Testability Measures

Testability measures used to get 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

This knowledge can be used to:

- Provided analysis of difficulty of testing internal circuit parts, might require redesigning or addition of special testing hardware
- Provides guidance for algorithms performing test pattern generation, avoid using hard-to-control lines
- Provides an estimation of fault coverage
- Provides an estimation of test vector length

Controllability: difficulty in setting a particular circuit node to 0 or 1.

Observability: difficulty of observing the state of a logic signal.

Testability Measures

Testability analysis attributes:

- Involves circuit topology analysis, but no test vectors
- It has linear complexity, otherwise it is pointless and one might as well use ***automatic test pattern generation (ATPG)*** algorithms

The origin of testability measures is in control theory. Several algorithms have been proposed:

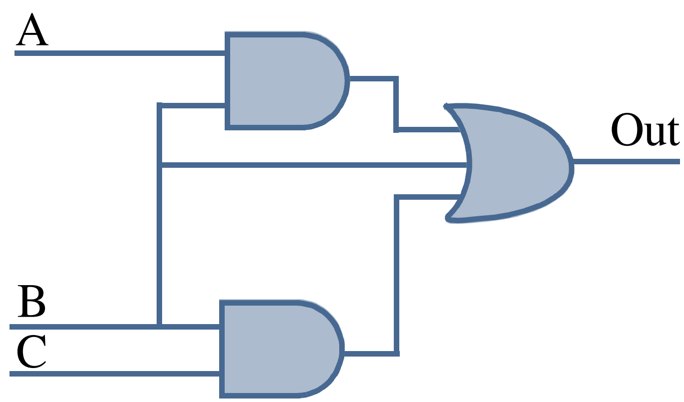
- Rutman 1972: First definition of controllability
- Goldstein 1979: SCOAP
  - First definition of observability
  - First elegant formulation
  - First efficient algorithm to compute controllability and observability
- Parker and McCluskey 1975: Definition of probabilistic controllability
- Brglez 1984: COP
  - First probabilistic measures
- Seth, Pan and Agrawal 1985: PREDICT
  - First exact probabilistic measures

SCOAP (Sandia Controllability/Observability Analysis Program)

Goldstein developed SCOAP testability measures and described a linear complexity algorithm to compute them

Before we go into SCOAP details, a few notes about assumptions

The algorithm assumes that signals at reconvergent fanout stems are independent



For e.g. B fans out to three branches, 2 feeding the 2 AND gates, and 1 to the OR gate

All the three signals reconverge at the OR gate, as outputs of 2 ANDs feed the other two inputs of the OR gate

SCOAP

The assumption of signal independence is the key behind SCOAP's linear time algorithm. However, this reduces its accuracy in predicting which individual faults will remain *undetected* and which will be detected.

SCOAP testability measures:

- *Controllability*: From 1 (easiest) to infinity (hardest).
- *Observability*: From 0 (easiest) to infinity (hardest).

Combinational measures are related to the number of signals that may be manipulated to control or observe a line *l*.

Sequential measures are related to the number of times a FF must be clocked to control or observe a line *l*.

Another approach is to make them probability based, i.e., they range between 0 and 1. Jain and Agrawal address this, in ***PREDICT*** using ***conditional probabilities*** for observability.

SCOAP Controllabilities

Goldstein's algorithm: SCOAP

Consists of 6 numerical measures for each signal (*l*) in the circuit:

- Combinational *0-controllability*, ***CC0(l)***; Sequential *0-controllability*, ***SC0(l)***
- Combinational *1-controllability*, ***CC1(l)***; Sequential *1-controllability*, ***SC1(l)***
- Combinational *observability*, ***CO(l)***; Sequential *observability*, ***SO(l)***

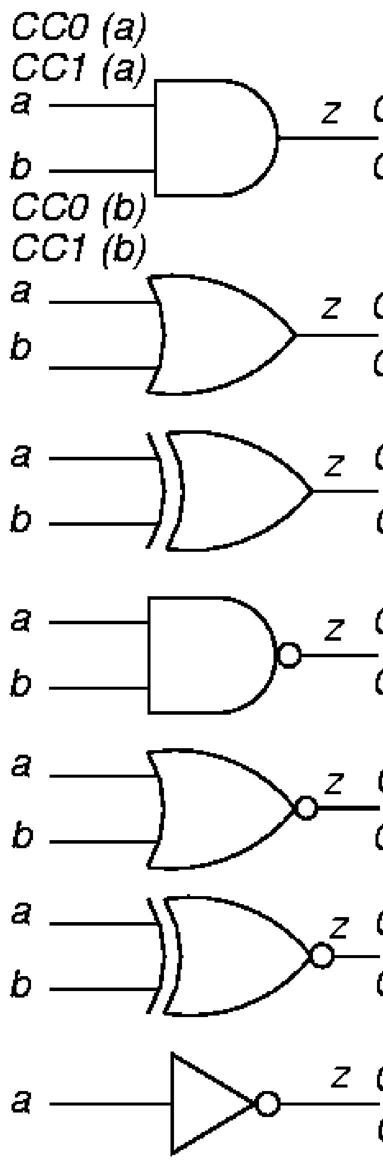
We will focus mainly on combinational circuits.

