EEDG/CE 6303: Testing and Testable Design

Mehrdad Nourani

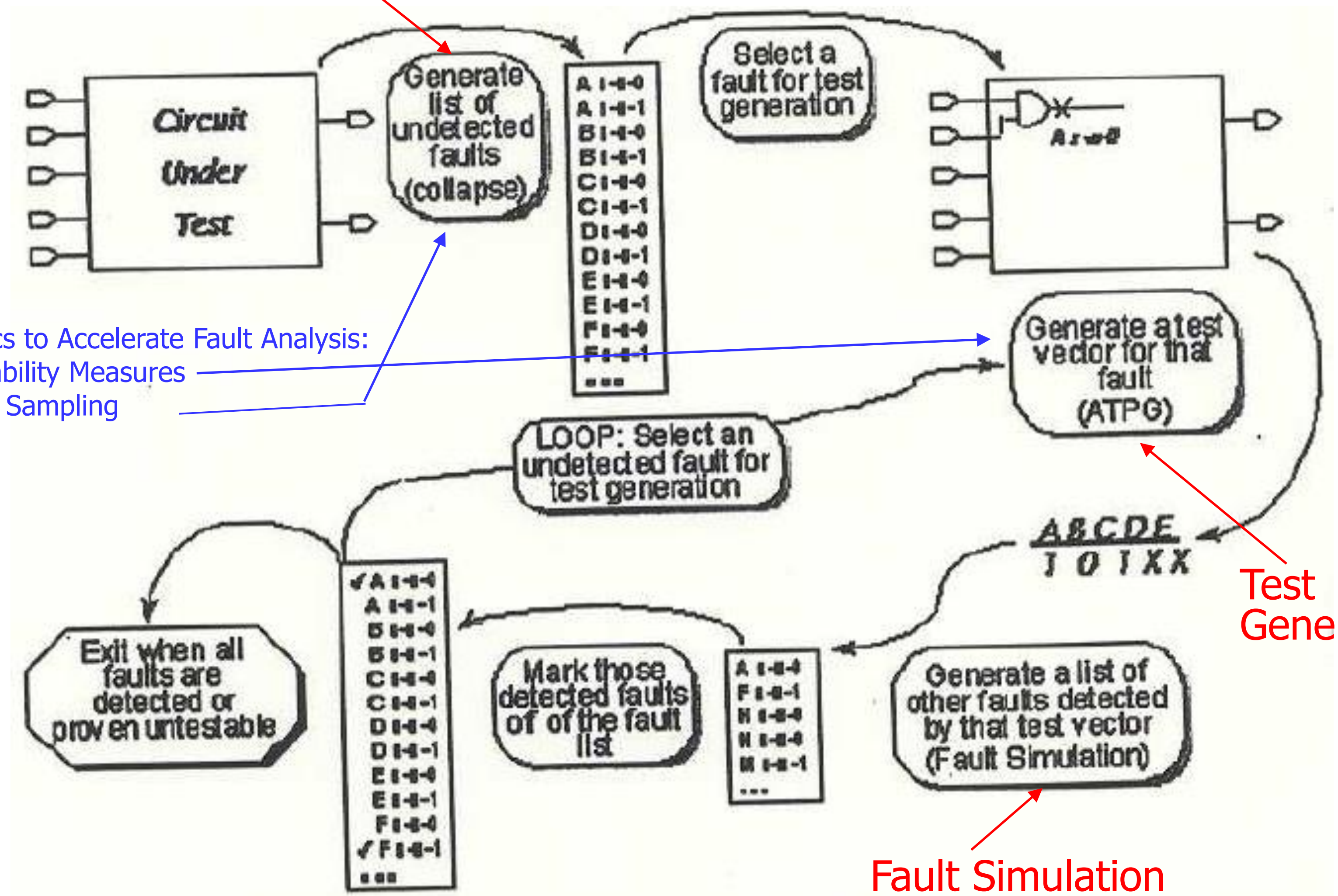
Dept. of ECE
Univ. of Texas at Dallas

Session 05

Acceleration Heuristics for Test Generation

Fault Analysis System (Review)

Fault Collapsing



Heuristics to Accelerate Fault Analysis:

- Testability Measures
- Fault Sampling

Test Generation

Fault Simulation

Accelerating Test Generation

- There are many approaches in the literature trying to speed up the test generation process.
- Two main categories
  - **Deterministic**: Guaranteed to reduce search complexity
  - **Heuristic**: Likely to reduce search complexity
  - Even a deterministic technique may not reduce overall complexity if the complexity required to implement the technique **exceeds** the reduction in search complexity

Testability Measures

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

Origin

- Control theory
- Rutman 1972 -- First definition of controllability
- Goldstein 1979 -- SCOAP
  - First definition of observability
  - First elegant formulation
  - First efficient algorithm to compute controllability and observability
- Parker & McCluskey 1975
  - Definition of Probabilistic Controllability
- Brglez 1984 -- COP
  - 1st probabilistic measures
- Seth, Pan & Agrawal 1985 – PREDICT
  - 1st exact probabilistic measures

