# Chapter 16: Design for Testability

Prof. Soo-Ik Chae

# Objectives

After completing this chapter, you will be able to:

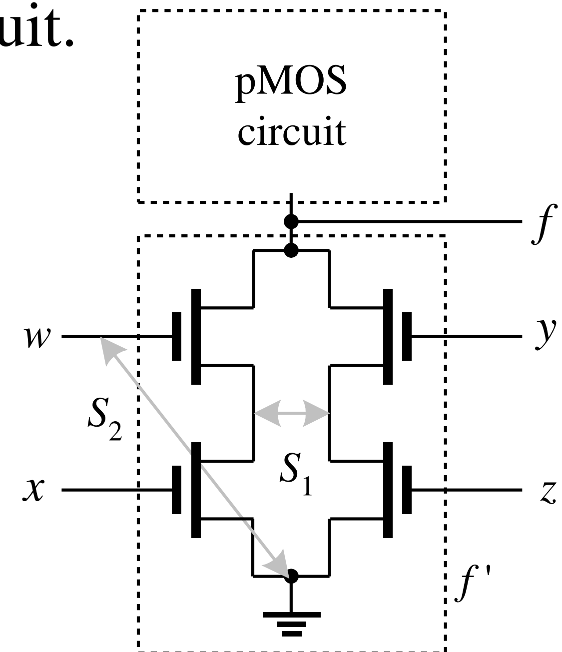
- Describe various fault models
- Understand the fundamentals of fault detection
- Understand the difficulties of sequential circuit tests
- Understand the basic principles of test vector generation
- Describe the basic principles of testable circuit design
- Understand the principle of boundary scan standard (IEEE1149.1)

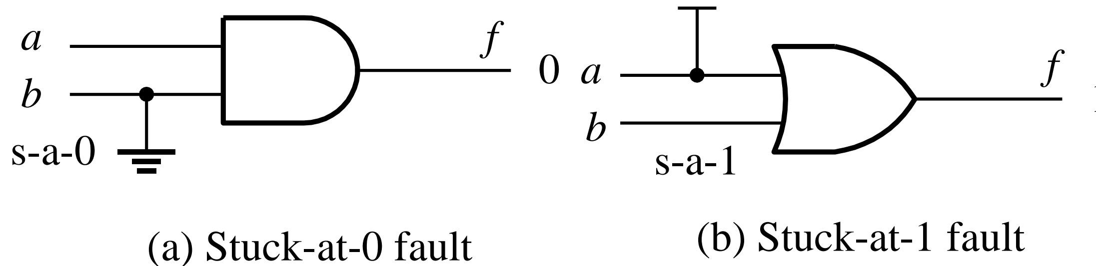
# Terminology Definition

- A defect means the unintended difference between the implemented hardware and its design.
  - Process defects
  - Material defects
  - Age defects
  - Package defects
- An error is the wrong output signal produced by a defective circuit.
- A fault is a representation of a defect at an abstracted function level.

# Fault Models --- Stuck-at/Bridge Faults

- The most common fault models used today:
  - A stuck-at-0 fault is modeled by assigning a fixed value of 0 to a signal line or net in the circuit.
  - A stuck-at-1 fault is modeled by assigning a fixed value of 1 to a signal line or net in the circuit.
  - A bridge fault is a shorting between any set of lines (nets).



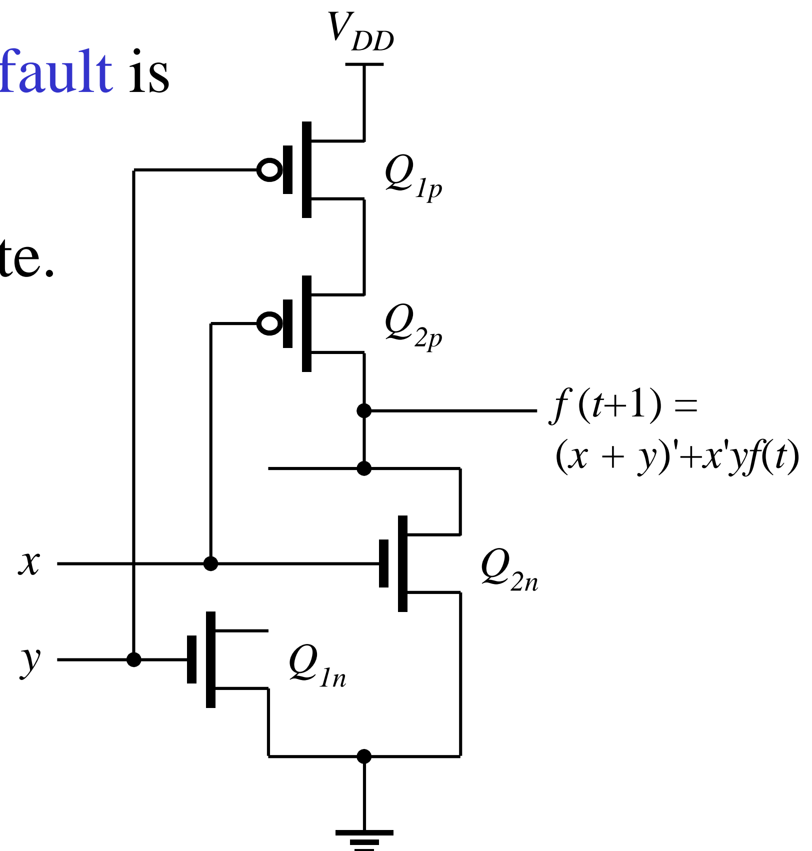




(a) Stuck-at-0 fault

(b) Stuck-at-1 fault

# Fault Models --- Stuck-Open/Closed Faults

- A stuck-open fault is modeled as a switch being permanently in open state.
- A stuck-closed (stuck-on) fault is modeled as a switch being permanently in shorted state.




- The effect of stuck-open fault is to produce a floating state at the output of the faulty logic circuit.
- The effect of stuck-closed fault is produce a power supply to ground conducting path.

# Fault Equivalence/Collapse/Coverage

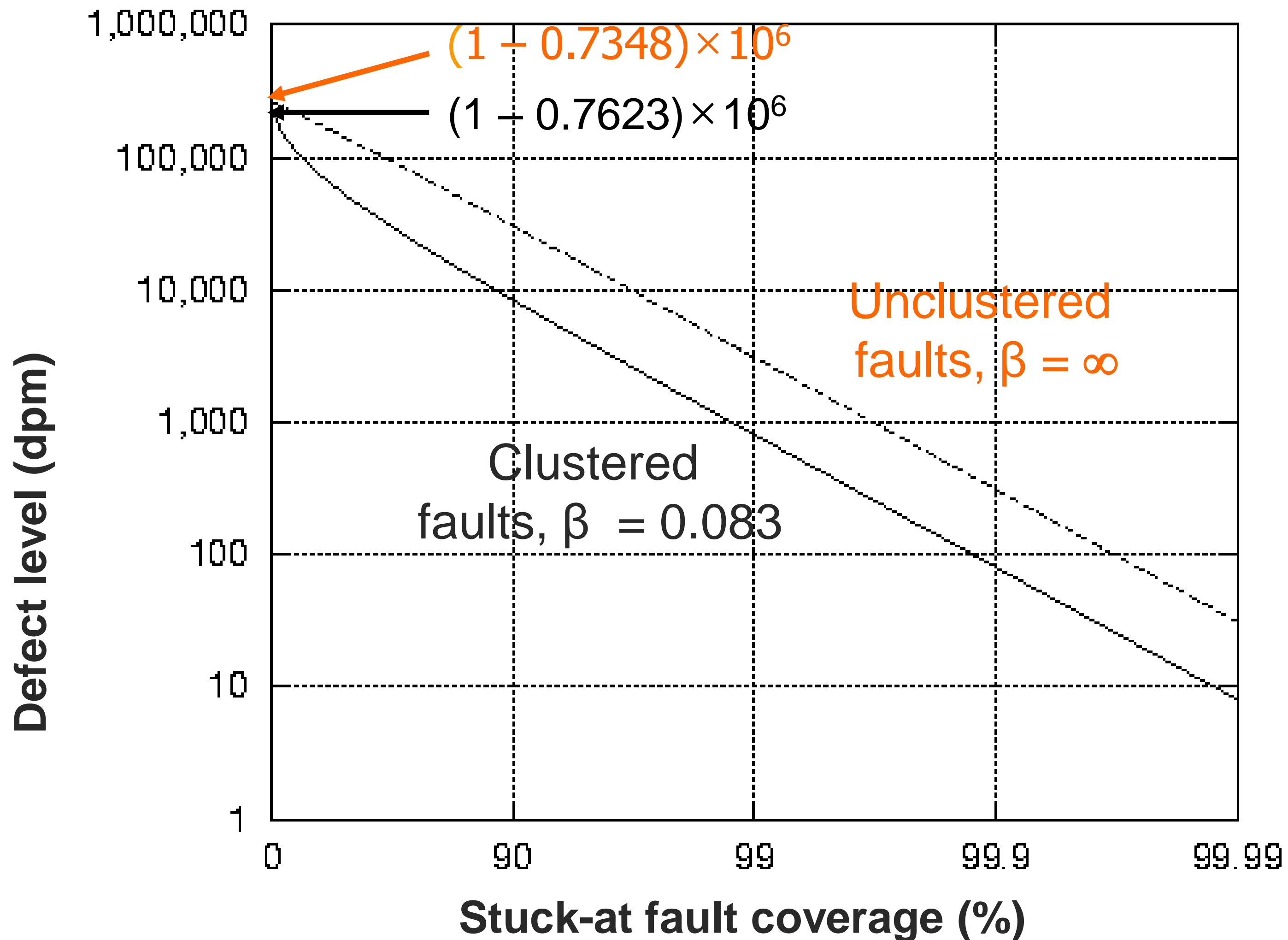
- Fault equivalence means two faults of a logic circuit transform the circuit such that two faulty circuits have the same output functions.
- Fault collapse means the process to select one fault from each equivalence set.

