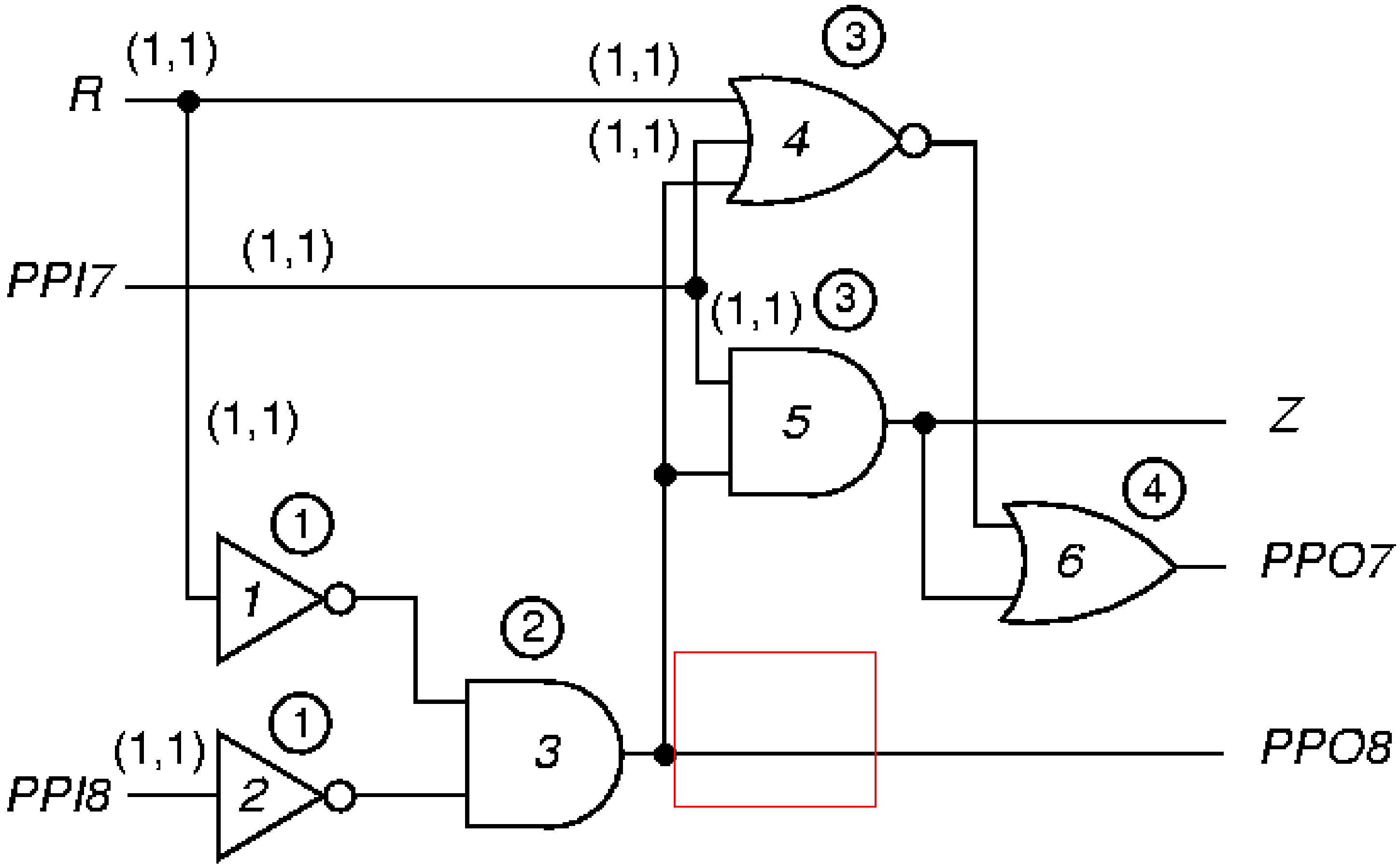
Levelization Algorithm 6.1

- 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

Combinational SCOAP…. Example(4)

Controllability Through Level 0

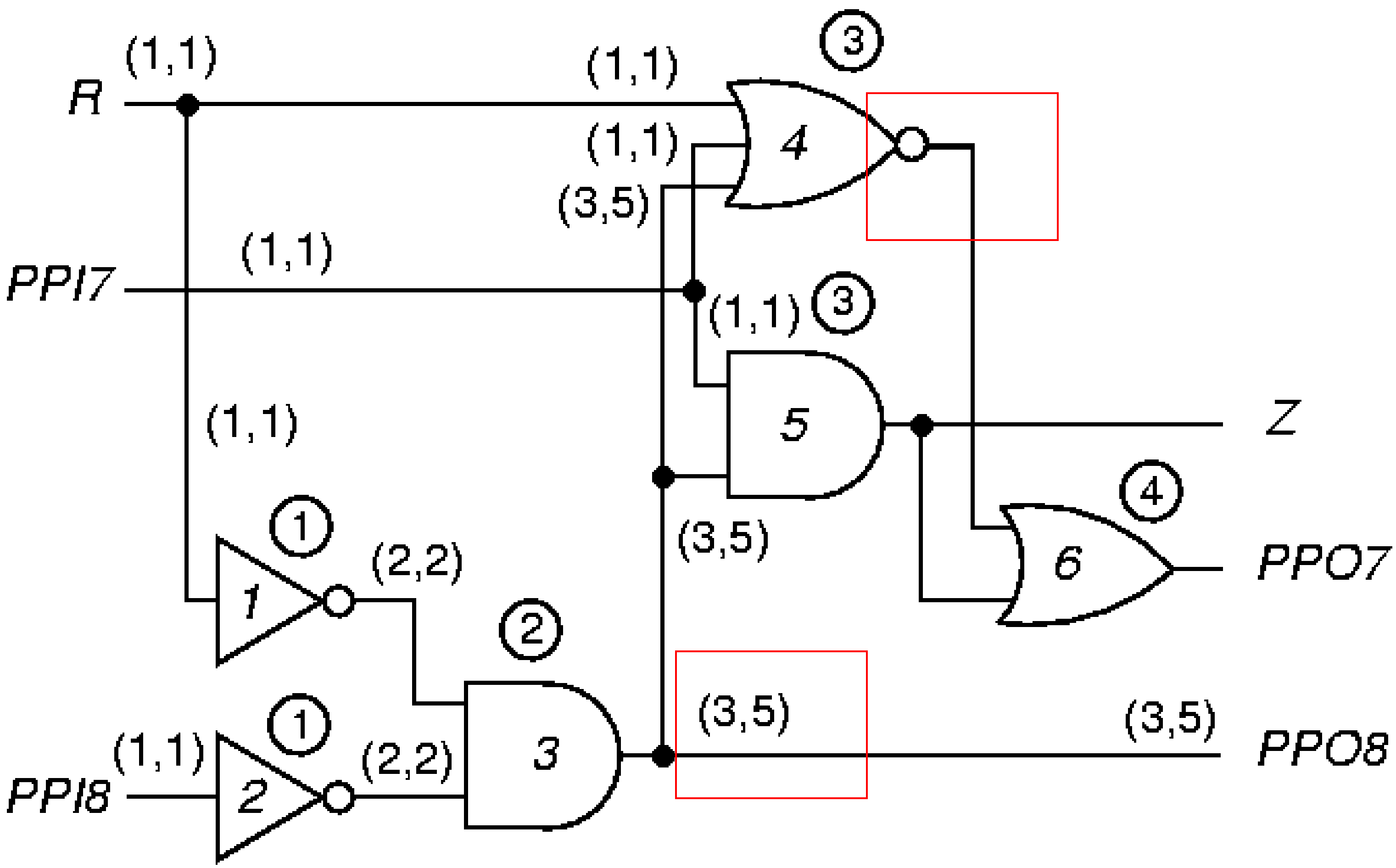
- Circled numbers give level number. (CC0, CC1)