Testability Analysis

- 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 and calculate:
    - Exact fault coverage
    - Exact test vectors

SCOAP and Its Metrics

- SCOAP – Sandia Controllability and Observability Analysis Program
- 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 (not discussed here):
  - *SC0*
  - *SC1*
  - *SO*

SCOAP and Its Metrics

- These metrics reflect the **level of difficulty**
- Controllabilities – 1 (easiest) to infinity (hardest)
- Observabilities – 0 (easiest) to infinity (hardest)
- Combinational measures:
  - —Roughly proportional to # circuit lines that must be set to control or observe given line
- Sequential measures (not discussed here):
  - —Roughly proportional to # times a flip-flop must be clocked to control or observe given line

Metrics Computation

- AND gate O/P 0 controllability:

  output_controllability = min (input_controllabilities) + 1

- AND gate O/P 1 controllability:

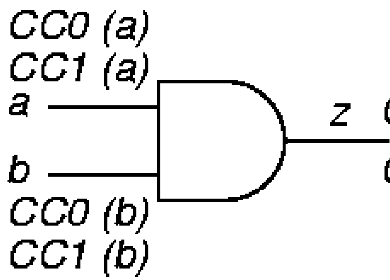
  output_controllability = Σ (input_controllabilities)+ 1

- XOR gate O/P controllability

  output_controllability = min (controllabilities of each input set) + 1

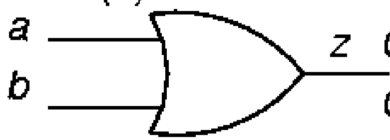
- To observe a gate input, observe output and make other input values non-controlling
- To observe a fanout stem, observe it through branch with best observability
- Fanout Stem observability:
  - Σ or min (some or all fanout branch observabilities)

Controllability Examples



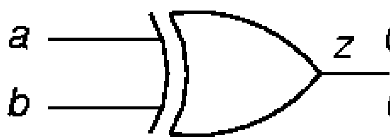
$CC0\ (z) = \min\ (CC0\ (a), CC0\ (b)) + 1$

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



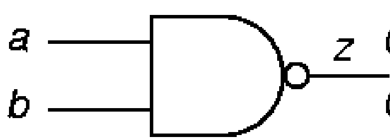
$CC0\ (z) = CC0\ (a) + CC0\ (b) + 1$

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



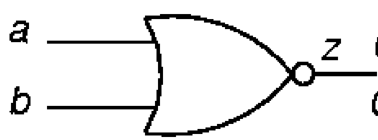
$CC0\ (z) = \min\ (CC0\ (a) + CC0\ (b), CC1\ (a) + CC1\ (b)) + 1$

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



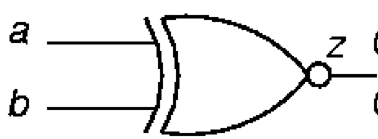
$CC0\ (z) = CC1\ (a) + CC1\ (b) + 1$

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



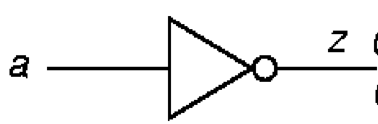
$CC0\ (z) = \min\ (CC1\ (a), CC1\ (b)) + 1$

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



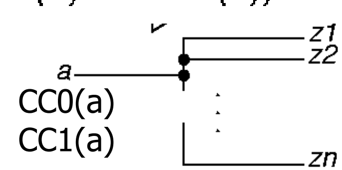
$CC0\ (z) = \min\ (CC1\ (a) + CC0\ (b), CC0\ (a) + CC1\ (b)) + 1$

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



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

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



$CC0(z_i)=CC0(a)$

$CC1(z_i)=CC1(a)$

Observability Examples

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

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

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

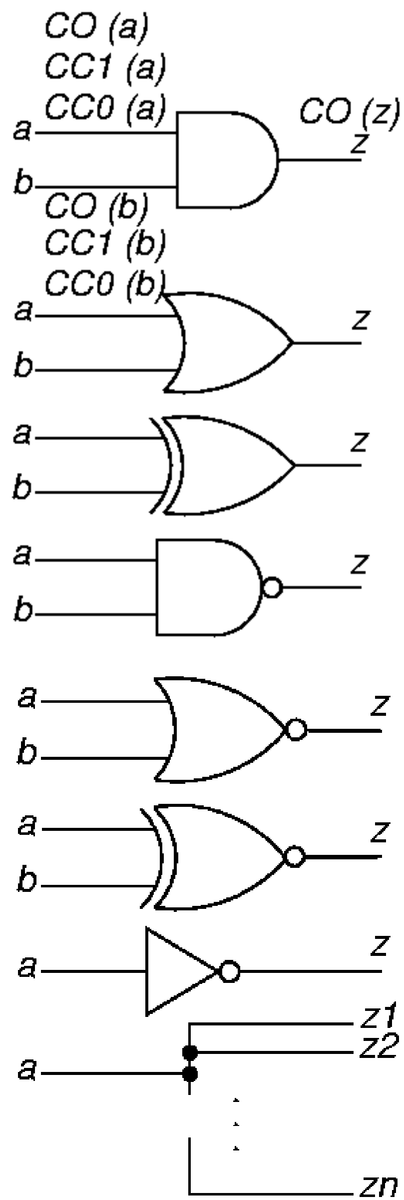
$$CO(a) = CO(z) + CC1(b) + 1$$
$$CO(b) = CO(z) + CC1(a) + 1$$