  Collapse ratio = |set of collapsed faults| / |set of faults|

- To achieve a world-class quality level, the fault coverage of a circuit has in excess of 98.5%.

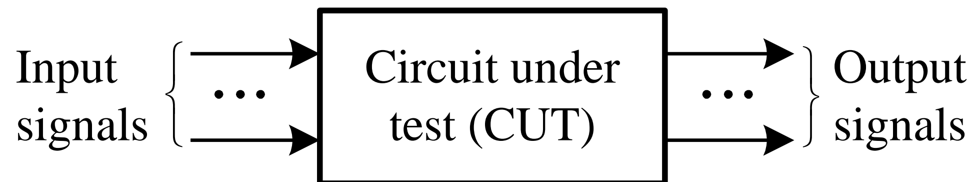
  Fault coverage = number of detected faults / total number of faults

# Defect level



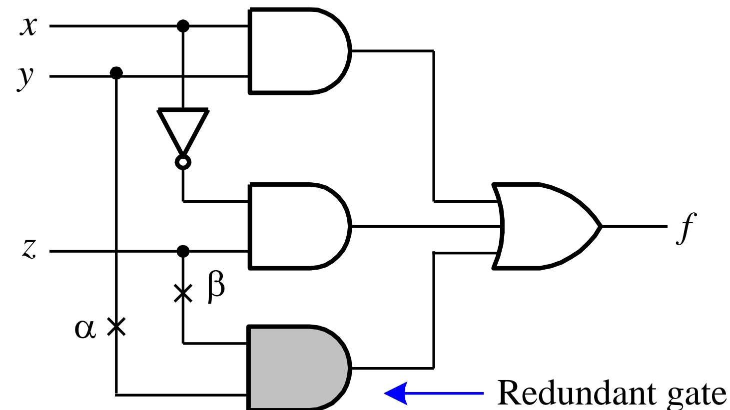
# Fault Detection --- A Basic Model of CUT

- The controllability of a particular node in a logic circuit is a measure of the ease of setting the node to a 1 or a 0 from the inputs.
- The observability of a particular node in a logic circuit is the degree to which you can observe the node at the outputs.

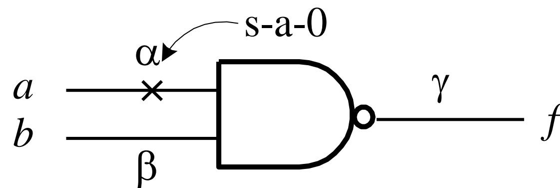


# Fault Detection --- Detectable and Undetectable Faults

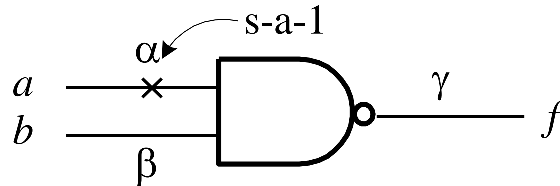
- A testable fault means that at least one input assignments can be found to make the outputs different between the fault-free and faulty circuits.
- An untestable fault means a fault for which no test can be found.
  - A redundant gate will cause some nets associated with it to be untestable.



# Fault Detection --- An Example of Complete Test Set




(a)




(b)

| $a$ | $b$ | $f_{fault_free}$ | $f_{faulty}$ | Detectable faults |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | γ/0 |
| 0 | 1 | 1 | 0 | α/1, γ/0 |
| 1 | 0 | 1 | 0 | β/1, γ/0 |
| 1 | 1 | 0 | 1 | α/0, β/0, γ/1 |

(c)

- With test vectors of {(0,1), (1,0), (1,1), all stuck-at faults can be tested.

# Difficulty of Sequential Circuit Test

- Verification of state table
  - Transfer the machine into a known state: a homing sequence
  - Test all transitions form that state
- Verify each transition by first taking the machine to a specific initial state, applying the input to perform the transition and then verifying the final state.

# Homing Sequence

- A homing sequence is an input sequence that produces an output response that uniquely indicates the state of a machine after the homing sequence has been applied.

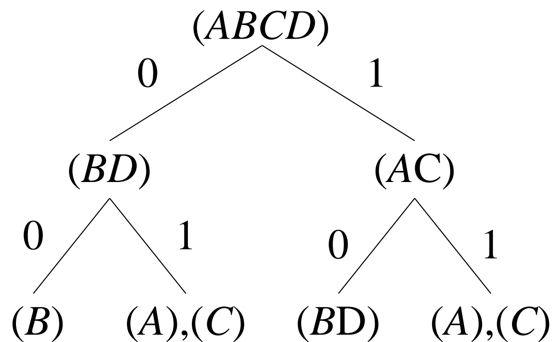
- A preset homing sequence is a homing sequence that does not employ the output response to determine subsequent inputs in the sequence.

# Example

| $PS$ | $NS, z$ ($x$ = 0) | $NS, z$ ($x$ = 1) |
|---|---|---|
| $A$ | $B$,0 | $A$,1 |
| $B$ | $B$,0 | $A$,0 |
| $C$ | $D$,0 | $C$,1 |
| $D$ | $B$,0 | $C$,1 |

# Example: Homing Sequence

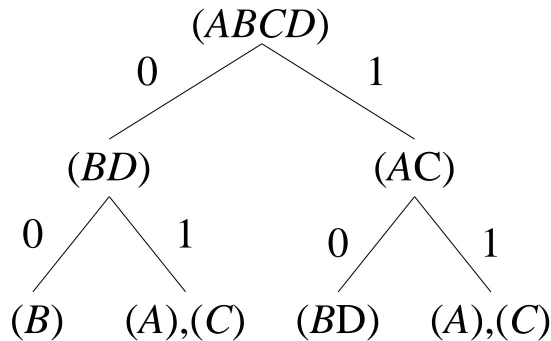
| *PS* | *NS, z* ($x$ = 0) | *NS, z* ($x$ = 1) |
|---|---|---|
| *A* | *B*,0 | *A*,1 |
| *B* | *B*,0 | *A*,0 |
| *C* | *D*,0 | *C*,1 |
| *D* | *B*,0 | *C*,1 |




| Initial state | Input sequence 00 | | 01 | | 11 | |
|---|---|---|---|---|---|---|
| *A* | 00 | *B* | 00 | *A* | 11 | *A* |
| *B* | 00 | *B* | 00 | *A* | 01 | *A* |
| *C* | 00 | *B* | 01 | *C* | 11 | *C* |
| *D* | 00 | *B* | 00 | *A* | 10 | *C* |

# Difficulty of Sequential Circuit Test

| PS | NS, z (x = 0) | NS, z (x = 1) |
|---|---|---|
| A | B,0 | A,1 |
| B | B,0 | A,0 |
| C | D,0 | C,1 |
| D | B,0 | C,1 |




| Initial state | Input sequence 00 | | Input sequence 01 | | Input sequence 11 | |
|---|---|---|---|---|---|---|
| A | 00 | B | 00 | A | 11 | A |
| B | 00 | B | 00 | A | 01 | A |
| C | 00 | B | 01 | C | 11 | C |
| D | 00 | B | 00 | A | 10 | C |

# Difficulty of Sequential Circuit Test

- A transfer sequence for state $S_i$ and $S_j$ of a sequential machine is the shortest input sequence that will take the machine from state $S_i$ to $S_j$.
- A circuit is strongly connected if and only if there exists for each ordered pair of states ($S_i$, $S_j$) of the circuit an input sequence that will transfer the circuit from state $S_i$ to $S_j$.
- So what are the difficulties of testing sequential circuits?

# Sequential Circuit Test

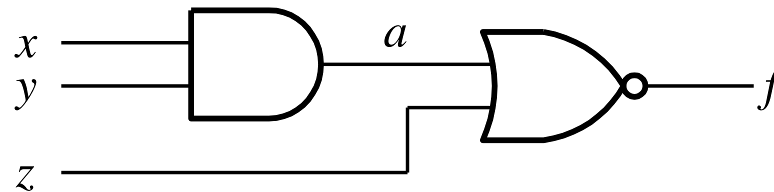
- For a strongly connected state diagram, a proper sequence can transfer the state diagram form one to another state.
- What if not strongly connected?
  - Difficult to test it
- Adding extra circuit that directly sets the state of the circuit.