Controllabilities: Set Primary Input (PI) controllabilities to 1, progress from PIs to Primary Outputs (POs), add 1 to account for logic depth.

General rules for setting controllabilities

- If only one input sets gate output:
  output controllability = *min* (input controllabilites) + 1
- If all inputs set gate output:
  output controllability = *sum* (input controllabilities) + 1
- If gate output is determined by multiple input sets, e.g., XOR:
  output controllability = *min*(controllabilities of input sets) + 1

SCOAP Controllability Examples



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

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

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

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

NOR:
$CC0(z) = \min(CC1(a), CC1(b)) + 1$
$CC1(z) = CC0(a) + CC0(b) + 1$

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

NOT:
$CC0(z) = CC1(a) + 1$
$CC1(z) = CC0(a) + 1$

SCOAP Observability

Observabilities: After controllabilities have been computed, set PO observabilities to 0, progress from POs to PIs, add 1 to account for logic depth.

The difficulty of observing a designated input to a gate is the sum of

(1) the output observability

(2) the difficulty of setting all other inputs to *non-dominant* values

(3) and 1 for the logic depth

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

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

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

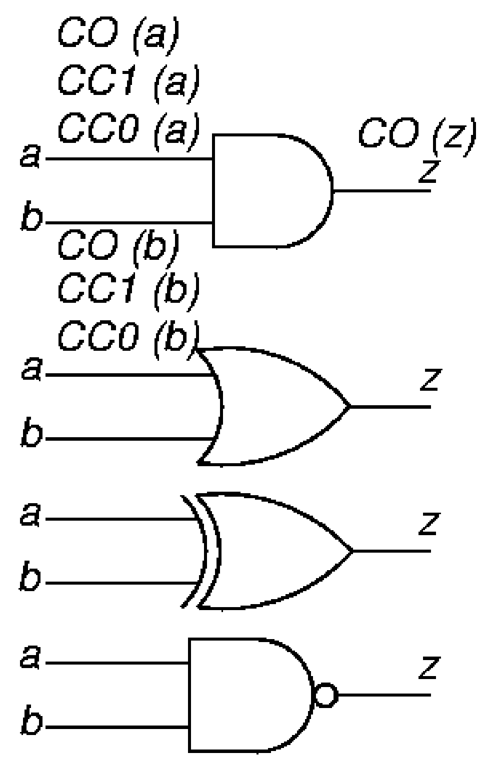
$CO(b) = CO(z) + CC0(a) + 1$

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

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

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

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



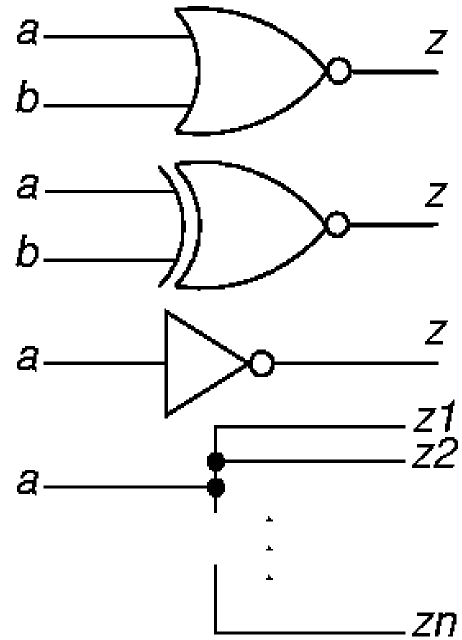
SCOAP Observability Examples

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

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

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

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



The accuracy problem occurs for the computation of the observability of a fanout stem with *n* branches. One attempt is to bound the stem probability by:

- *min*(all fanout branch observabilities)

  The events of observing a signal through each branch are ***independent*.**

- *max*(all fanout branch observabilities)

  They are all ***dependent***, therefore the branch that is hardest to observe is the correct choice.