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

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

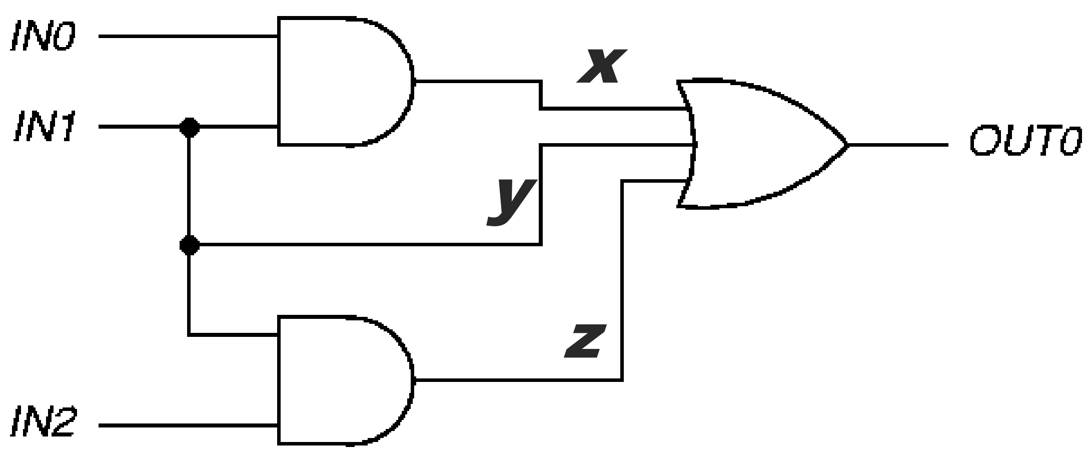
$$CO(a) = CO(z) + 1$$

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



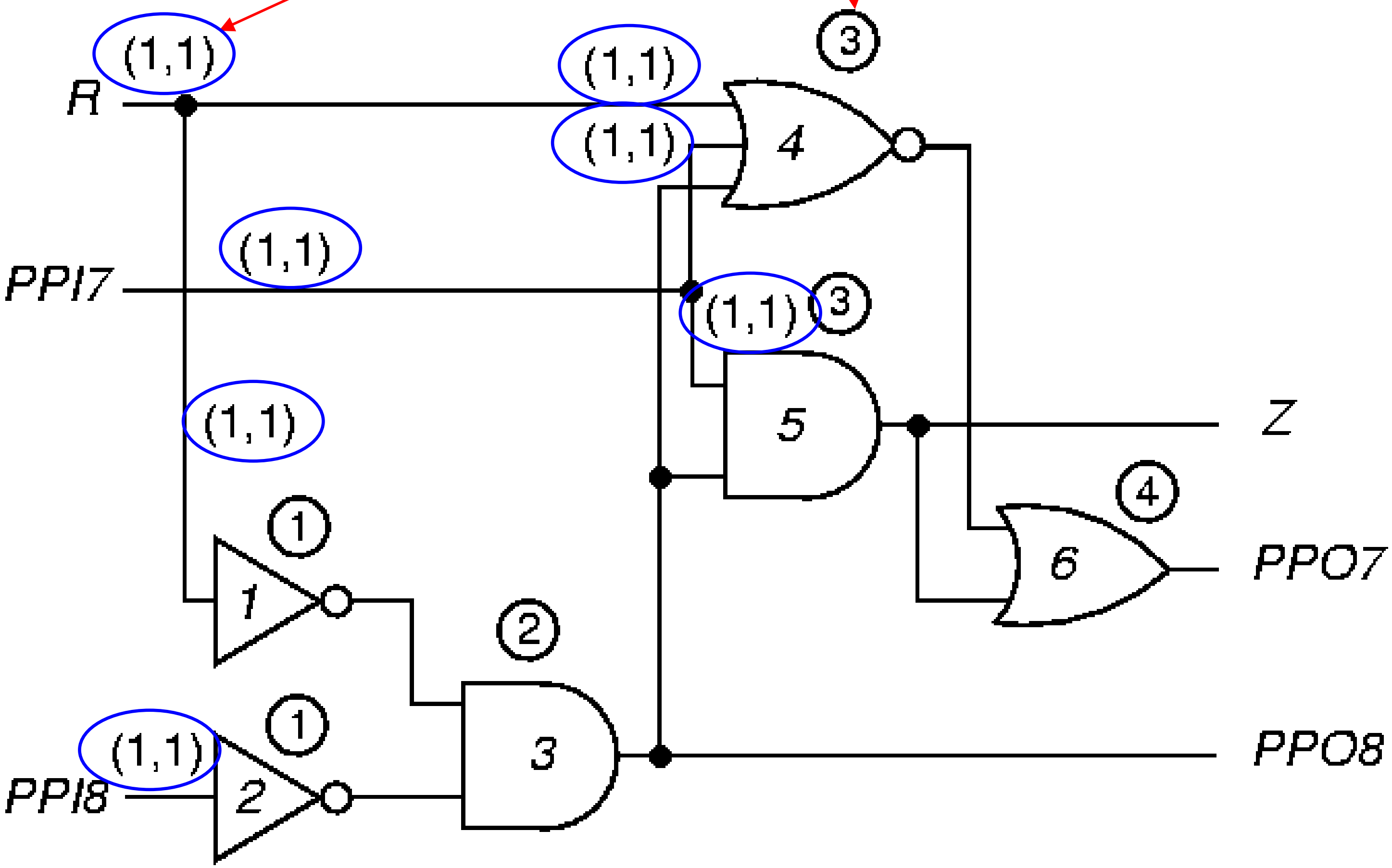
Errors Due to Reconverging Fanouts

- Exact computation of measures is NP-complete and impractical.
- 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