# Test Vector Generation

- Four basic test vector generation approaches:
  - Fault tables
  - Fault simulation
  - Boolean differences
  - Path sensitization

# Fault Tables

- A fault table is a table that displays a set of faults and a set of test inputs.




| $x$ | $y$ | $z$ | $f$ | $f_{x0}$ | $f_{x1}$ | $f_{y0}$ | $f_{y1}$ | $f_{z0}$ | $f_{z1}$ | $f_{a0}$ | $f_{a1}$ | $f_{f0}$ | $f_{f1}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

Truth table — Single stuck-at-fault truth table

# Fault Tables

- A fault detection table is a table that identifies which input combinations detect a certain faults.
  - It is formed by performing the XOR of each output column in fault table with the output of fault-free circuit.

| $x$ | $y$ | $z$ | $f$ | $x_0$ | $x_1$ | $y_0$ | $y_1$ | $z_0$ | $z_1$ | $a_0$ | $a_1$ | $f_0$ | $f_1$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

May delete two rows

Equivalent faults

Equivalent faults

# Fault Tables

- A reduced fault table is a fault table that combines equivalent faults and deletes the redundant rows in the fault detection table.

| $x$ | $y$ | $z$ | $f$ | $\sqrt{g}$ | $\sqrt{x_1}$ | $\sqrt{y_1}$ | $\sqrt{z_0}$ | $\sqrt{h}$ | $\sqrt{f_1}$ |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | (1) | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | (1) | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | (1) | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | (1) | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

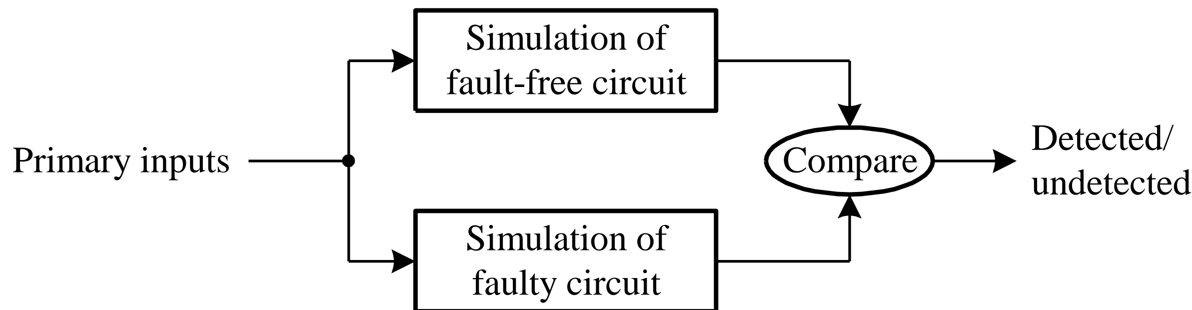
$g = \{x_0, y_0, a_0\}$

$h = \{z_1, a_1, f_0\}$

- The test vectors are: {(0,0,1), (0,1,0), (1,0,0), (1,1,0)}.

# Fault Simulation

- The basic concept of fault simulation approach to test pattern generation is to select a primary input combination and determine its fault detection capabilities by simulation. The process is repeated until:
  - All faults are covered,
  - At least an acceptable number of faults are covered, or
  - Some predefined stopping point is reached.

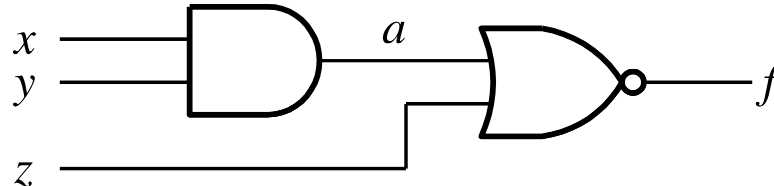


# Fault Simulation

- Types of fault simulation:
  - The row method
  - The column method

# Fault Simulation

- The row method




| $x$ | $y$ | $z$ | $f$ | $f_{x0}$ | $f_{x1}$ | $f_{y0}$ | $f_{y1}$ | $f_{z0}$ | $f_{z1}$ | $f_{a0}$ | $f_{a1}$ | $f_{f0}$ | $f_{f1}$ | Faults detected |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | $z1$, $a1$, $f0$ |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | * | 0 | * | * | 1 | $z0$, $f1$ |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | * | * | 1 | * | * | * | $x1$ |
| 0 | 1 | 1 | 0 | 0 | * | 0 | 0 | * | * | 0 | * | * | * | |
| 1 | 0 | 0 | 1 | 1 | * | 1 | 0 | * | * | 1 | * | * | * | $y1$ |
| 1 | 0 | 1 | 0 | 0 | * | 0 | * | * | * | 0 | * | * | * | |
| 1 | 1 | 0 | 0 | 1 | * | 1 | * | * | * | 1 | * | * | * | $x0$, $y0$, $a0$ |
| 1 | 1 | 1 | 0 | * | * | * | * | * | * | * | * | * | * | |

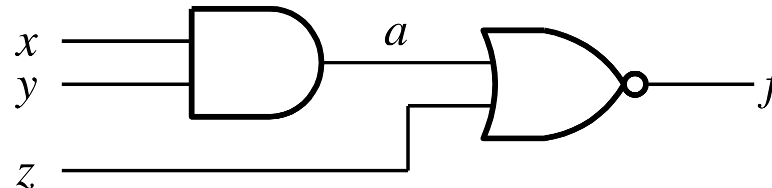
Truth table

* --- not simulated for this input pattern.

- Test vectors = {(0,0,0), (0,0,1), (0,1,0), (1,0,0), (1,1,0)}.

# Fault Simulation

- The column method




| $x$ | $y$ | $z$ | $f$ | $f_{x0}$ | $f_{x1}$ | $f_{y0}$ | $f_{y1}$ | $f_{z0}$ | $f_{z1}$ | $f_{a0}$ | $f_{a1}$ | $f_{f0}$ | $f_{f1}$ | Faults detected |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 | * | 1 | * | 0 | 1 | 0 | 0 | 1 | $z1$, $a1$, $f0$ |
| 0 | 0 | 1 | 0 | 0 | 0 | * | 0 | * | * | 0 | * | * | 1 | $f1$ |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | * | * | 1 | * | * | * | $x1$ |
| 0 | 1 | 1 | 0 | 0 | * | 0 | 0 | * | * | 0 | * | * | * | |
| 1 | 0 | 0 | 1 | 1 | * | 1 | 0 | 1 | * | 1 | * | * | * | $y1$ |
| 1 | 0 | 1 | 0 | 0 | * | 0 | * | 1 | 0 | 0 | * | * | * | $z0$ |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | * | 0 | 1 | 0 | * | * | $x0$, $y0$, $a0$ |
| 1 | 1 | 1 | 0 | * | 0 | * | 0 | * | 0 | * | 0 | * | * | |

Truth table

* --- not simulated for this input pattern.

- Test vectors = {(0,0,0), (0,0,1), (0,1,0), (1,0,0), (1,0,1), (1,1,0)}.

# Boolean Differences

- Boolean difference of a switching function $f(x_{n-1}, \ldots, x_1, x_0)$ is defined as:

$$\frac{df(X)}{dx} = f(x_{n-1},\ldots,x_{i+1},0,x_{i-1},\ldots,x_0) \oplus f(x_{n-1},\ldots,x_{i+1},1,x_{i-1},\ldots,x_0)$$
$$= f_i(0) \oplus f_i(1)$$

  - Boolean difference sometimes is called Boolean partial derivative.
  - Boolean difference is not an effective way to compute test patterns for large circuits.

# Boolean Differences

- Example: $f(X) = x_1 x_2 + x_3$

$$\frac{df(X)}{dx_1} = (1 \cdot x_2 + x_3) \oplus (0 \cdot x_2 + x_3) = (x_2 + x_3) \oplus x_3$$
$$= (x_2 + x_3)' x_3 + (x_2 + x_3) x'_3 = x_2 x'_3$$

- To test stuck-at-0 fault at net xi, it is required to compute:

$$x_i \frac{df(X)}{dx_i} = 1$$

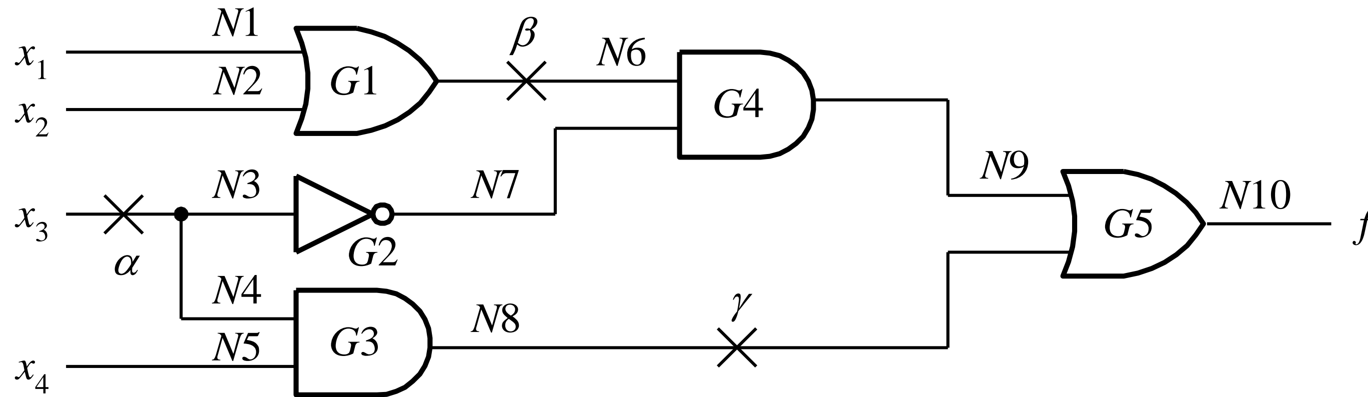
$x_i$=1일 때 output이 다른 경우가 test vector

- To test stuck-at-1 fault at net xi, it is required to compute:

$$x'_i \frac{df(X)}{dx_i} = 1$$

$x_i$=0일 때 output이 다른 경우가 test vector

# Boolean Differences




$$f(X) = (x_1 + x_2)x'_3 + x_3x_4$$

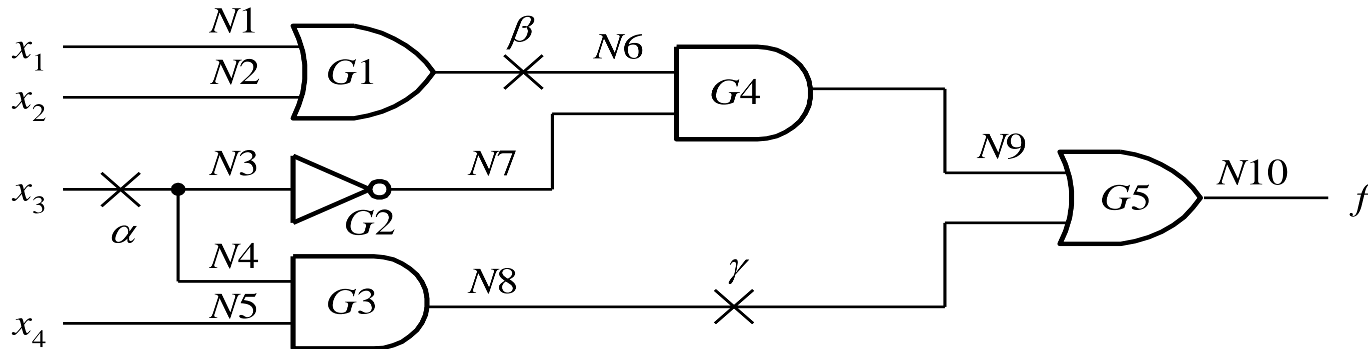
- Stuck-at-0 fault at net α is

$$x_3 \frac{df(X)}{dx_3} = 1 = x_3[f_3(0) \oplus f_3(1)]$$

$$= x'_1\, x'_2\, x_3x_4 + x_1x_3x'_4 + x_2x_3x'_4$$

- The test vector set is {(0, 0, 1, 1), (1, $\phi$, 1, 0), ($\phi$, 1, 1, 0)}.