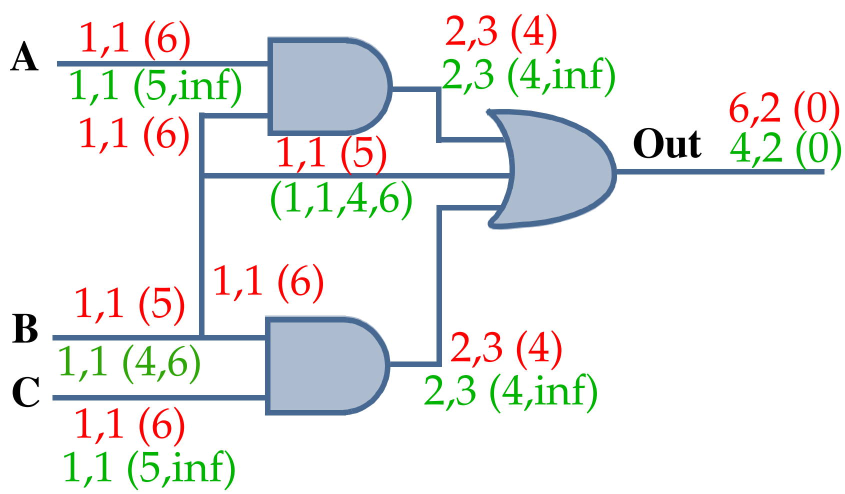
SCOAP Observability Examples

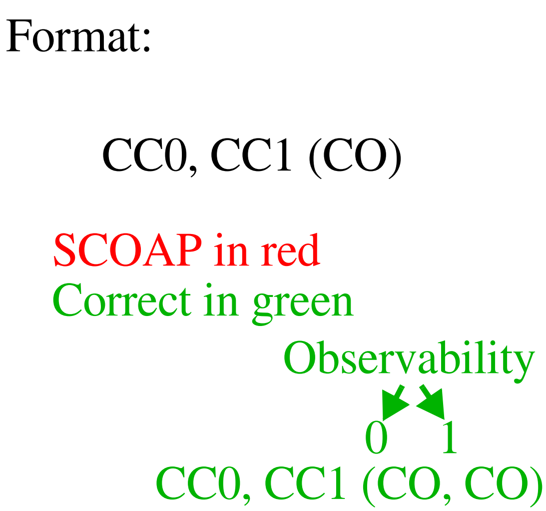
Problem: These ignore the possibility that observing a signal may require its propagation through *some* or *all* fanout branches.

Goldstein uses: CO(*stem*) = *min*(CO(*branches*))

Therefore, observability calculation errors occur and ATPG algorithms which use them may be misled.

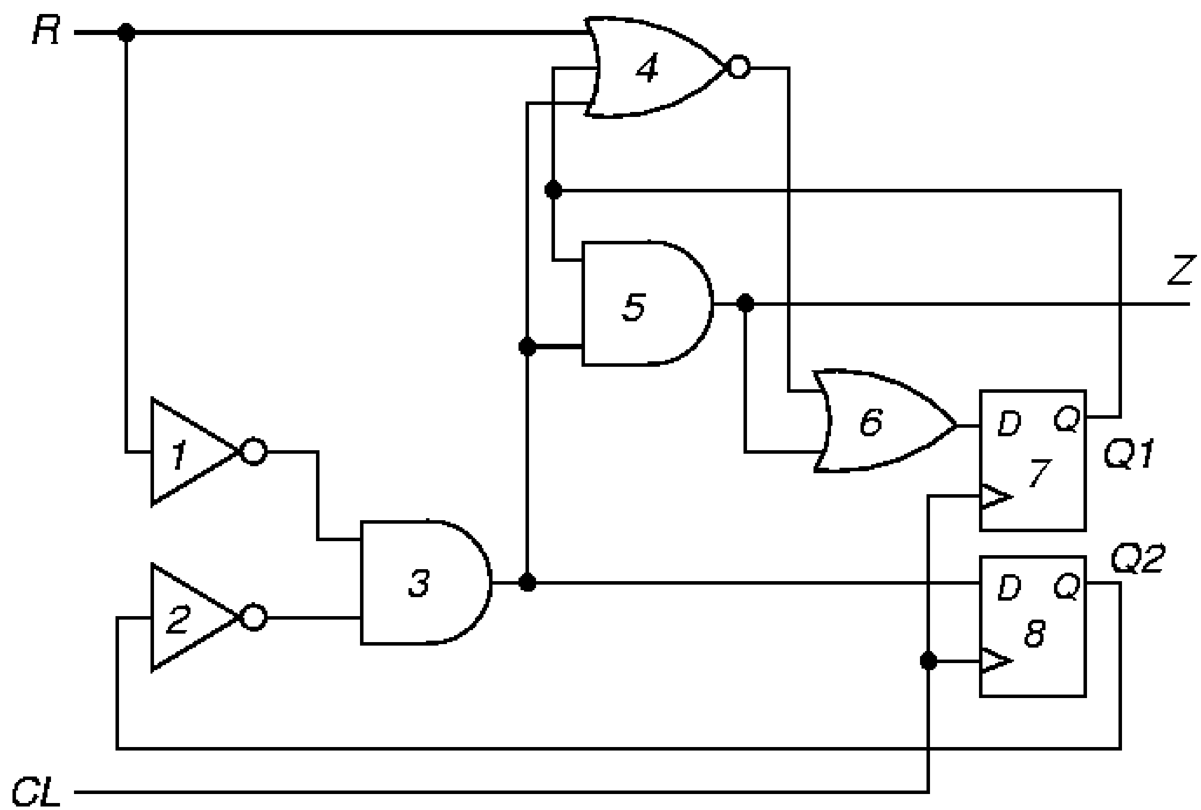
Therefore, Goldstein's algorithm has only O(2**n*) or O(*n*) complexity.