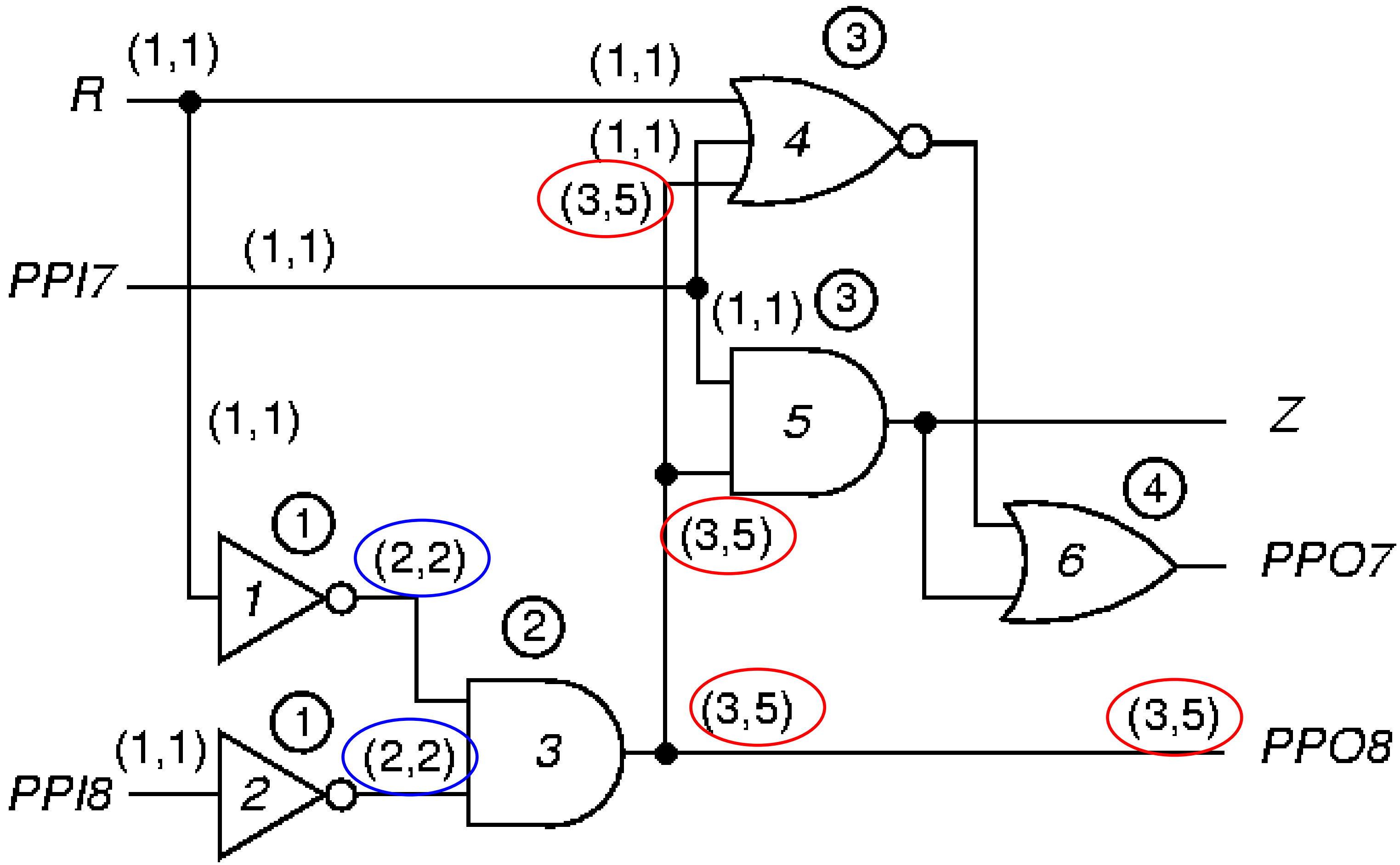


Controllability Example - Level 0