# Boolean Differences




- Stuck-at-0 fault at net $\beta$ is

$$y = x_1 + x_2$$

$$f(X, y) = yx'_3 + x_3x_4$$

$$y \cdot \frac{df(X, y)}{dy} = y[x_3x_4 \oplus (x'_3 + x_3x_4)] = yx'_3$$

$$= (x_1 + x_2)x'_3 = x_1x'_3 + x_2x'_3$$

- The test vector set is {(1, $\phi$, 0, $\phi$), ($\phi$, 1, 0, $\phi$)}.

# Path Sensitization

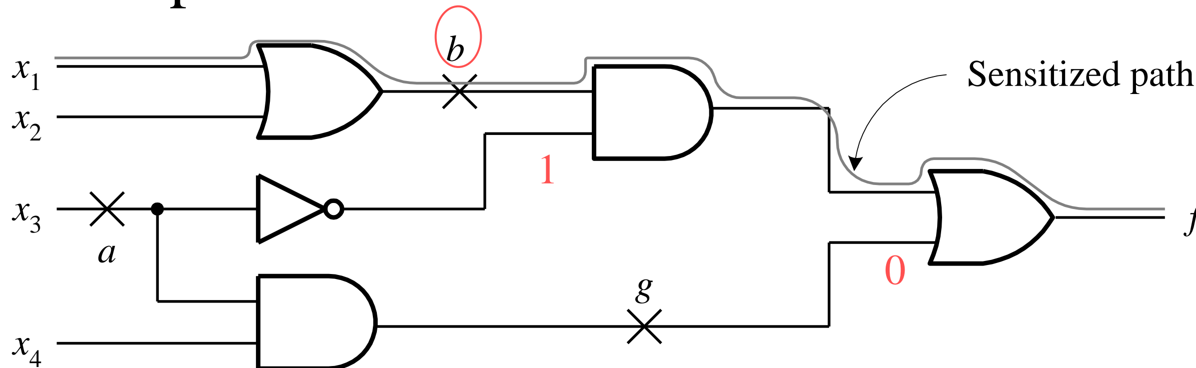
- Basic operations of path sensitization method:
  - Fault sensitization: A stuck-at fault is activated by forcing the signal driving it to an opposite value from the fault value.
  - Fault propagation: The fault effect is propagated through one or more paths to a primary output of the circuit.
  - Line justification (Consistency operation): The internal signal assignments previously made to sensitize a fault or propagate its effect are justified by setting primary inputs of the circuit.

# Path Sensitization

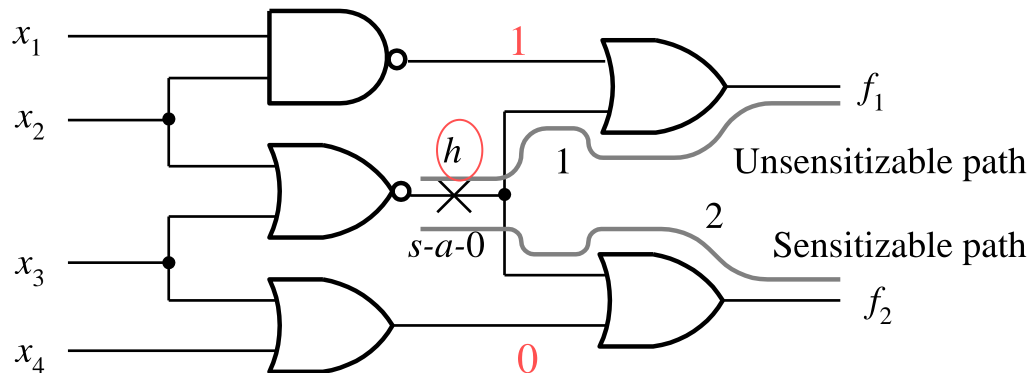
- Sensitized path is a path that can propagate the net value of the primary inputs with a consistency operation for a stuck-at-fault
- Otherwise, unsensitized path

# Path Sensitization

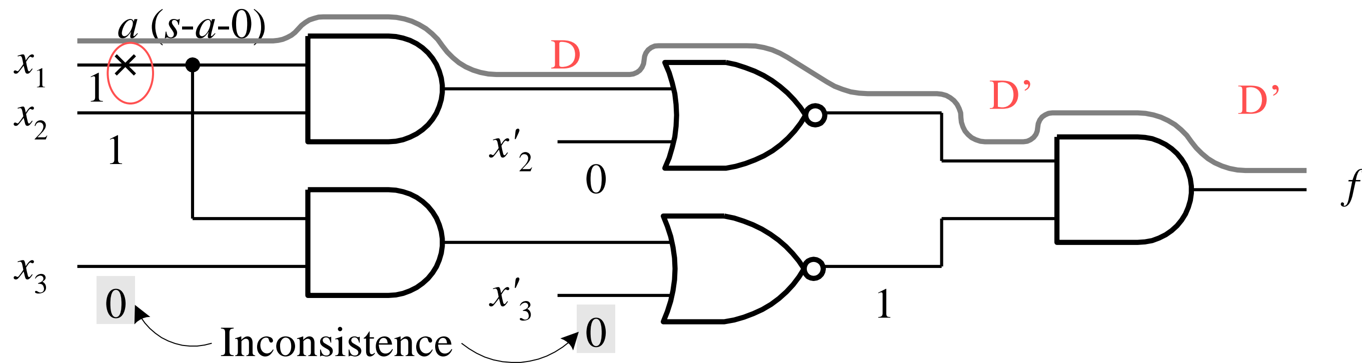
- Sensitized path



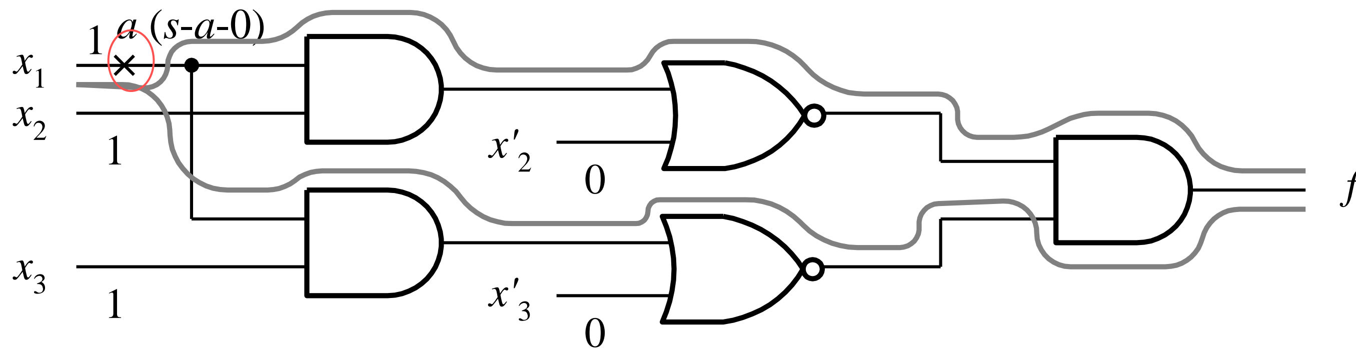

- Unsensitized path



# Path Sensitization




(a) Single sensitized path




(b) Multiple sensitized paths

# A Simplified *D* Algorithm

- pdcf (primitive *D*-cube of fault) of the five most commonly used gates.

| $x$ | $y$ | $z$ | Faults covered |
|---|---|---|---|
| 0 | 0 | $D'$ | $z/1$ |
| 0 | 1 | $D'$ | $x/1$, $z/1$ |
| 1 | 0 | $D'$ | $y/1$, $z/1$ |
| 1 | 1 | $D$ | $x/0$, $y/0$, $z/0$ |

(a) AND gate

| $x$ | $y$ | $z$ | Faults covered |
|---|---|---|---|
| 0 | 0 | $D'$ | $x/1$, $y/1$, $z/1$ |
| 0 | 1 | $D$ | $x/0$, $z/0$ |
| 1 | 0 | $D$ | $y/0$, $z/0$ |
| 1 | 1 | $D$ | $z/0$ |

(b) OR gate

| $x$ | $y$ | $z$ | Faults covered |
|---|---|---|---|
| 0 | 0 | $D$ | $z/0$ |
| 0 | 1 | $D$ | $x/1$, $z/0$ |
| 1 | 0 | $D$ | $y/1$, $z/0$ |
| 1 | 1 | $D'$ | $x/0$, $y/0$, $z/1$ |