Note that the red numbers are given by the SCOAP algorithm and the green number are the exact values.

SCOAP Example



The above circuit is sequential. However, let's assume that special test hardware is present in the circuit so that

- We can read out the present state of the flip-flop
- We can set the present state of the flip-flop

SCOAP Example

Then for testing purposes, the two flip-flops can be modeled as a pair of primary input, primary output pairs

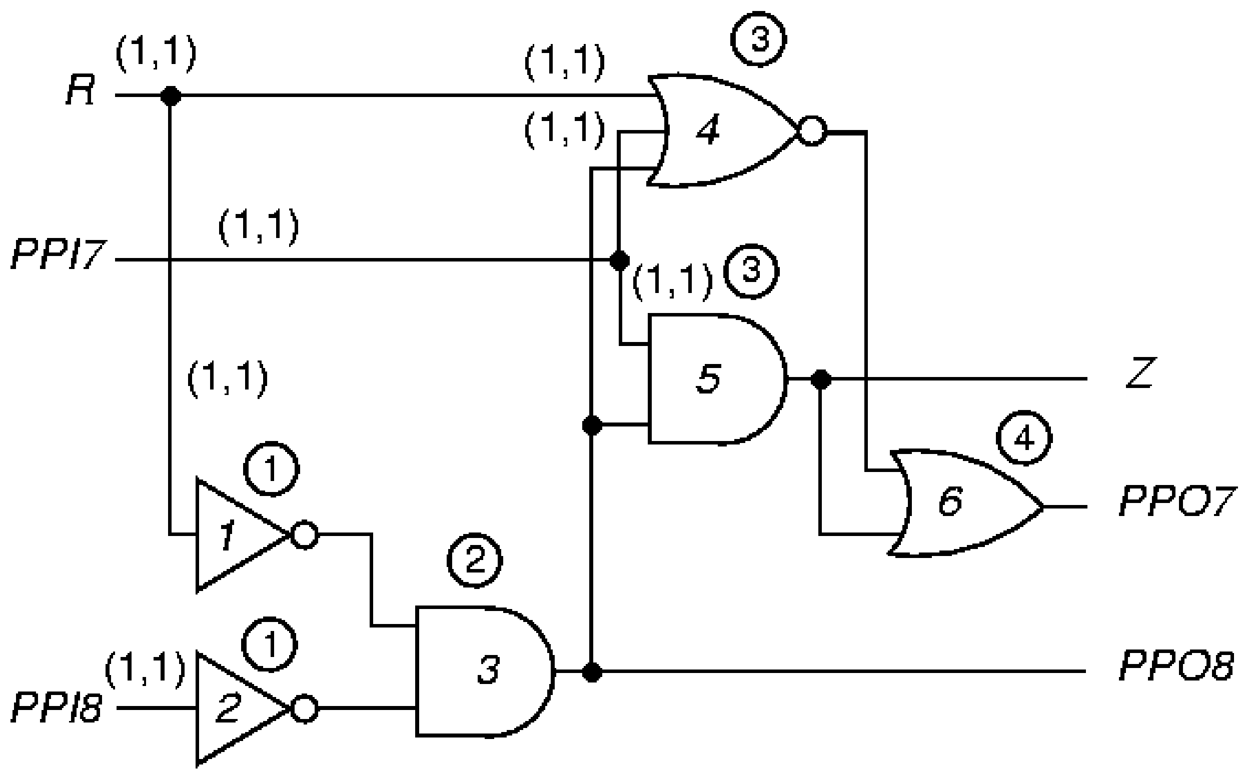
- The D line is referred to as ***pseudo-primary output (PPO)***, since it can be observed
- The Q line is referred to as ***pseudo-primary input (PPI)***, since it can be controlled