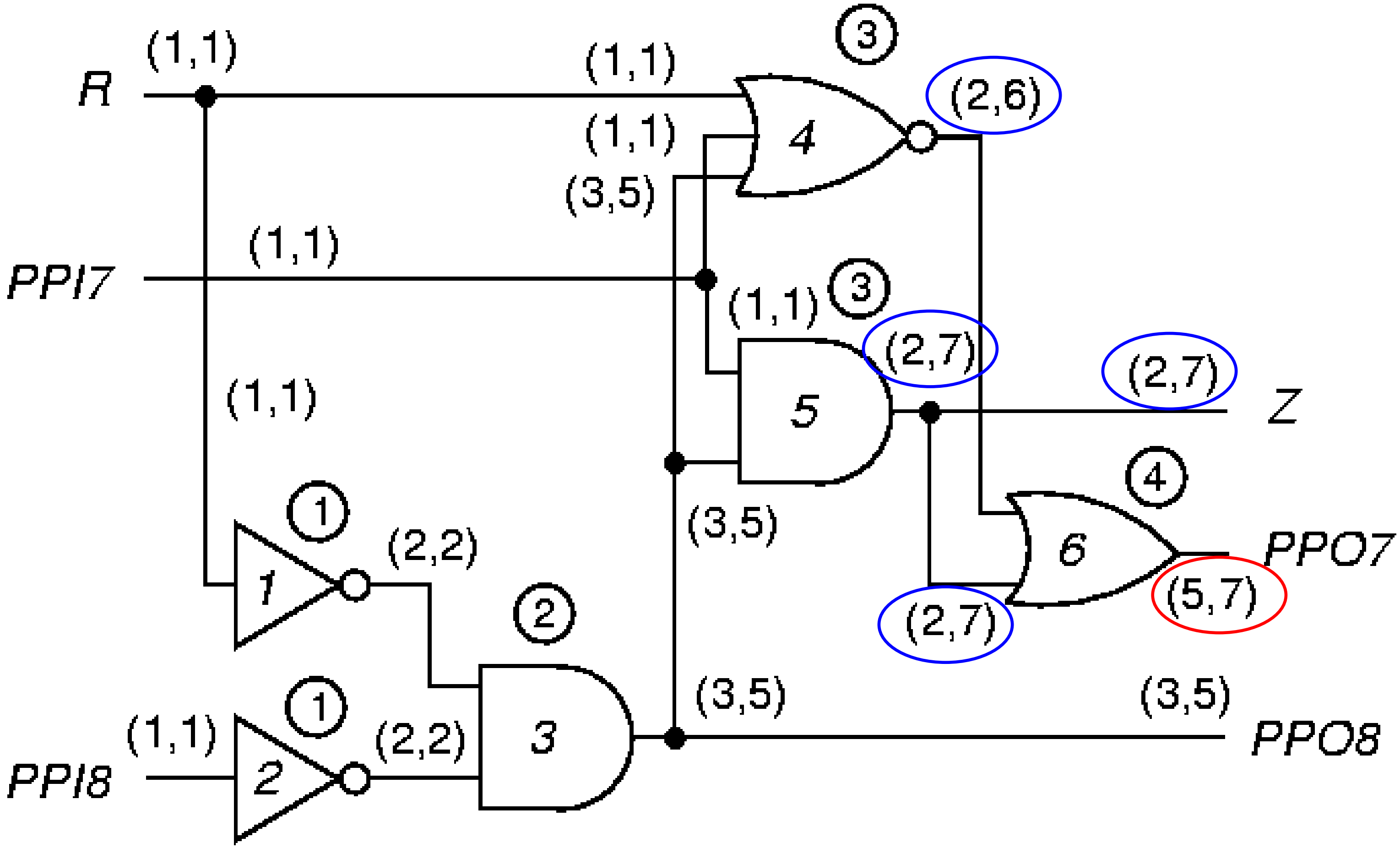
- Circled numbers give level number.
- Notation is: (CC0, CC1)