Combinational SCOAP…. Example(5)

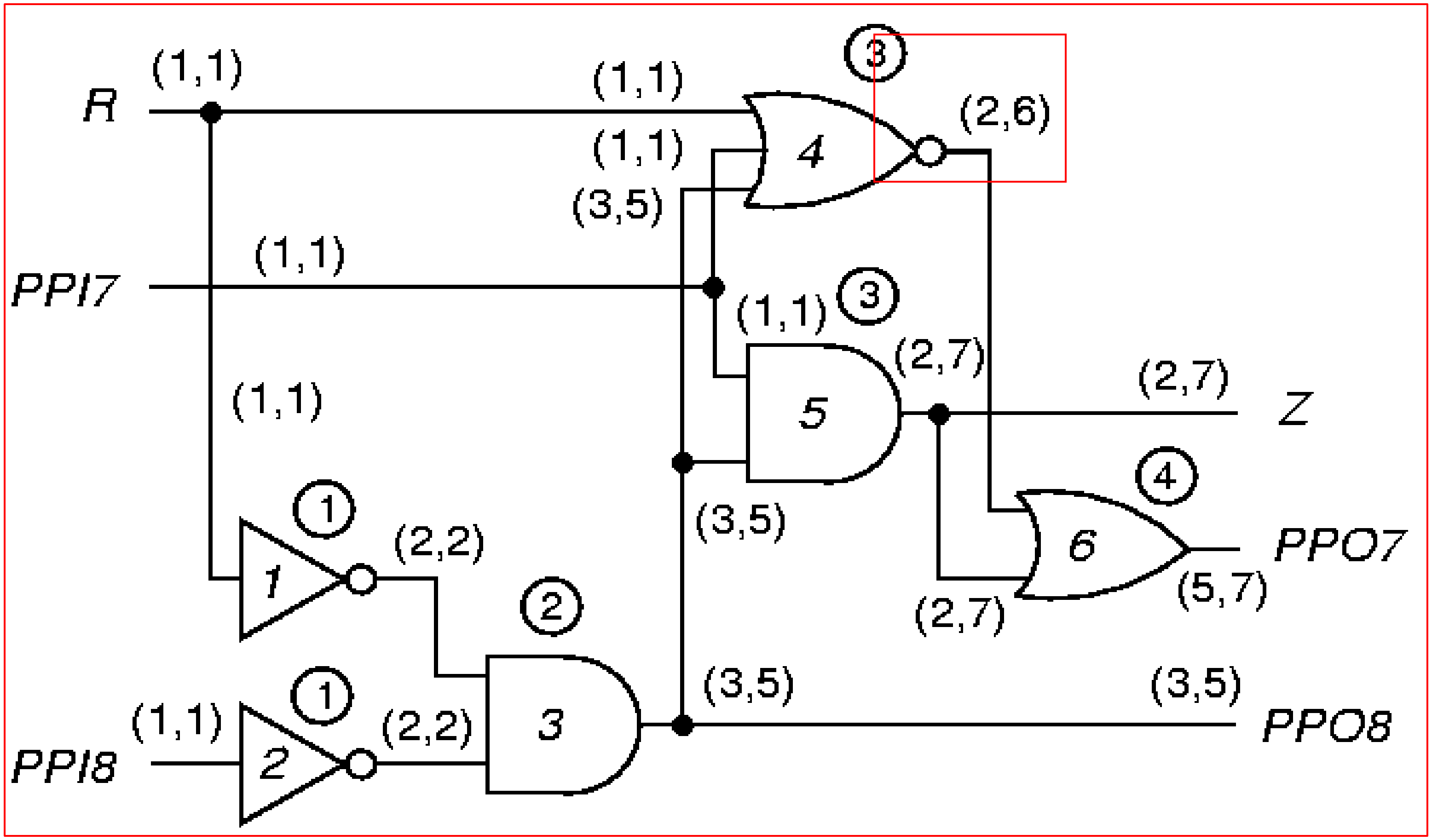
Controllability Through Levels 1 & 2

- Circled numbers give level number. (CC0, CC1)



Combinational SCOAP…. Example(6)