Levellization Algorithm

- Label each gate with maximum no. of logic levels from primary inputs (or maximum no. of levels from primary outputs when computing observabilities)
- Assign level no. 0 to all PIs
- For each PI fanout
  - Label that line with the PI level no.
  - Queue logic gate driven by that fanout for evaluation
- While queue is not empty
  - Dequeue next logic gate
  - If all gate inputs have level nos., label the gate with the maximum of them + 1
  - Else, requeue the gate

SCOAP Example

Controllabilities through level 0

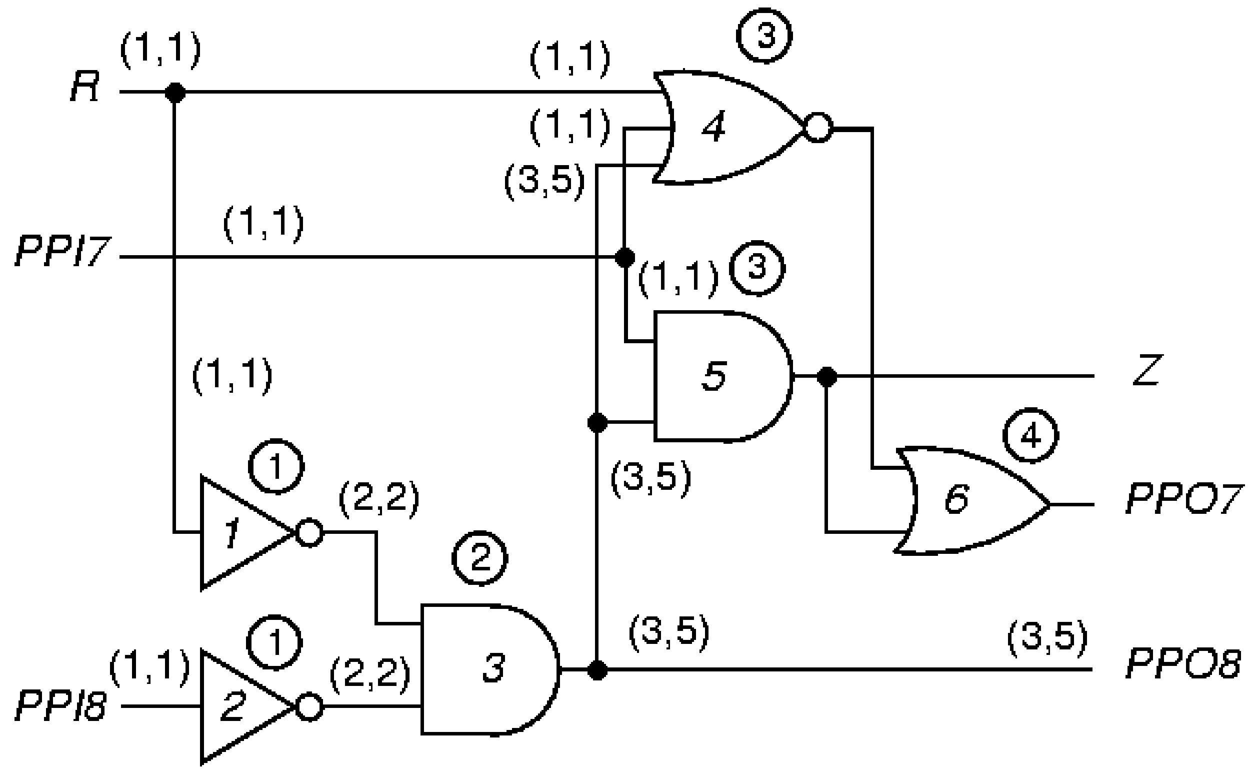


Format

Numbers in circles on top of each gate give level numbers
(CC0, CC1) next to each signal line