Controllability Example - Level 1 and 2

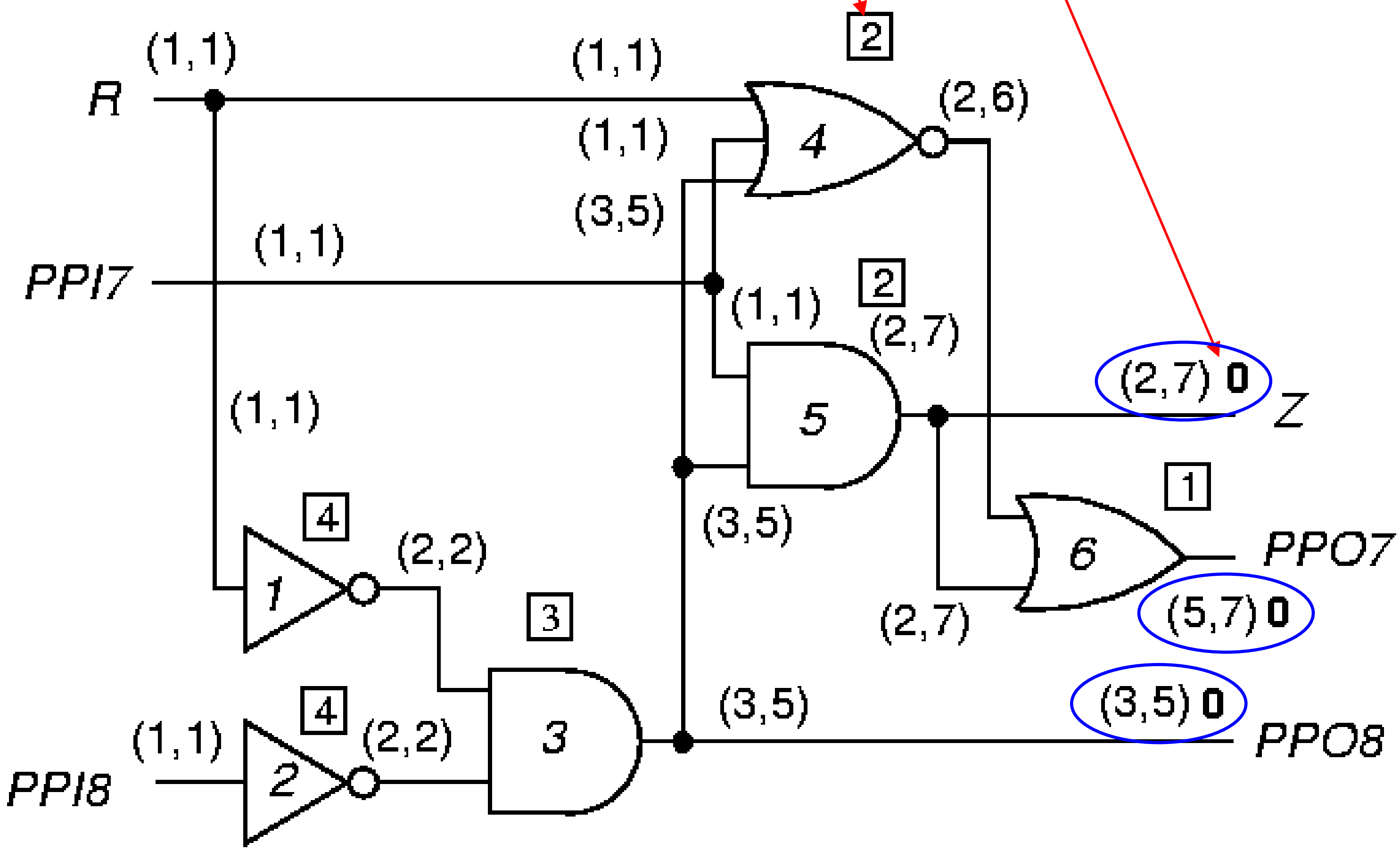


Controllability Example - Level 3 and 4

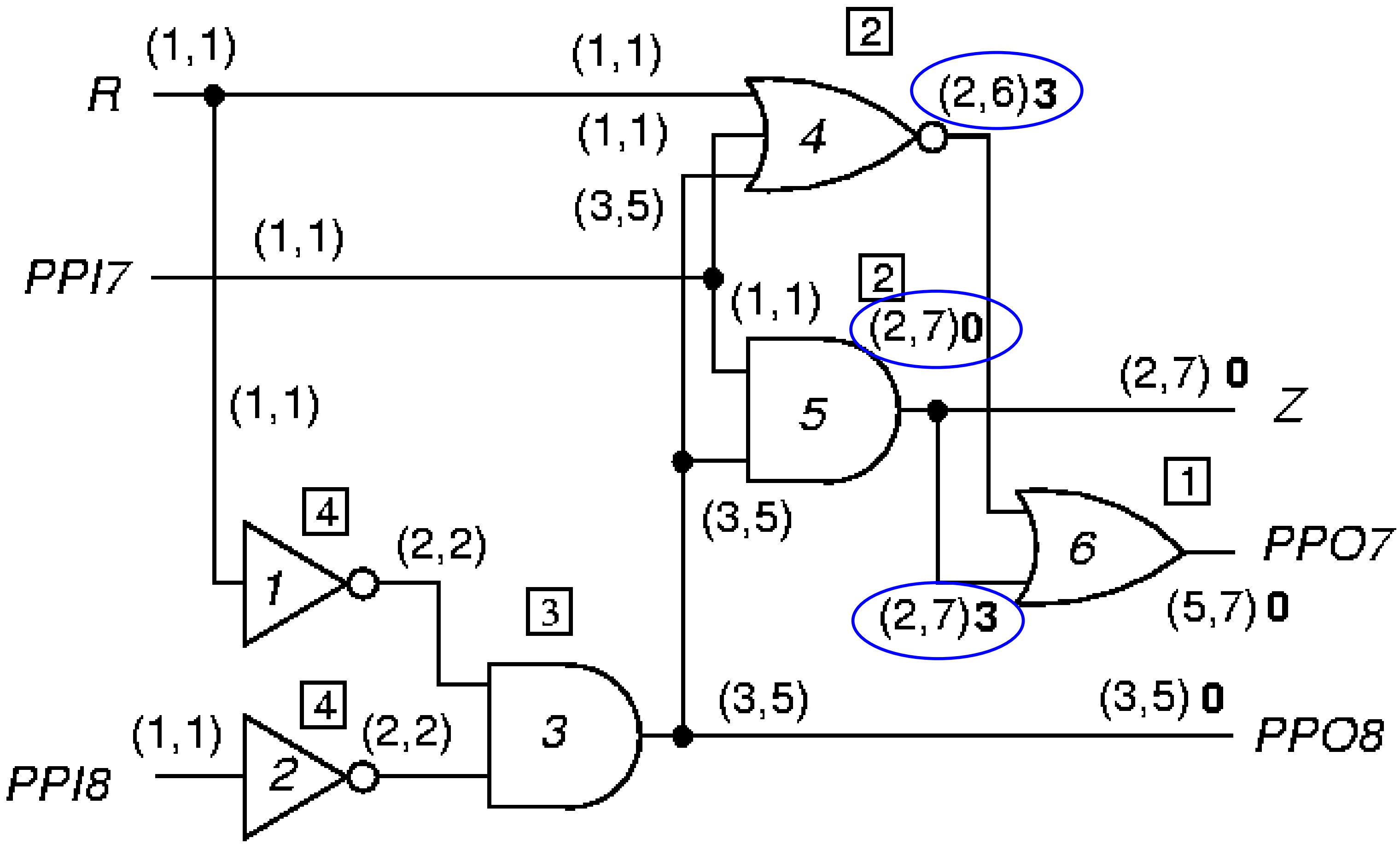