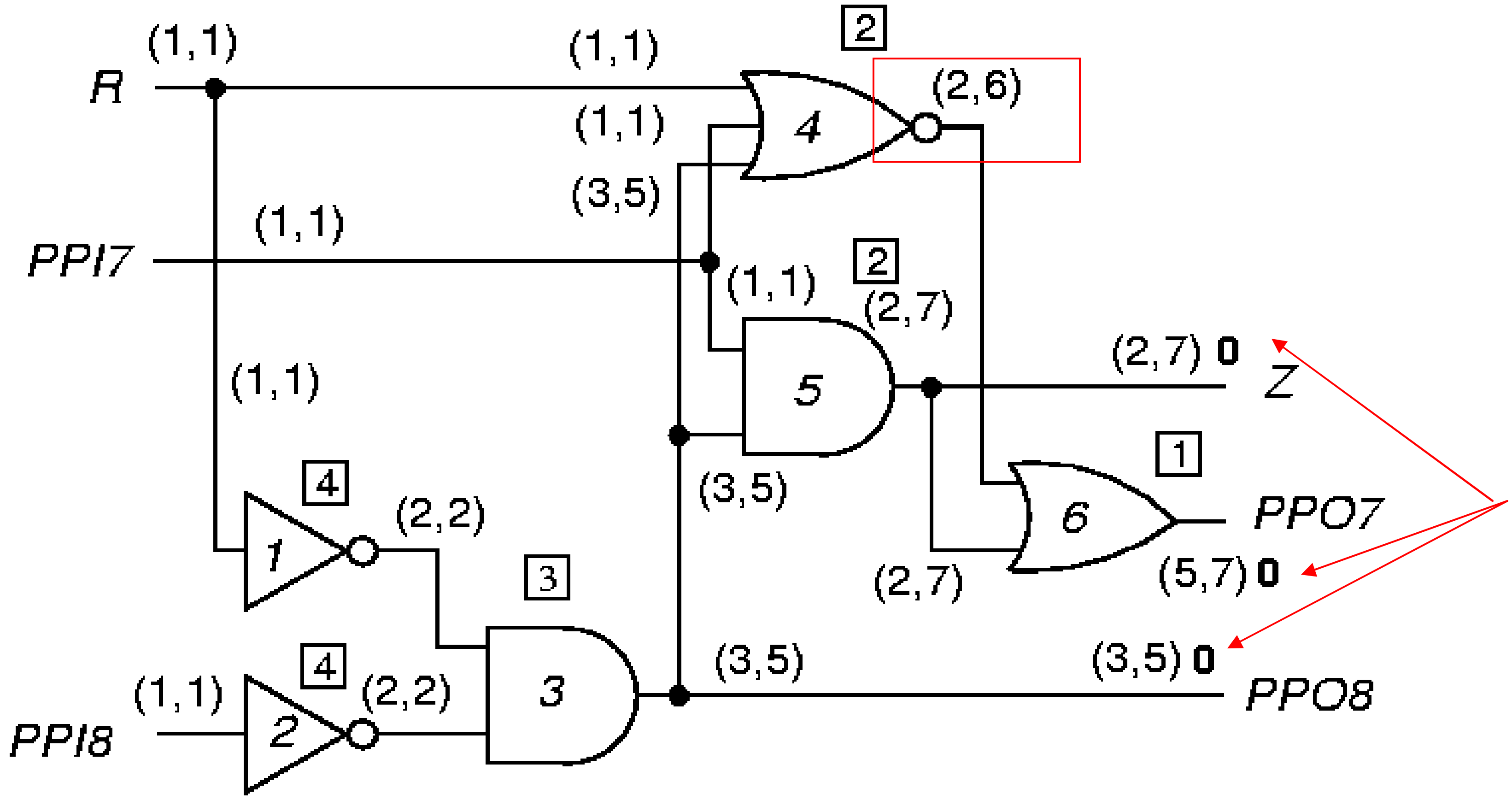
- **Final controllability**
  - Circled numbers give level number. (CC0, CC1)



Combinational SCOAP…. Example(7)

Observability through Level 1

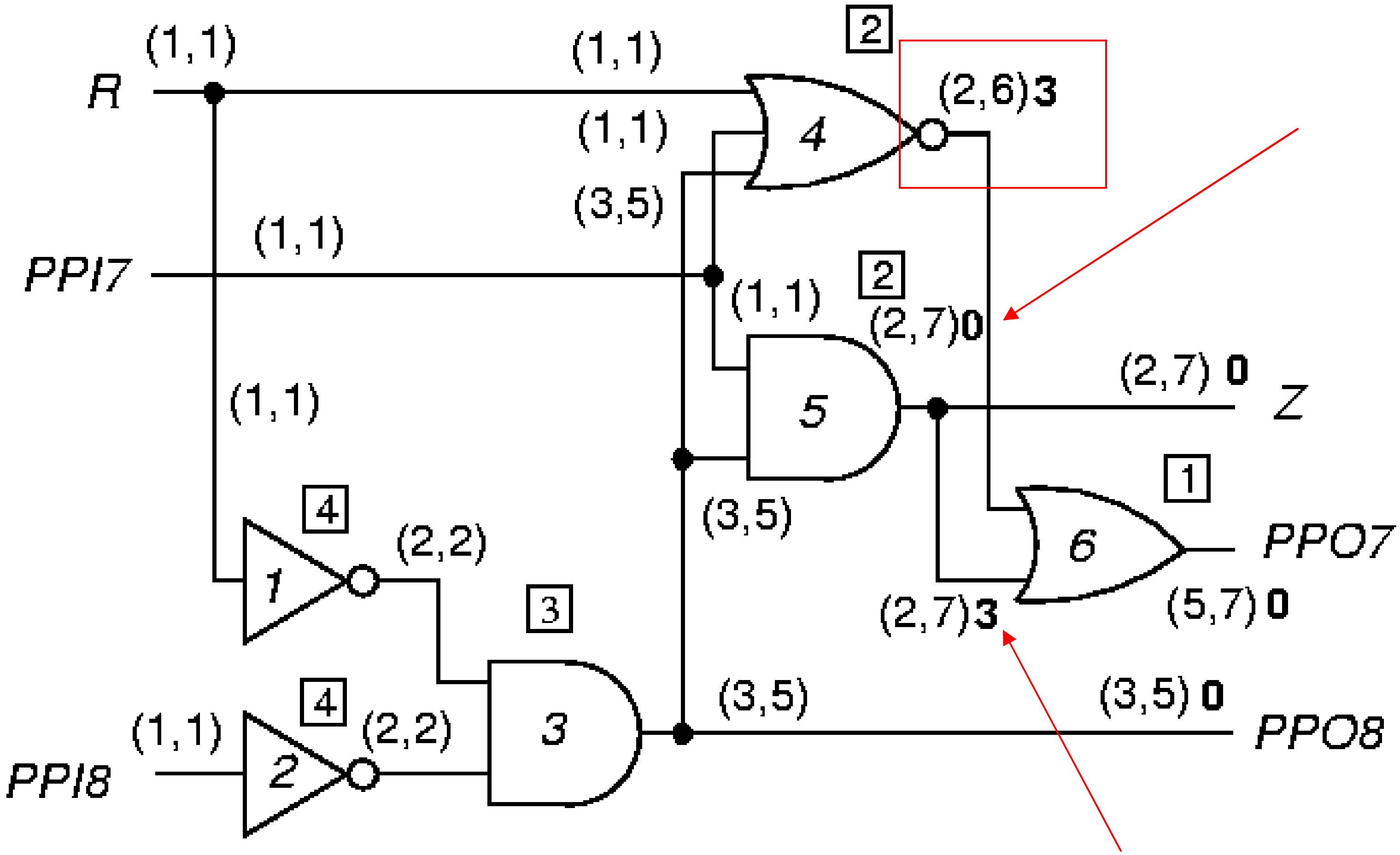
- Number in square box is level from *primary outputs*
- (CC0, CC1) **CO**



Combinational SCOAP…. Example(8)