SCOAP Example

Controllabilities through level 2

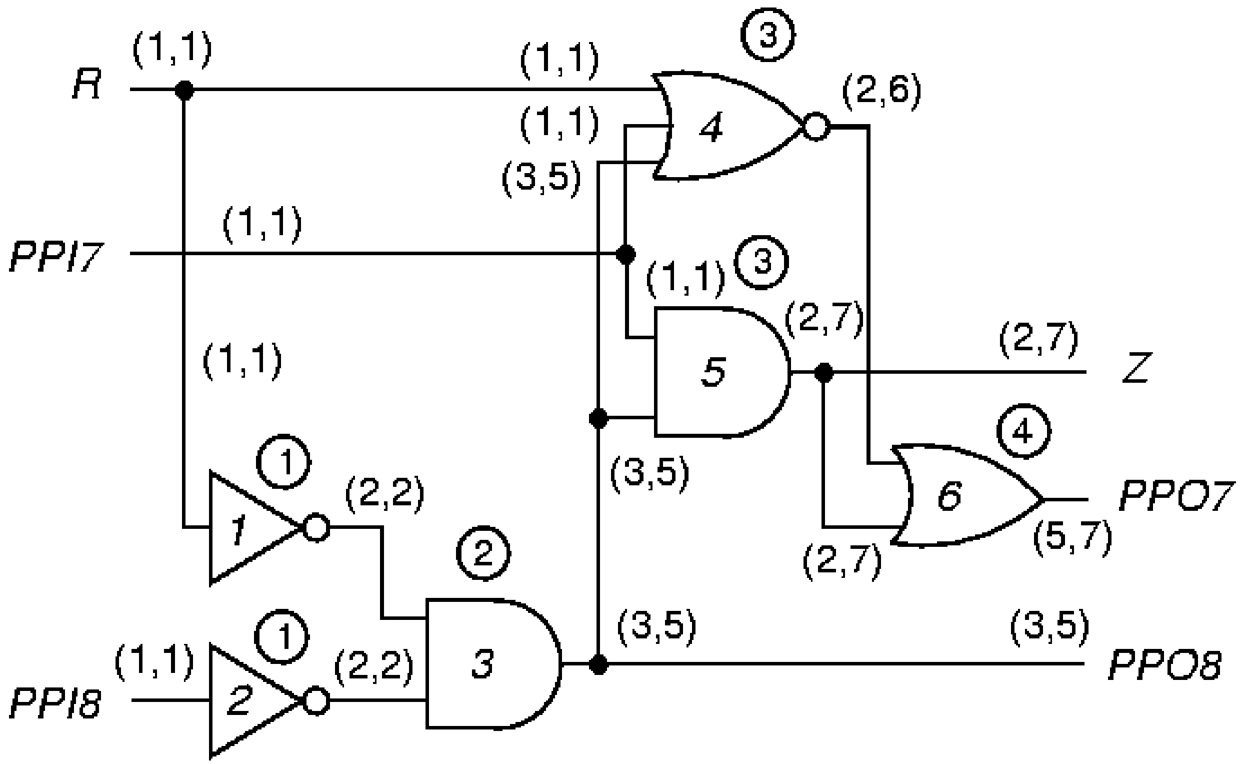


Format

Numbers in circles on top of each gate give level numbers
(CC0, CC1) next to each signal line