Observability Example – Level 1

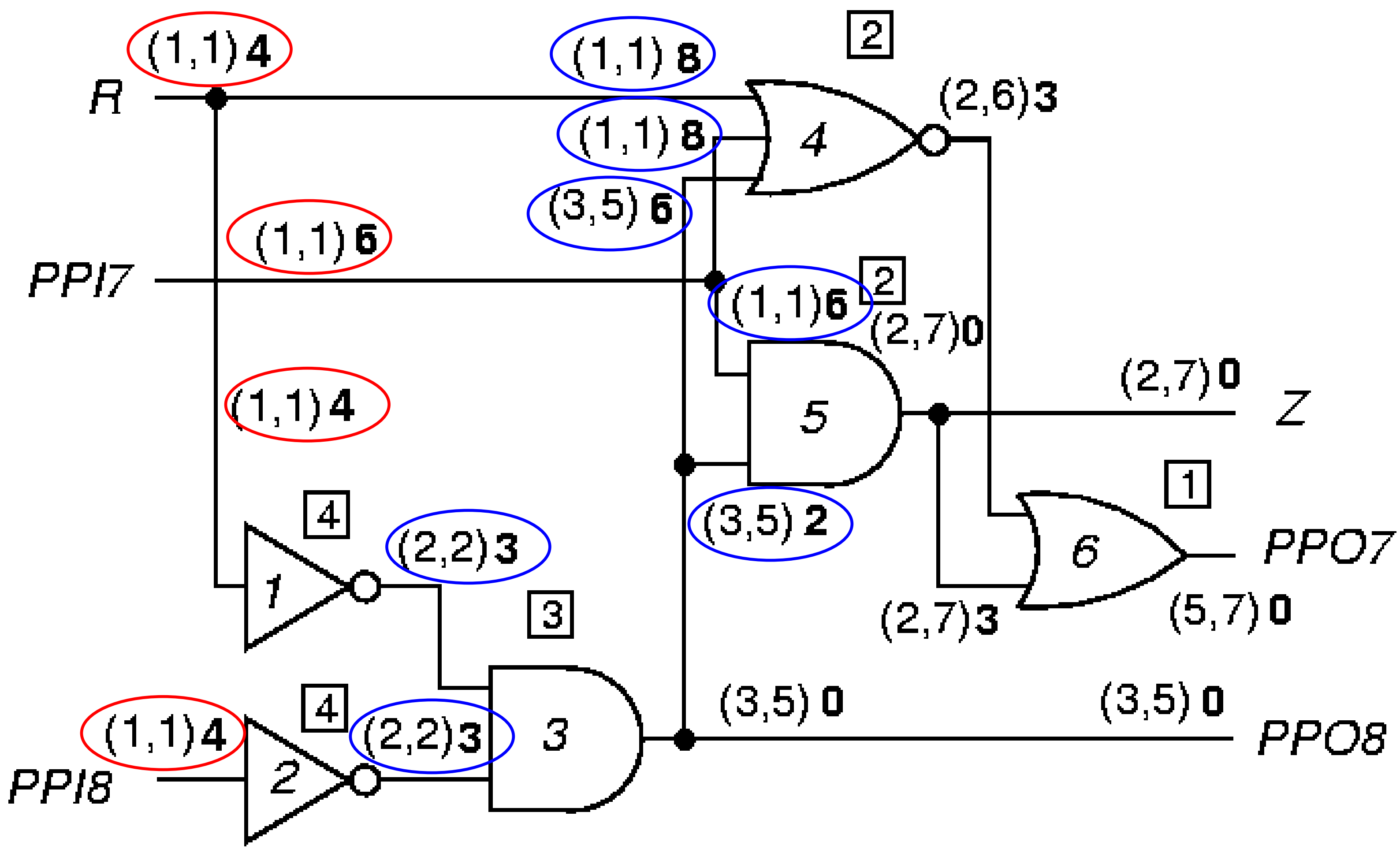
- Squared numbers give level number.
- Notation is: (CC0, CC1) **CO**