Observability through Level 2

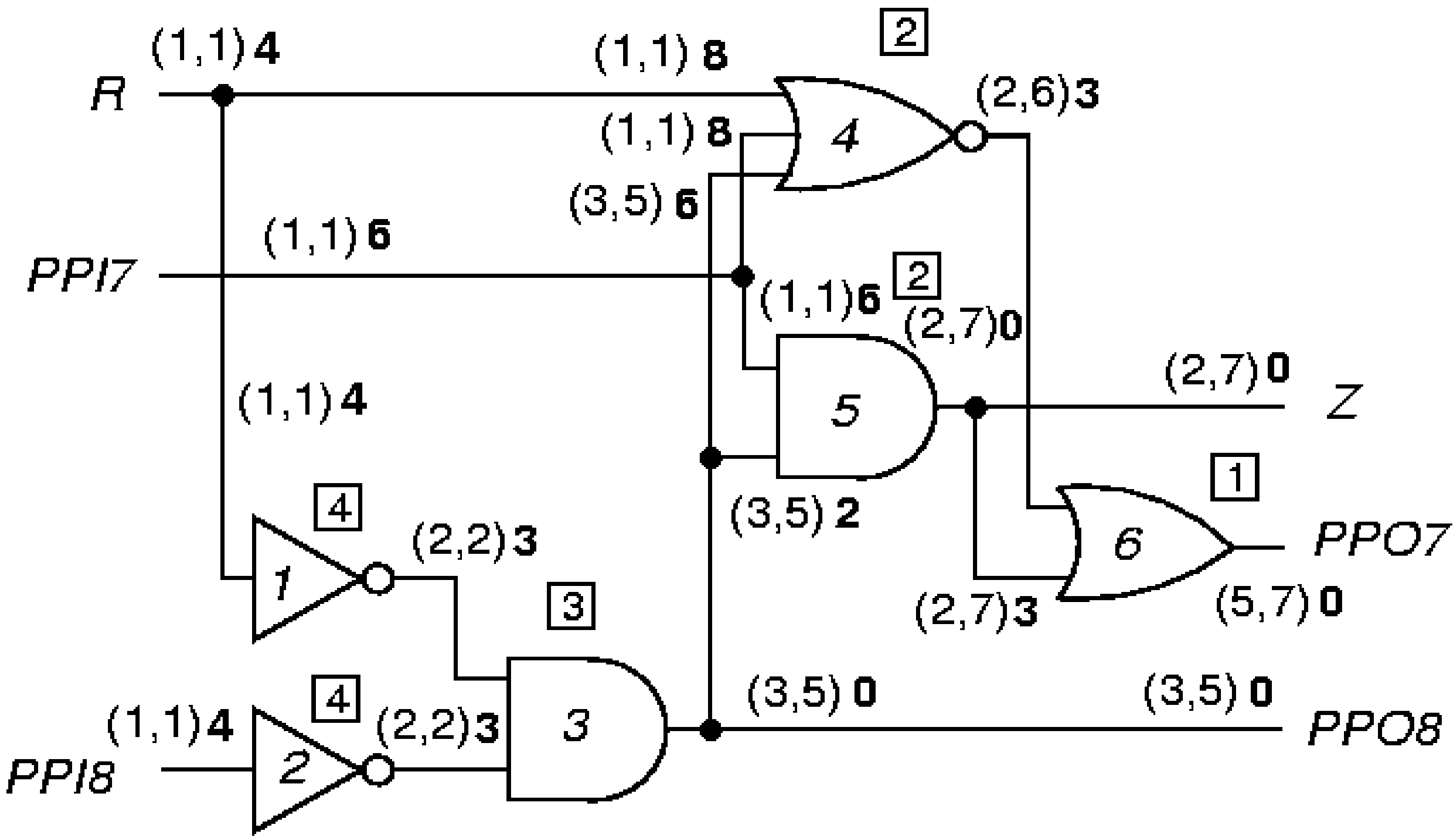
- Number in square box is level from *primary outputs*
- (CC0, CC1) **CO**



Combinational SCOAP…. Example(9)

Final observability

- Number in square box is level from *primary outputs*
- (CC0, CC1) **CO**

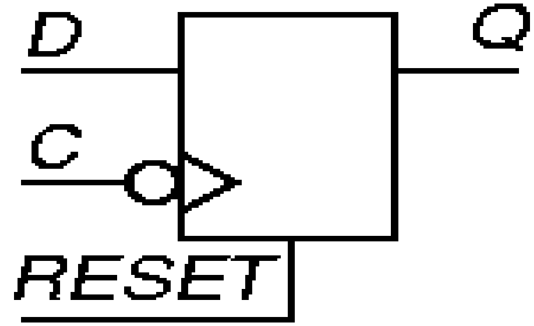


Combinational v Sequential Measure

- **Combinational**
  - Increment *CC0*, *CC1*, *CO* whenever you pass through a gate, either forwards or backwards
- **Sequential**
  - Increment *SC0*, *SC1*, *SO* only when you pass through a flip-flop, either forwards or backwards, to *Q, Q*, D, CL, SET,* or *RESET*
  - 1 is **not added** when we move from one level of logic to another one (without passing a FF)
- Both
  - Must **iterate on feedback loops** until controllabilities stabilize

Sequential SCOAP: D Flip-Flop Equations

- Assume a synchronous RESET.



- Controlling Q to 1
  - **CC1(Q) = CC1(D) + CC1(C) + CC0(C) + CC0(RESET)**
  - ***SC1(Q) = SC1(D) + SC1(C) + SC0(C) + SC0(RESET) + 1***
- Controlling Q to 0
  - ***CC0(Q) = min [CC1(RESET) + CC1(C) + CC0(C), CC0(D) + CC1(C) + CC0(C)]***
  - **SC0(Q) = min [*SC1(RESET)* + *SC1(C)* + *SC0(C)*, *SC0(D) + SC1(C) + SC0(C)*] *+1***
- Observing D at Q
  - ***CO (D) = CO(Q) + CC1(C) + CC0(C) + CC0(RESET)***
  - ***SO (D)*= CO(Q) + CC1(C) + CC0(C) + CC0(RESET) + 1**