(c) NAND gate

| $x$ | $y$ | $z$ | Faults covered |
|---|---|---|---|
| 0 | 0 | $D$ | $x/1$, $y/1$, $z/0$ |
| 0 | 1 | $D'$ | $x/0$, $z/1$ |
| 1 | 0 | $D'$ | $y/0$, $z/1$ |
| 1 | 1 | $D'$ | $z/1$ |

(d) NOR gate

| | $z$ | Faults covered |
|---|---|---|
| 0 | $D$ | $x/1$, $z/0$ |
| 1 | $D'$ | $x/0$, $z/1$ |

(e) NOT gate

# A Simplified $D$ Algorithm

- pdc (propagation of D-cube): error propagation

| AND gate | | | OR gate | | | NAND gate | | | NOR gate | | | NOT gate | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $x$ | $y$ | $z$ | $x$ | $y$ | $z$ | $x$ | $y$ | $z$ | $x$ | $y$ | $z$ | $x$ | $z$ |
| $D'$ | 1 | $D'$ | $D'$ | 0 | $D'$ | $D'$ | 1 | $D$ | $D$ | 0 | $D'$ | $D'$ | $D$ |
| 1 | $D'$ | $D'$ | 0 | $D'$ | $D'$ | 1 | $D'$ | $D$ | 0 | $D$ | $D'$ | | |
| $D$ | 1 | $D$ | $D$ | 0 | $D$ | $D$ | 1 | $D'$ | $D$ | 0 | $D'$ | $D$ | $D'$ |
| 1 | $D$ | $D$ | 0 | $D$ | $D$ | 1 | $D$ | $D'$ | 0 | $D$ | $D'$ | | |

- sc (singular cover): fixed output

| AND gate | | | OR gate | | | NAND gate | | | NOR gate | | | NOT gate | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $x$ | $y$ | $z$ | $x$ | $y$ | $z$ | $x$ | $y$ | $z$ | $x$ | $y$ | $z$ | $x$ | $z$ |
| 0 | $\phi$ | 0 | 1 | $\phi$ | 1 | 0 | $\phi$ | 1 | 1 | $\phi$ | 0 | 1 | 0 |
| $\phi$ | 0 | 0 | $\phi$ | 1 | 1 | $\phi$ | 0 | 1 | $\phi$ | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | | |

# A Simplified *D* Algorithm

- A simplified *D* algorithm
  1. Fault sensitization: Select a pdcf for the fault for which a test pattern is to be generated.
  2. *D*-drive: Use pdcs to propagate the *D* signal to at least one primary outputs of the circuit.
  3. Consistency operations: Use the sc of each logic module to perform the consistency operation.

# A Simplified *D* Algorithm



| | N1 | N2 | N3 | N4 | N5 | N6 | N7 | N8 | N9 | N10 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| pdcf of G4 → | | | | | | D | 1 | | *D* | | Fault sensitization |
| pdc of G5 → | | | | | | | | 0 | *D* | *D* | *D*-drive |
| sc of G1 | 1 | $\phi$ | | | | 1 | | | | | Consistency operations |
| sc of G1 | $\phi$ | 1 | | | | | | | | | Consistency operations |
| sc of G2 | | | 0 | | | | 1 | | | | Consistency operations |
| sc of G3 ← | | | | 0 | $\phi$ | | | 0 | | | Consistency operations |

# Testable Circuit Design

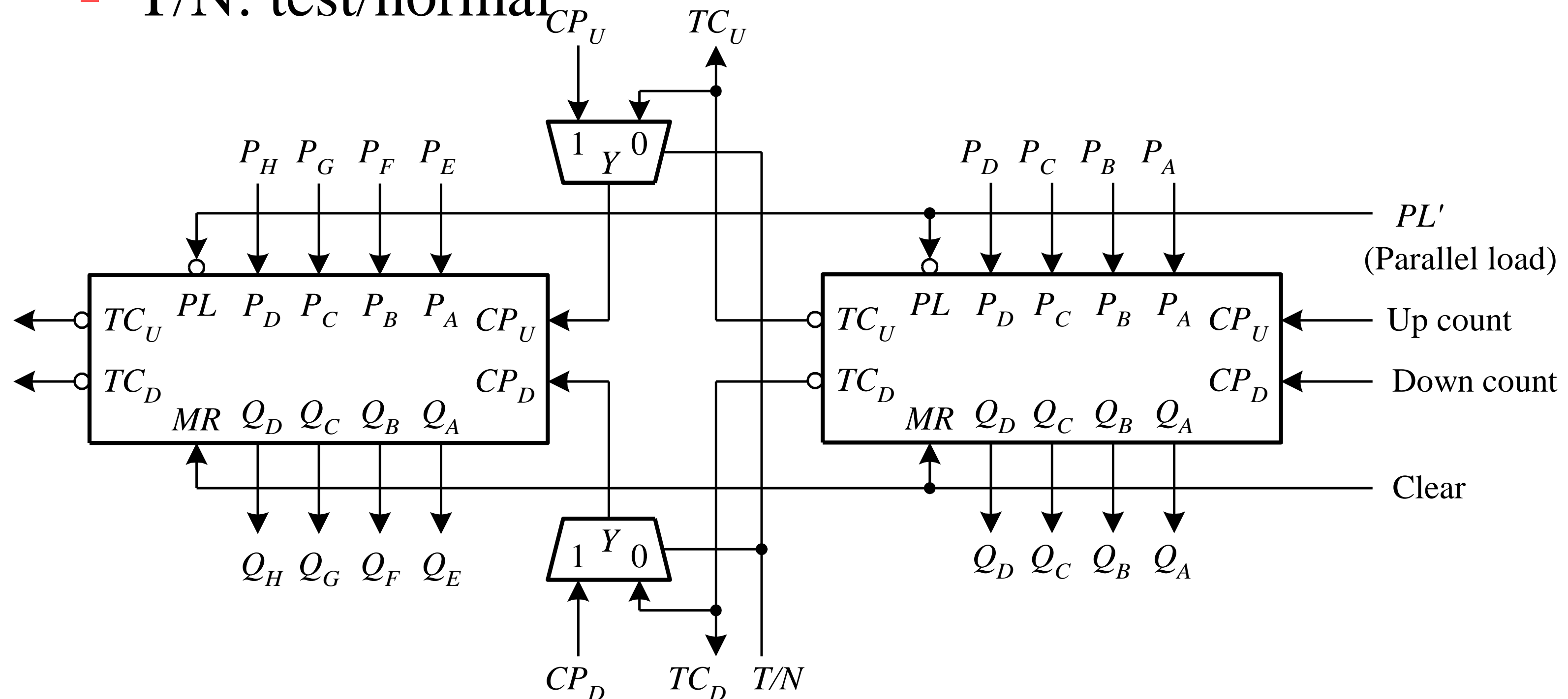
- Testable circuit design is also called design for testability (DFT).
- Ad hoc testing
  - uses some means to increase both observability and controllability of the circuits.
- Scan-path methods
  - provide the observability and controllability at each register.
- Built-in self-test (BIST)
  - relies on augmenting circuits to allow them to perform operations so as to prove the correct operation of the circuit.

# Ad hoc Testing

- Basic principles are to increase both the observability and controllability of circuits.
  - Providing more control and test points
  - Using multiplexers to increase the number of internal control and test points
  - Breaking feedback paths
  - Using state registers to reduce the additional of I/O pins required for testing signals.