SCOAP Example

Final Controllabilities

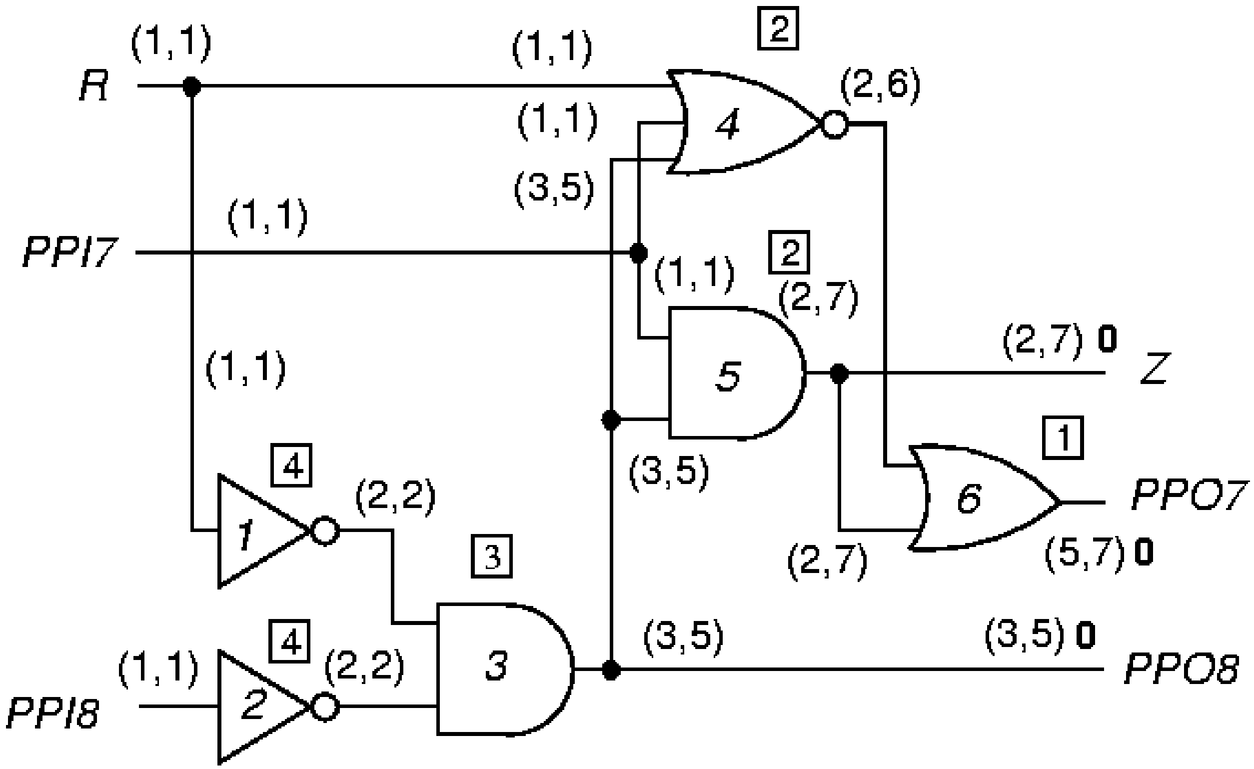


Format

Numbers in circles on top of each gate give level numbers
(CC0, CC1) next to each signal line

SCOAP Example

Observabilities for Level 1

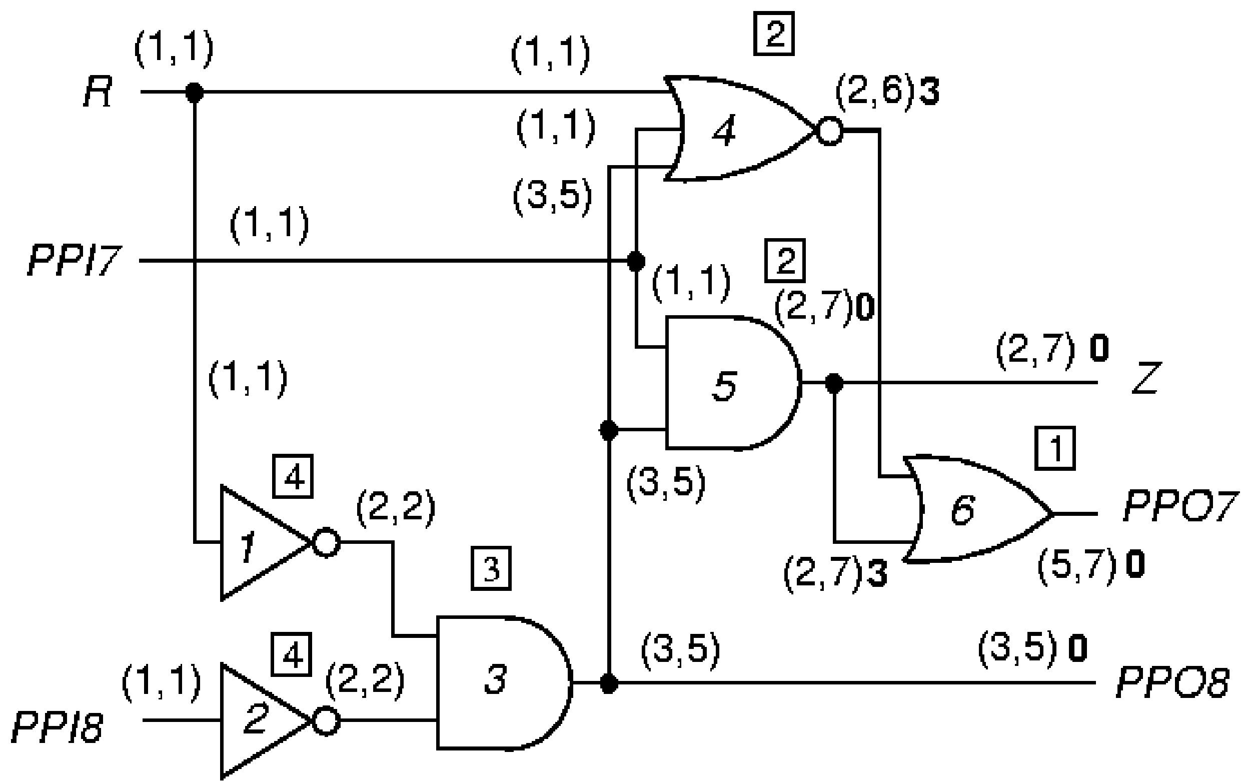


Format

Numbers in squares on top of each gate give level numbers from primary outputs
(CC0, CC1) **CO** next to each signal line