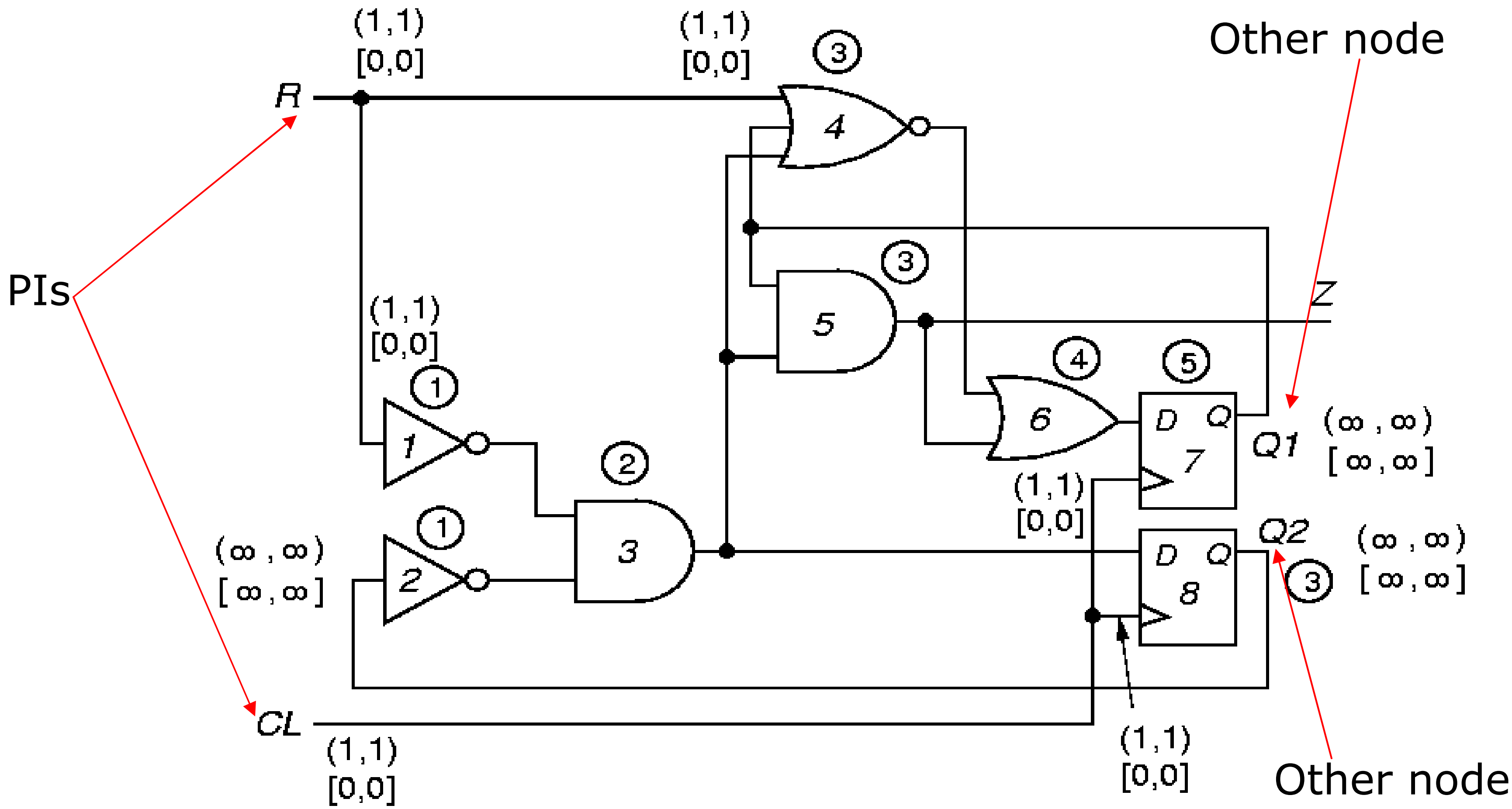
Sequential SCOAP……..Computation

Algorithm 6.2

- For all PIs, *CC0* = *CC1* = 1 and *SC0* = *SC1* = 0
- For all other nodes, *CC0* = *CC1* = *SC0* = *SC1* =∞
- 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
- For other nodes N, CO(N)=SO(N)= ∞
- 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)

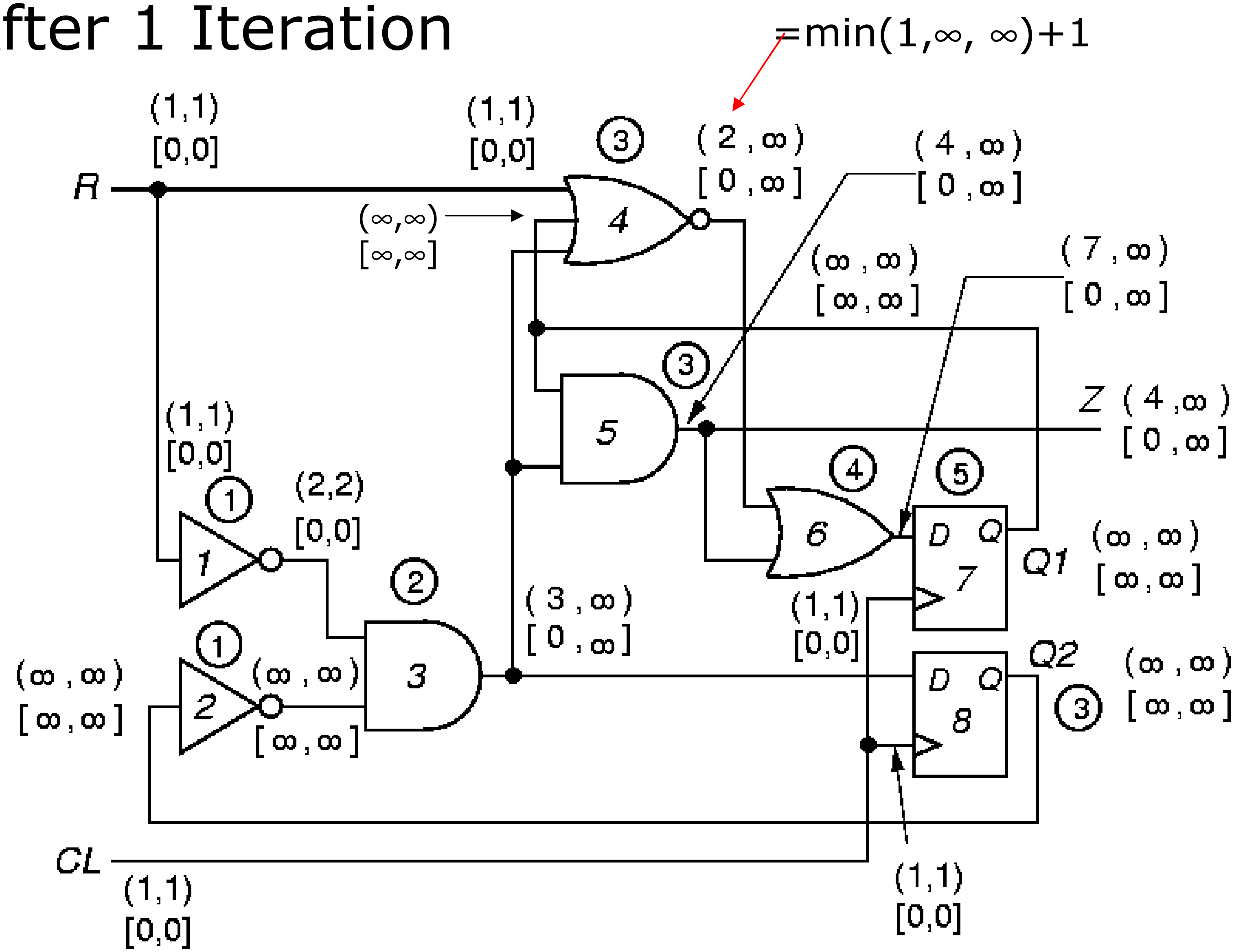
Sequential SCOAP……….Example(1)