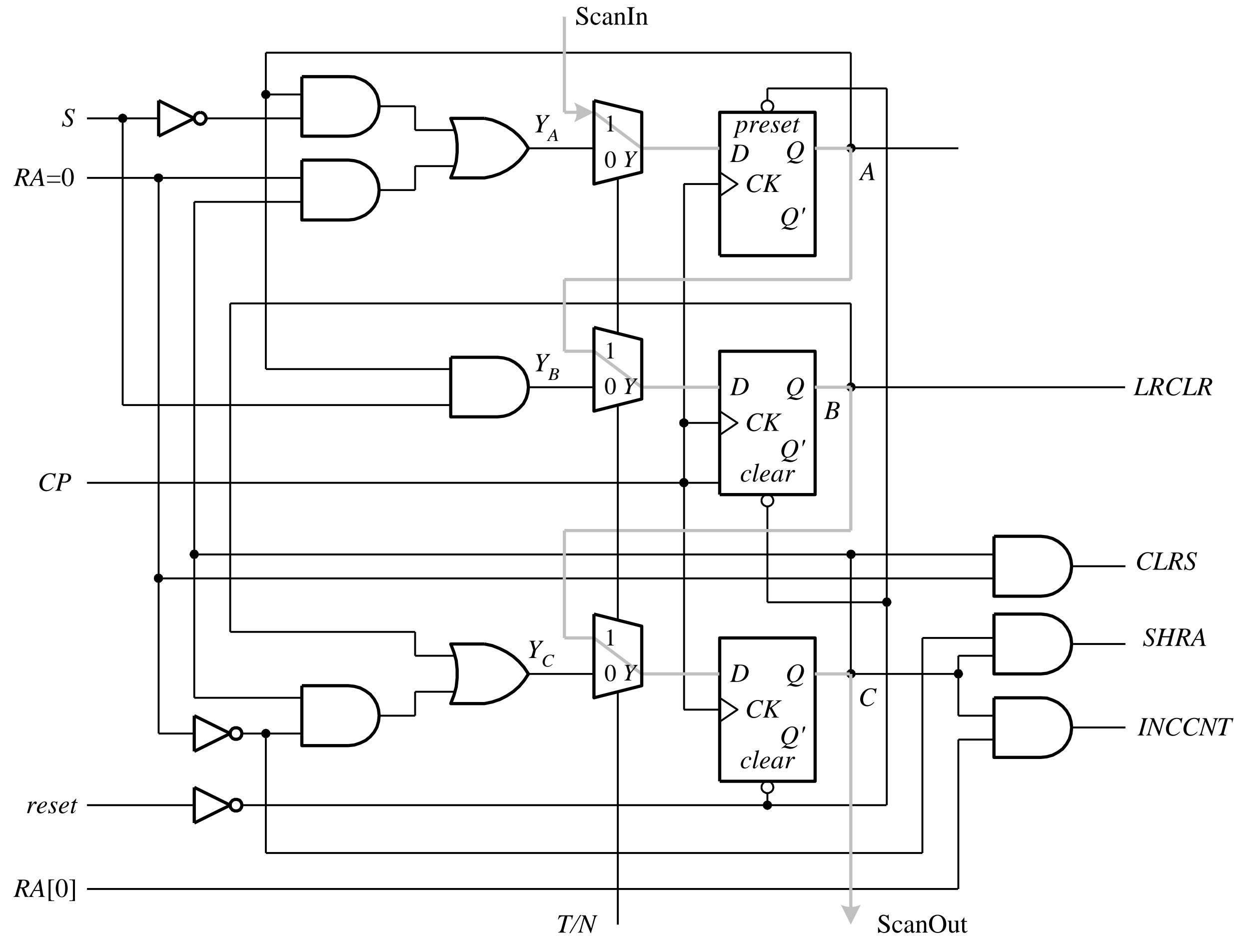
# Ad hoc Testing

- An example of exhaustive test: 8-but up/down counter with two 4-bit up/down counters
  - 256 cycles vs 16 cycles
  - T/N: test/normal

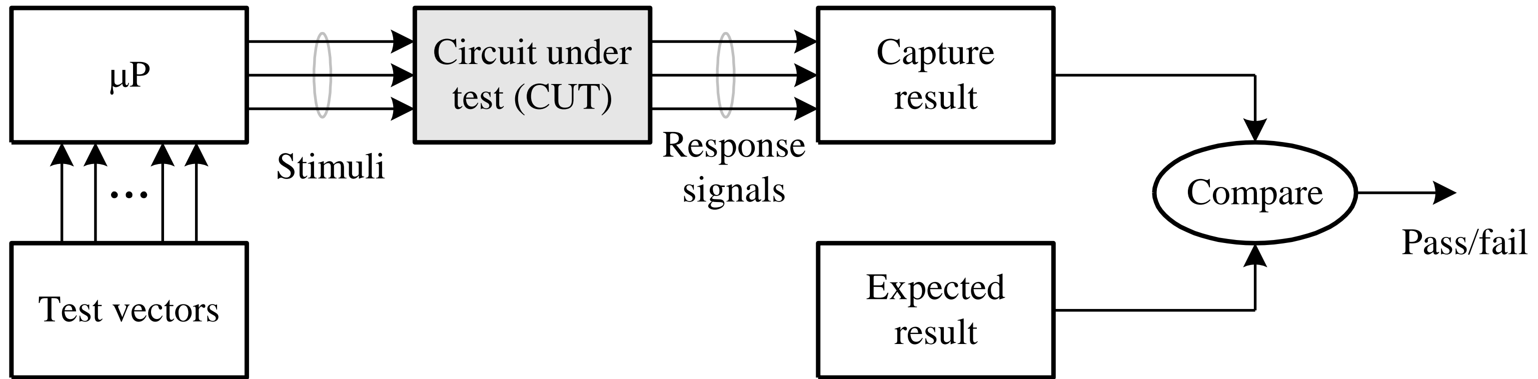


# Scan-Path Methods

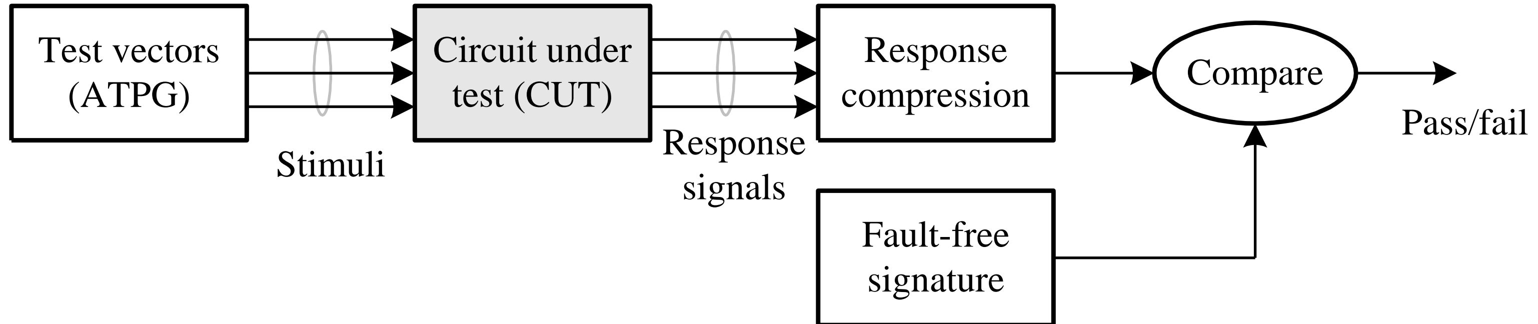
- An example of scan-path method



# BIST Principles



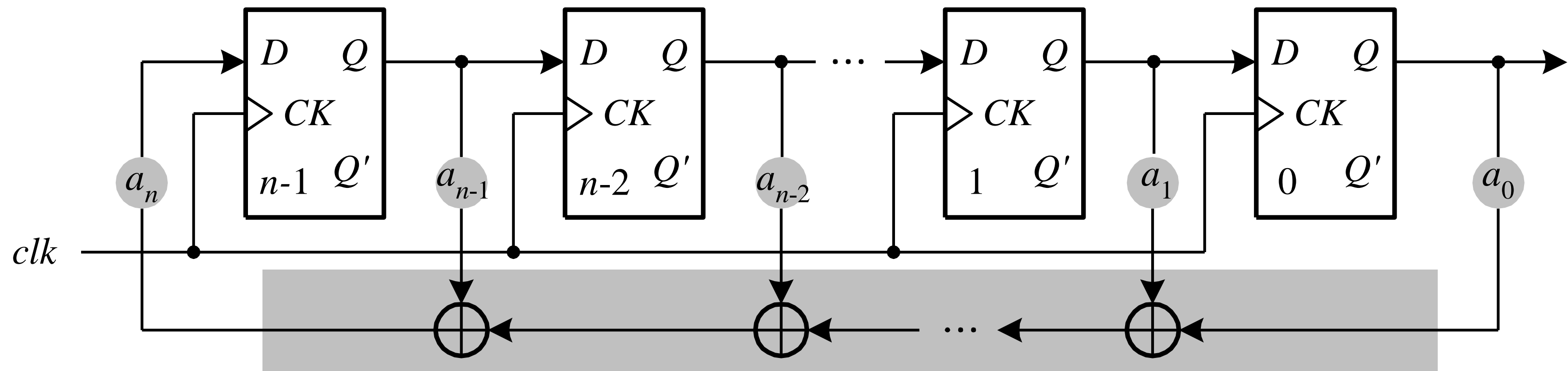

(a) ATE (Automatic test equipment)



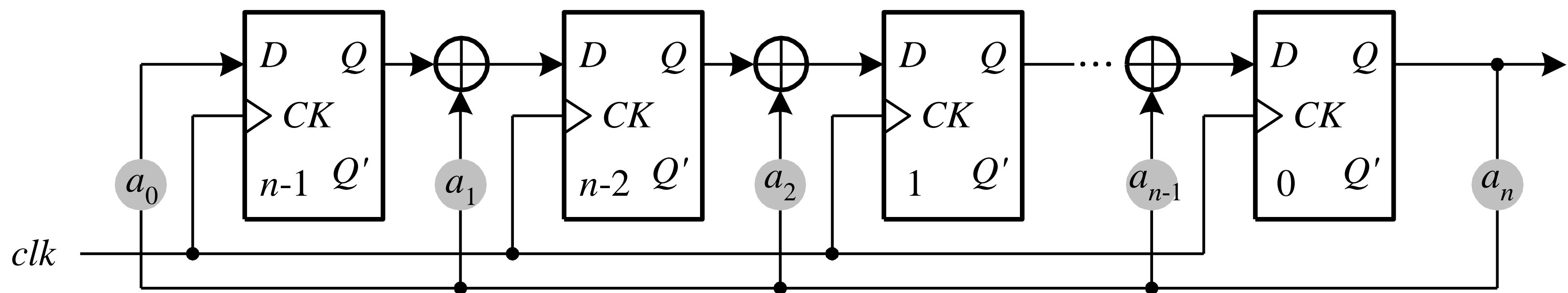

(b) Built-in self test (BIST)

# Automatic Test Pattern Generation (ATPG)(ALFSR)

- PR-sequence generator




(a) Standard format




(b) Modular format

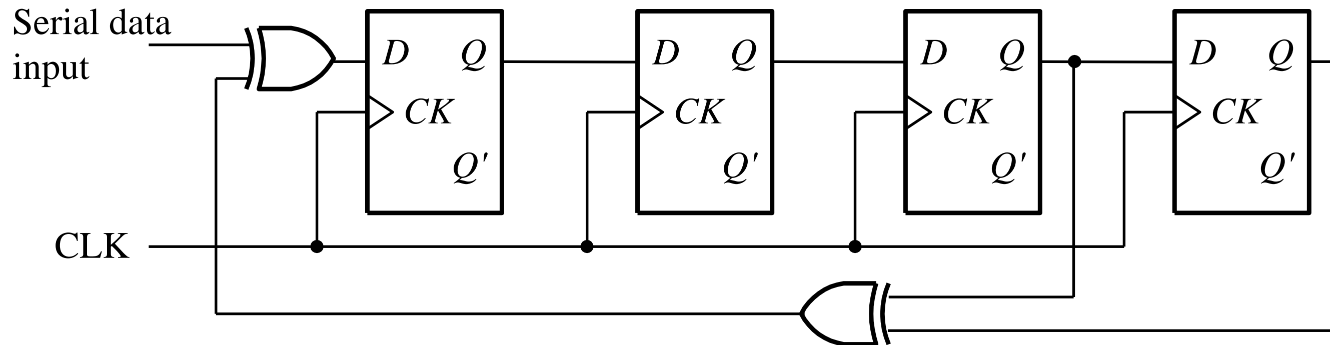
# Automous linear feedback shift register (ALFSR)

- A sample primitive polynomials for n from 1 to 60.