Observability Example – Level 2



Observability Example – Level 3 & 4



Sequential Measure Differences

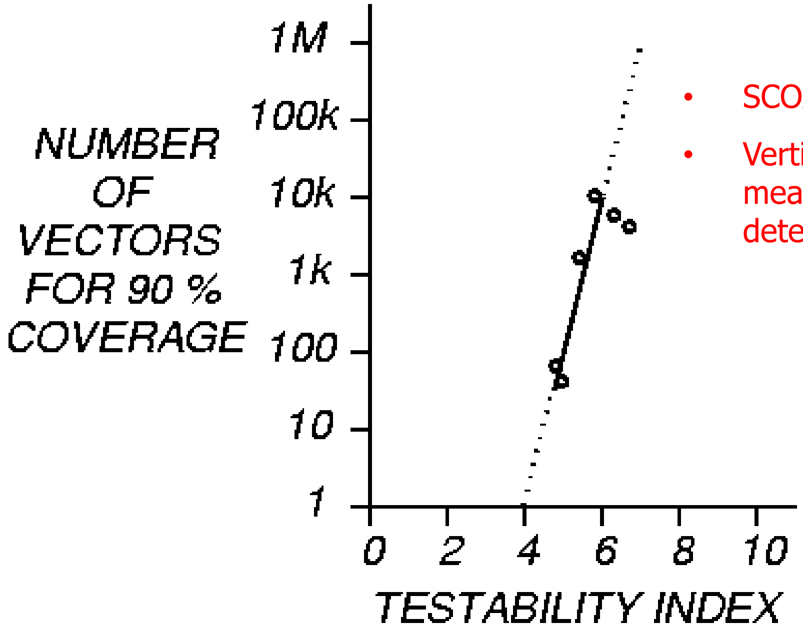
- Combinational
  - Increment *CC0, CC1, CO* whenever you pass through a gate, either forwards or backwards
- Sequential (not discussed here)
  - Increment *SC0, SC1, SO* only when you pass through a flip-flop, either forwards or backwards, to Q, $\overline{Q}$, D, C, *SET*, or *RESET*
- Both
  - Must iterate on feedback loops until controllabilities stabilize

Testability Computation Algorithm

1. For all PIs, *CC0* = *CC1* = 1 [*SC0* = *SC1* = 0]
2. For all other nodes, *CC0* = *CC1* = ∞ [*SC0* = *SC1* = ∞]
3. Go from PIs to POS, using *CC* [*SC*] equations to get controllabilities [-- Iterate on loops until *SC* stabilizes -- convergence guaranteed]
4. For all POs, set *CO* =0 [*SO* = 0]
5. For all other nodes, *CO* = ∞ [*SO* = ∞]
6. Work from POs to PIs, Use *CO* [*SO*] and controllabilities to get observabilities. Fanout stem *CO* = min branch (CO_i), [*SO* = min branch (SOi)]
7. If a *CC* or *CO* [*SC* or *SO*] is ∞ , that node is uncontrollable (or unobservable)

Test Vector Length Prediction

- To detect a fault at x, we need to
  - Set x to the opposite value from the fault
  - Observe x at PO
- Compute *testabilities* for stuck-at faults
  - T(x sa0) = $CC1(x) + CO(x)$
  - T(x sa1) = $CC0(x) + CO(x)$
  - *Testability index* = log Σ T (f_i) , i.e. computed for all faults f_i (after collapsing – e.g. to avoid considering equivalent faults multiple times)



- SCOAP is a Linear complexity algorithm
- Vertical axis is exponential (higher index means much more difficulty and patterns to detect all faults)