- **Initialization**
  - Combinational: (CC0, CC1)**CO**
  - Sequential**:** [SC0, SC1]**S0**



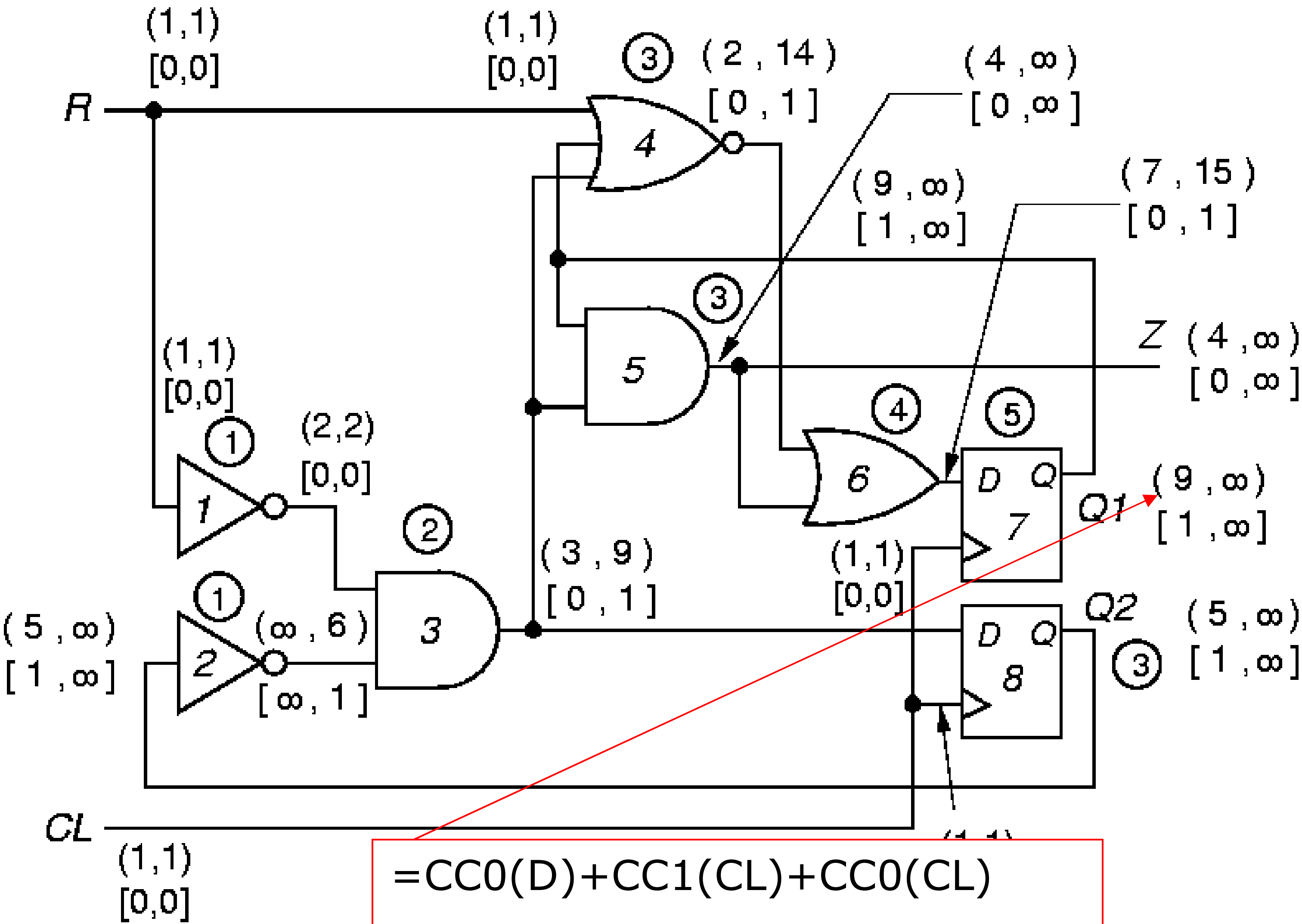
Sequential SCOAP..........Example(2)

- After 1 Iteration



Sequential SCOAP……….Example(3)