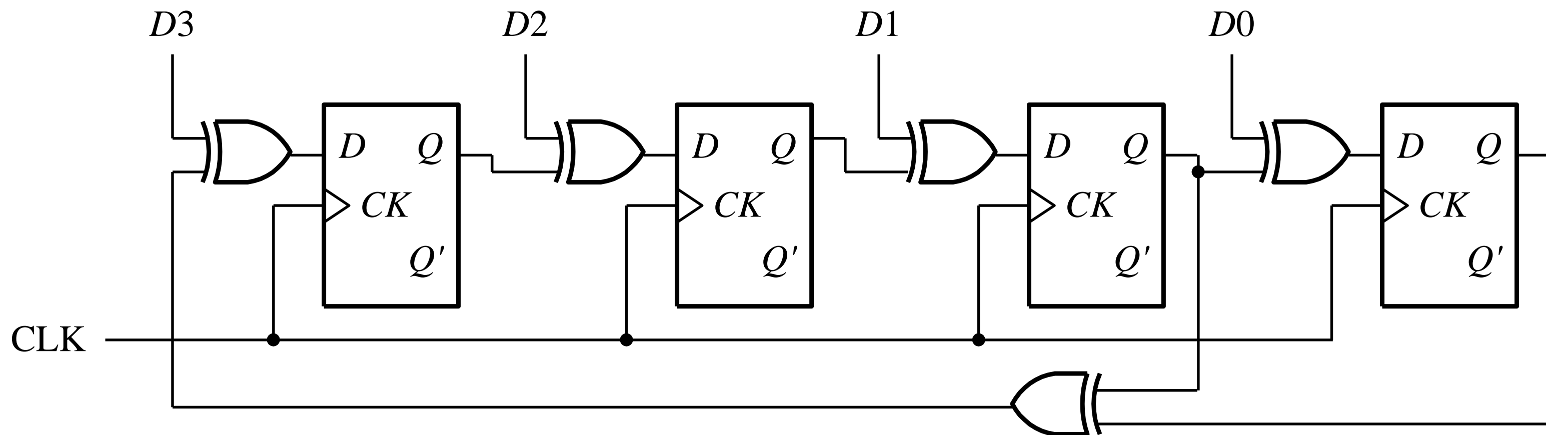
| $n$ | $f(x)$ | $n$ | $f(x)$ | $n$ | $f(x)$ |
|---|---|---|---|---|---|
| 1, 2, 3, 4, 6, 7, 15, 22, 60 | $1+x+x^n$ | 24 | $1+x+x^2+x^7+x^n$ | 43 | $1+x+x^5+x^6+x^n$ |
| | | 26 | $1+x+x^2+x^6+x^n$ | 44, 50 | $1+x+x^{26}+x^{27}+x^n$ |
| 5, 11, 21, 29 | $1+x^2+x^n$ | 30 | $1+x+x^2+x^{23}+x^n$ | 45 | $1+x+x^3+x^4+x^n$ |
| 10, 17, 20, 25, 28, 31, 41, 52 | $1+x^3+x^n$ | 32 | $1+x+x^2+x^{22}+x^n$ | 46 | $1+x+x^{20}+x^{21}+x^n$ |
| | | 33 | $1+x^{13}+x^n$ | 48 | $1+x+x^{27}+x^{28}+x^n$ |
| 9 | $1+x^4+x^n$ | 34 | $1+x+x^{14}+x^{15}+x^n$ | 49 | $1+x^9+x^n$ |
| 23, 47 | $1+x^5+x^n$ | 35 | $1+x^2+x^n$ | 51, 53 | $1+x+x^{15}+x^{16}+x^n$ |
| 18 | $1+x^7+x^n$ | 36 | $1+x^{11}+x^n$ | 54 | $1+x+x^{36}+x^{37}+x^n$ |
| 8 | $1+x^2+x^3+x^4+x^n$ | 37 | $1+x^2+x^{10}+x^{12}+x^n$ | 55 | $1+x^{24}+x^n$ |
| 12 | $1+x+x^4+x^6+x^n$ | 38 | $1+x+x^5+x^6+x^n$ | 56,59 | $1+x+x^{21}+x^{22}+x^n$ |
| 13 | $1+x+x^3+x^4+x^n$ | 39 | $1+x^4+x^n$ | 57 | $1+x^7+x^n$ |
| 14, 16 | $1+x^3+x^4+x^5+x^n$ | 40 | $1+x^2+x^{19}+x^{21}+x^n$ | 58 | $1+x^{19}+x^n$ |
| 19, 27 | $1+x+x^2+x^5+x^n$ | 42 | $1+x+x^{22}+x^{23}+x^n$ | | |

# Signature Generators

Serial input signature register
Multiple input signature register

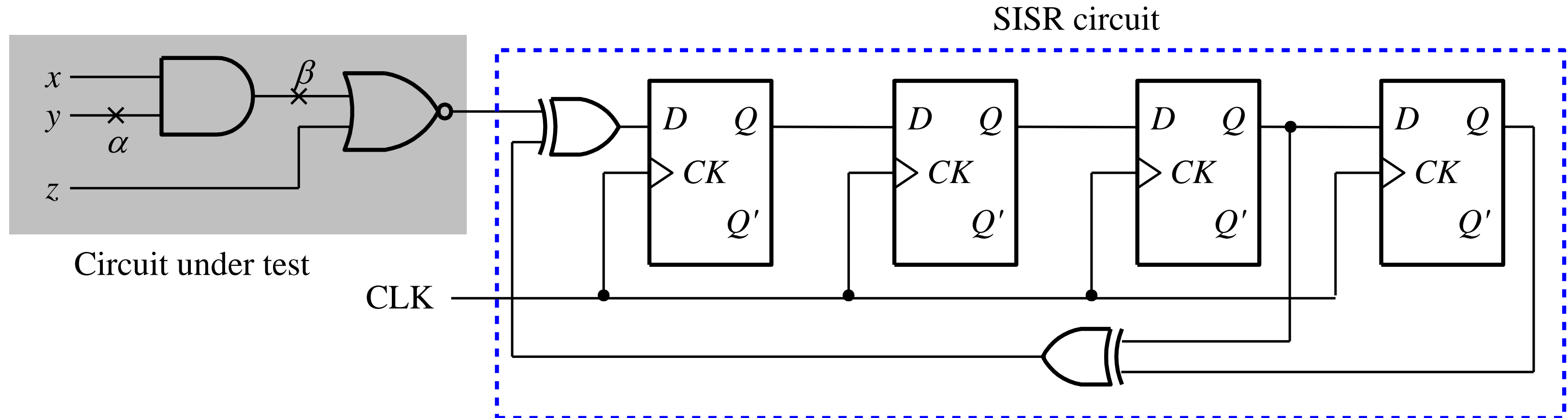



(a) SISR Circuit example




(b) MISR circuit example

# A Signature Application Example




(a) Logic diagram

| Input $x$ | $y$ | $z$ | Fault-free output $f$ | Faulty output $f_{\alpha/0}$ and $f_{\beta/0}$ | $f_{\alpha/1}$ | $f_{\beta/1}$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| | | | 0101 | 0001 | 1100 | 0000 |

(b) A numerical example of signature

# Signature Capability

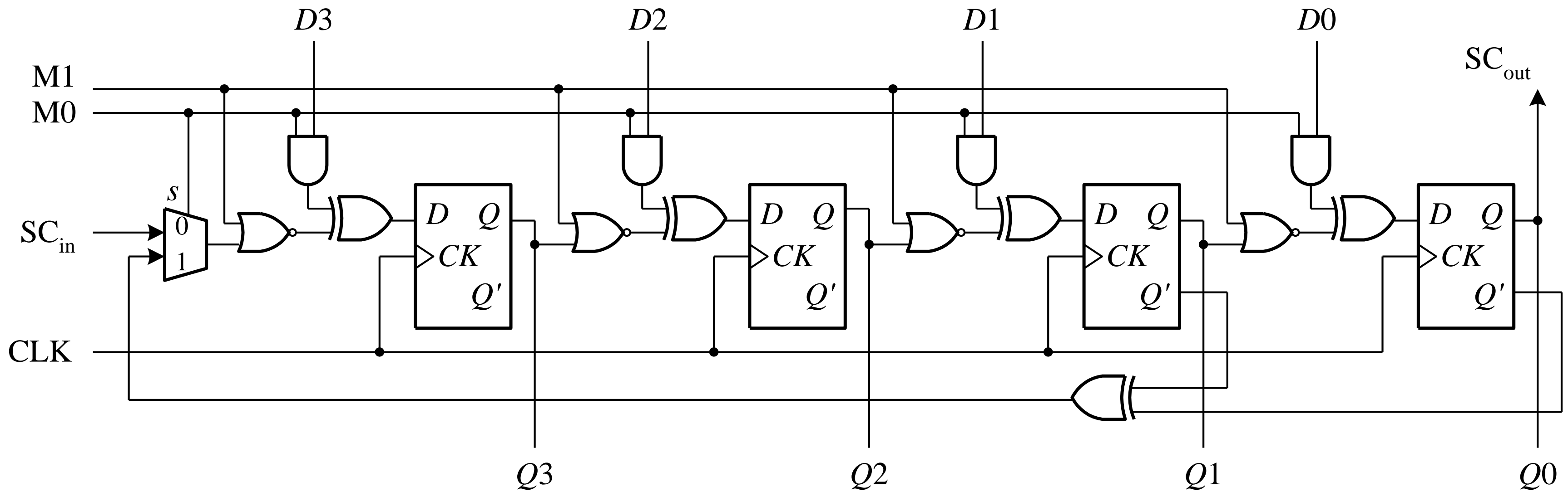
- For a particular fault-response, there are $2^{m-n} - 1$ erroneous bit streams that will produce the same signature.