SCOAP Example

Observabilities for Level 2

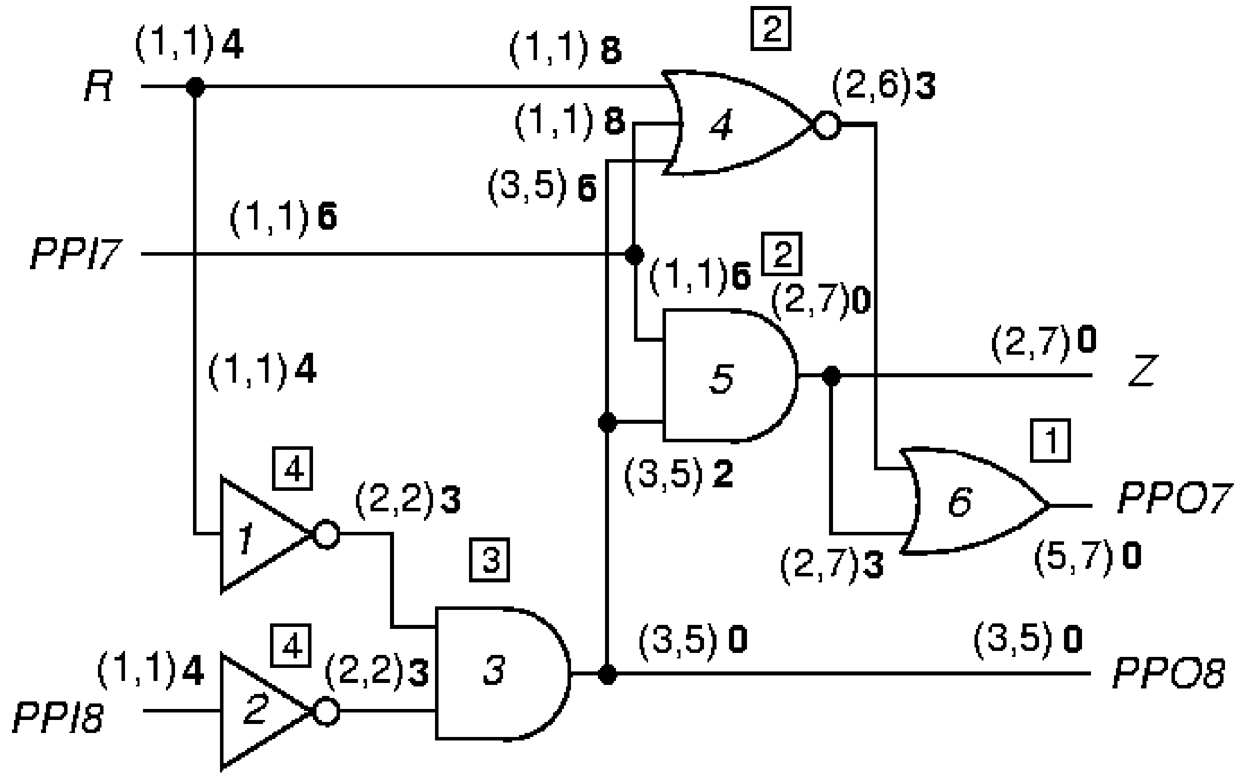


Format

Numbers in squares on top of each gate give level numbers from primary outputs
(CC0, CC1) **CO** next to each signal line

SCOAP Example

Final Observabilities



Format

Numbers in squares on top of each gate give level numbers from primary outputs
(CC0, CC1) **CO** next to each signal line

SCOAP Sequential Differences

Sequential SCOAP measures differences:

- Increment the sequential measure by 1 only when:
  - Signals propagate from FF inputs to Q or $\overline{Q}$, or
  - Signals propagate from FF outputs backwards to D, Clk, SET or RESET inputs.
- One must iterate on feedback loops until controllabilities stabilize.

SC0, SC1 and SO formulas differ from CC0, CC1 and C0 only in that you do NOT add one when moving from one level to another.

SC0 and SC1 roughly measure the number of times various FFs must be clocked to control a signal.

If line *l* can only be set to 1 but clocking FF *a* twice and FF *b* three times, SC1(*l*) should be 5.

SO correspondingly measure the number of times a FF must be clocked to observe a combinational signal.

Read section in text for details on D flip-flop SC0, SC1 and SO.