- After 2 Iteration

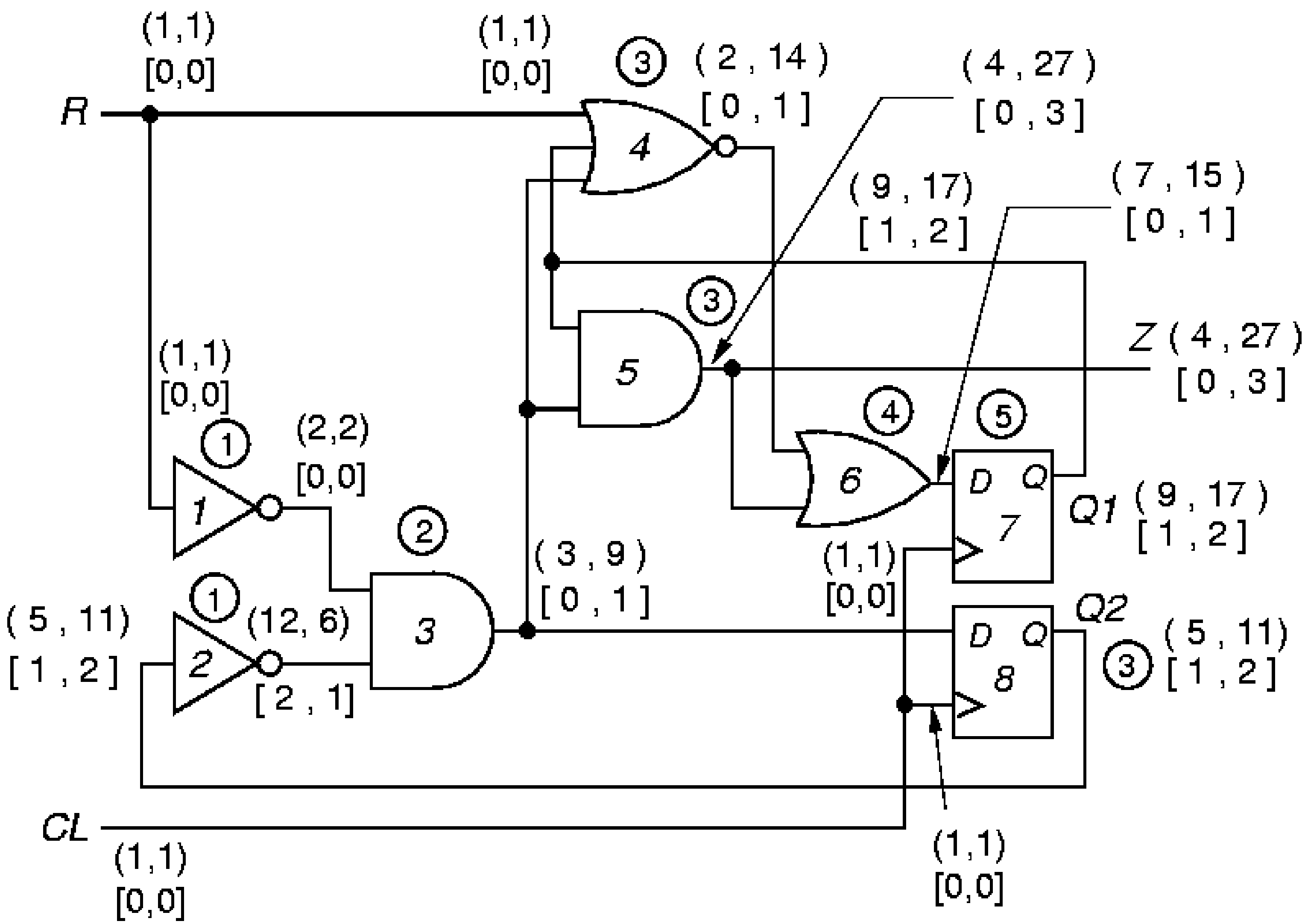


=CC0(D)+CC1(CL)+CC0(CL)

= 7+1+1 = 9

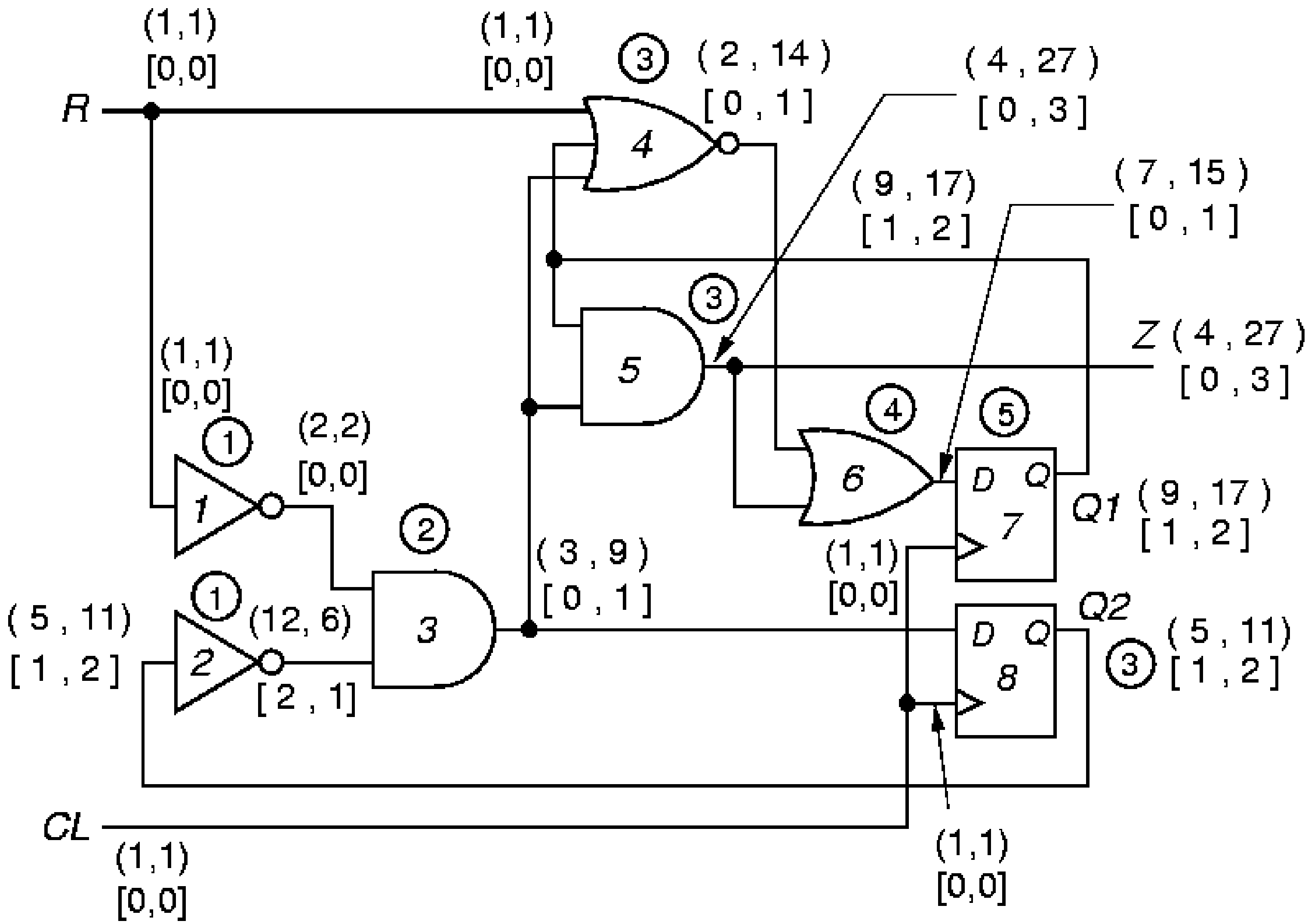
Sequential SCOAP……….Example(4)

- After 3 Iteration



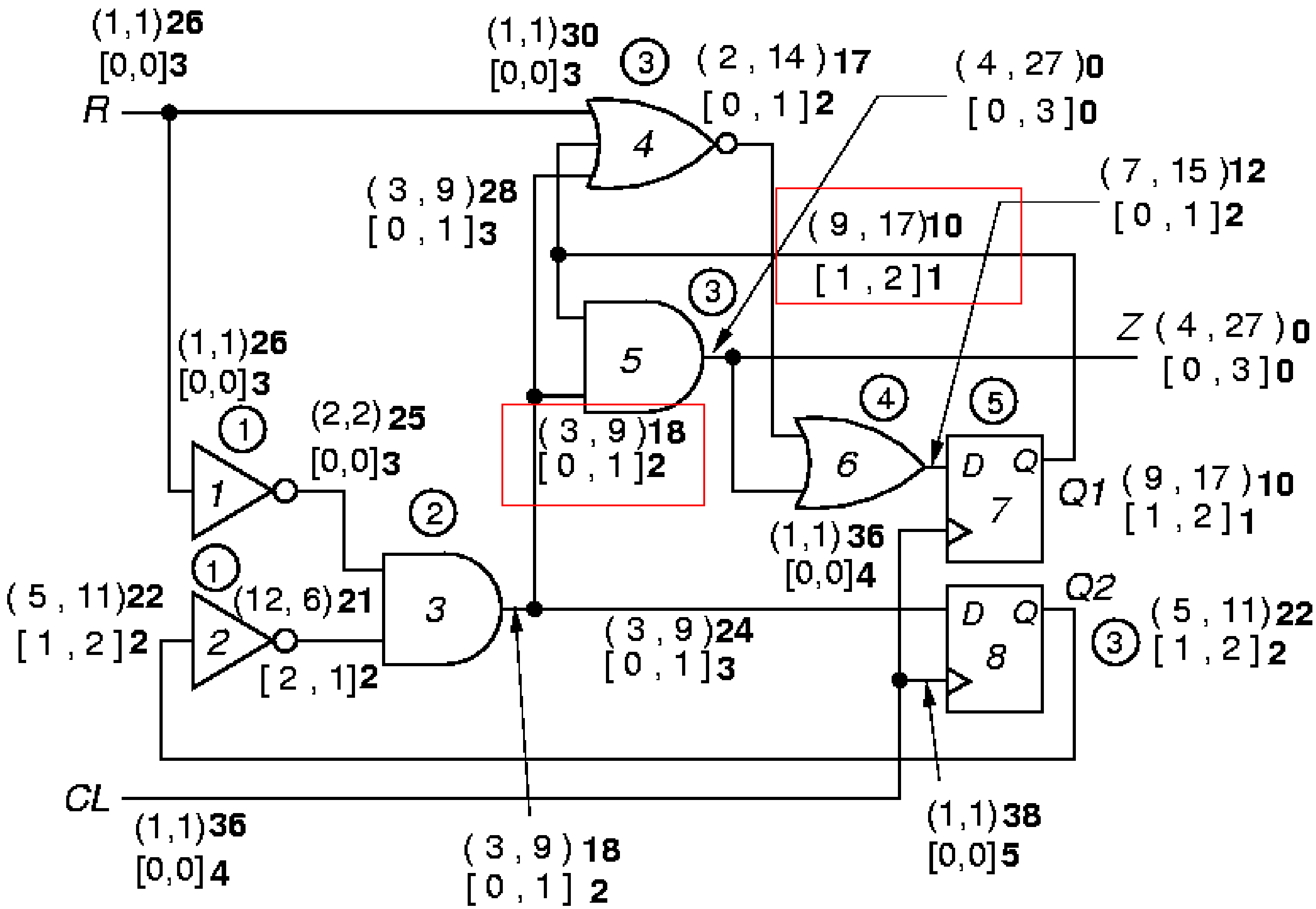
Sequential SCOAP……….Example(5)

- Stable Sequential Measures



Sequential SCOAP……….Example(6)

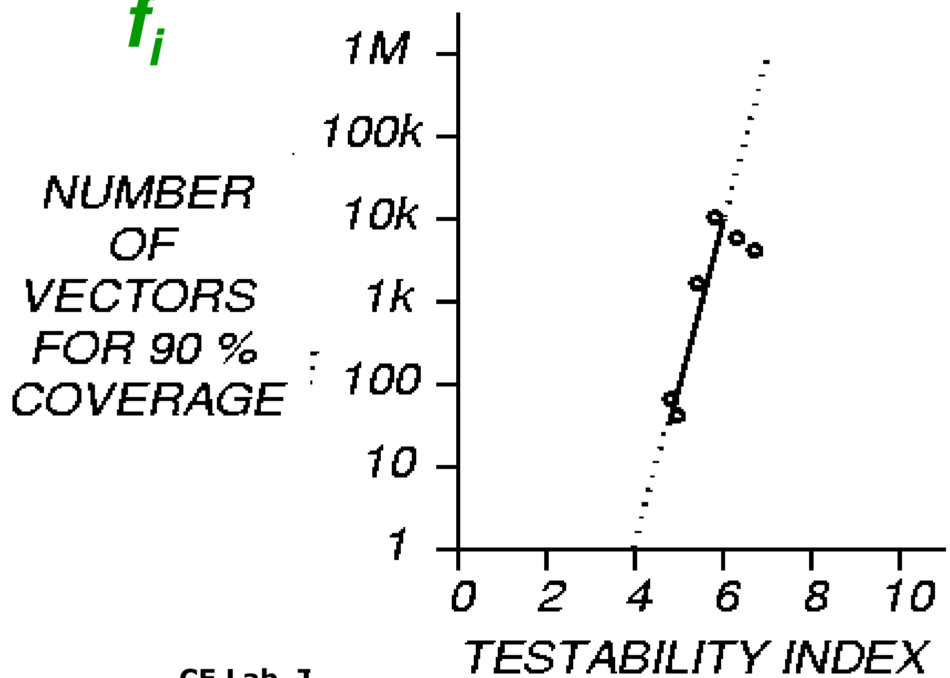
- Final Sequential Observabilities
  - Combinational: (CC0, CC1)**CO**
  - Sequential**:** [SC0, SC1]**S0**



SCOAP for Test Vector Length Prediction

- SCOAP can be used to predict the length of the test vector set
  - The testability of the stuck at faults at node x are:
  - T (x sa0) = $CC1$ (x) + CO (x)
  - T (x sa1) = $CC0$ (x) + CO (x)
  - **Testability index = log $\sum_{f_i}$ T (f $_i$)**

- # of test vectors v. Test. index



High Level Testability*

- Improve the testability at the behavior level by designing register-transfer structure satisfying:
  - **Easy controllability**: one can easily apply a test sequence to sensitize a fault from PI
  - **Easy oberevability**: one can easily propagate the fault effect to PO

- *Use "Data path control graph* (DPCG)" for circuit
  - Node = register
  - Arc= combinational circuit, an input to REG or REG to output
  - Compute sequential depth -- # arcs along path between PIs, registers, and POs
  - Improve Register Transfer Level Testability with redesign
    - Reduce the sequential depth between REG
    - Whenever possible, allocate a REG to a least one PI or PO variable…

Summary

- Testability approximately measures:
  - Difficulty of setting circuit lines to 0 or 1
  - Difficulty of observing internal circuit lines
- Uses:
  - Analysis of difficulty of testing internal circuit parts
    - Redesign circuit hardware or add special test hardware where measures show bad controllability or observability
  - Guidance for algorithms computing test patterns – avoid using hard-to-control lines
  - Estimation of fault coverage – 3-5 % error
  - Estimation of test vector length

- Next lecture – we will talk about ATPG