- Theorem: For an input data stream of length $m$, if all possible error patterns are equally likely, then the probability that an $n$-bit signature generator will not detect an error is

$$P(M) = \frac{2^{m-n}}{2^m - 1}$$

where $2^m - 1$ is the total possible erroneous streams. For $m >> n$, $P(M)$ will approach $2^{-n}$.

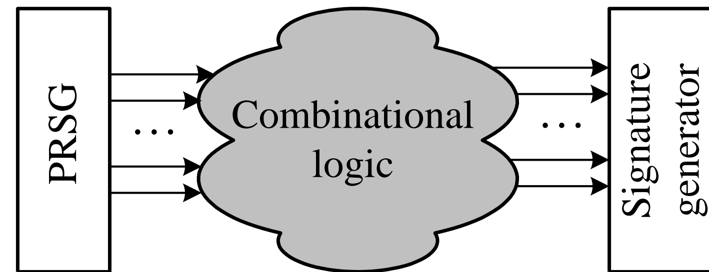
# A BILBO (built-in block observer) Example




(a) Logic diagram

| M1 | M0 | Function |
|---|---|---|
| 0 | 0 | Scan mode |
| 0 | 1 | MISR |
| 1 | 0 | Clear register |
| 1 | 1 | Parallel load |

(b) Function selection



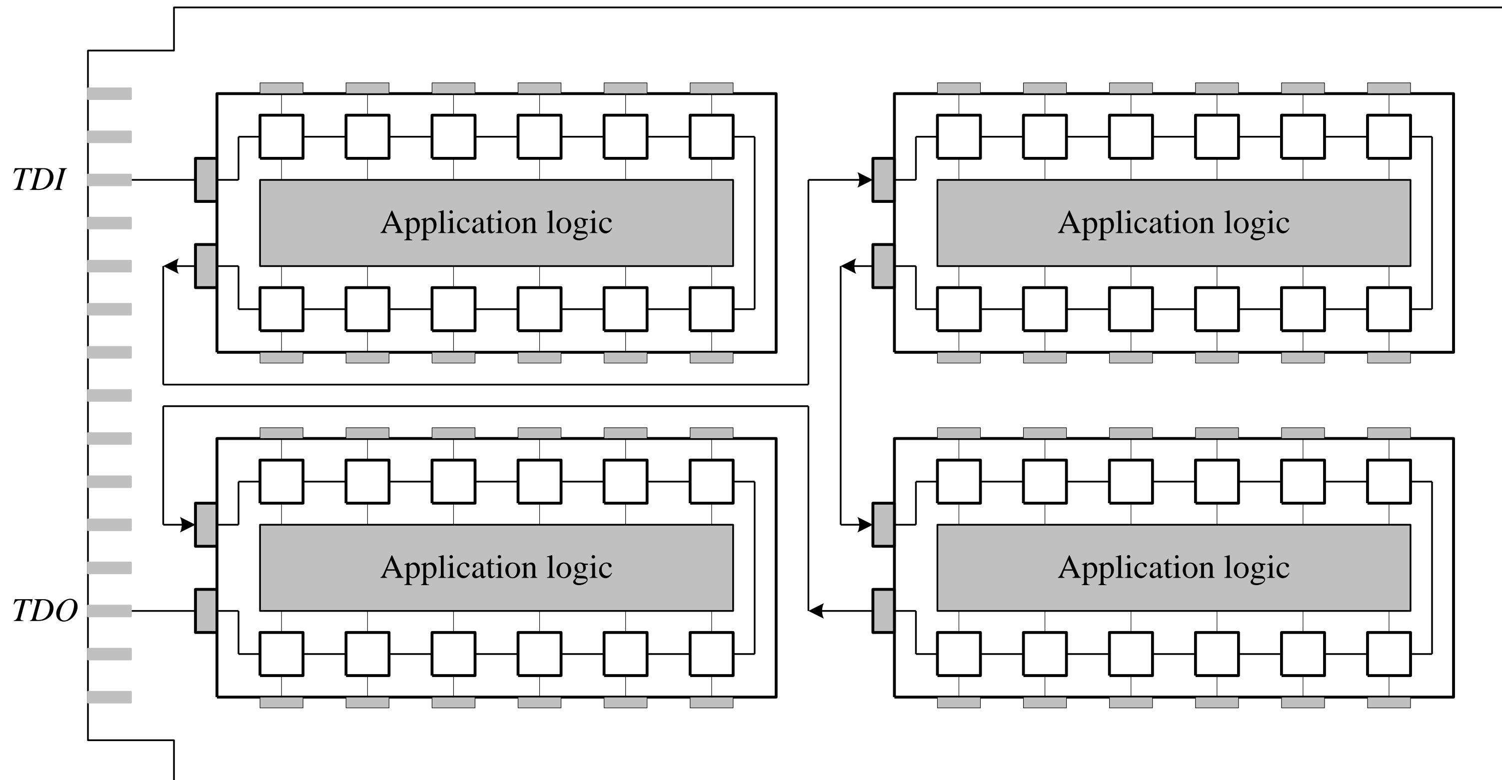

(c) Application example

# IEEE 1149.1 --- The Boundary Scan Standard

- The goals of the boundary scan standard are to :
  - provide a data transfer standard between ATE and the devices on a PCB,
  - provide a method of interconnecting devices on a PCB,
  - provide a way of using test bus standard or BIST hardware to find the faulty devices on a PCB.
- Functions of boundary scan standard:
  - interconnection test
  - normal operation data observation
  - each device test

# IEEE 1149.1 --- An Application Example

- A boundary scan architecture used to test the entire board-level module.

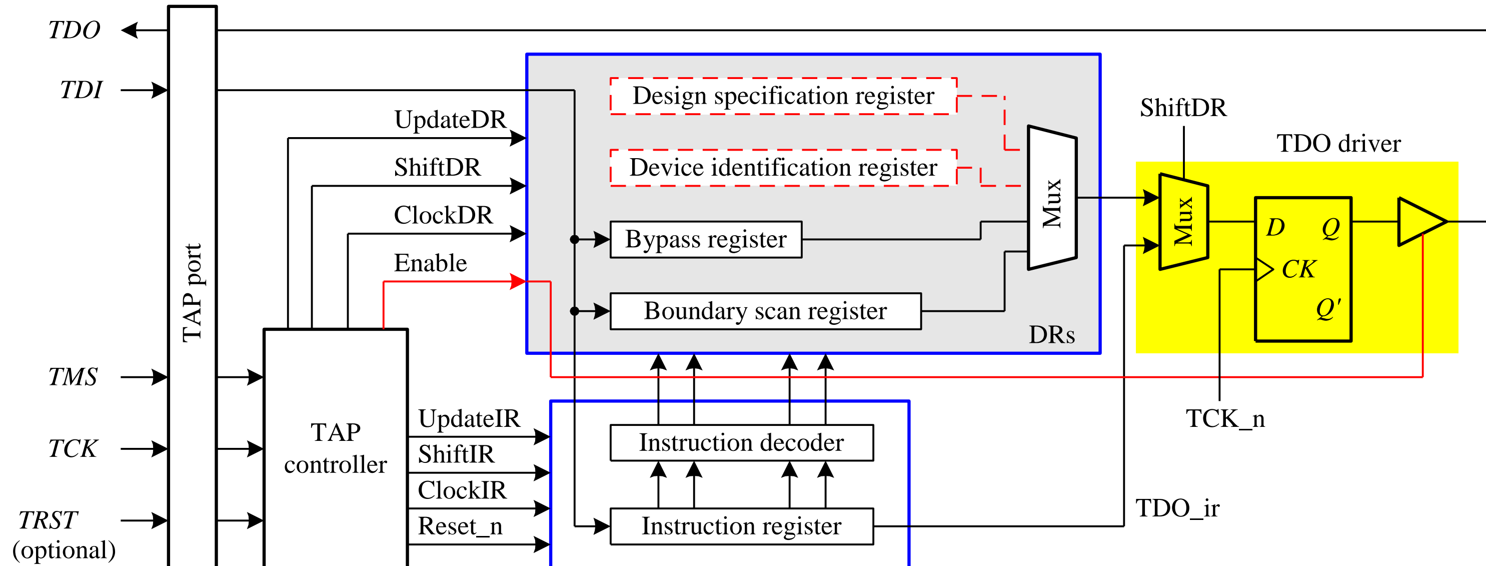


# IEEE 1149.1 --- The Structure

- The boundary scan standard mainly include five parts:
  - test access port (TAP),
  - data registers,
  - TDO driver,
  - instruction register and decoder
  - TAP controller.

# IEEE 1149.1 --- The TAP Structure

- TAP (test access port) consists of:
  - Test data registers (test DRs)
  - Instruction register and decoder
  - TAP controller
  - TDO driver



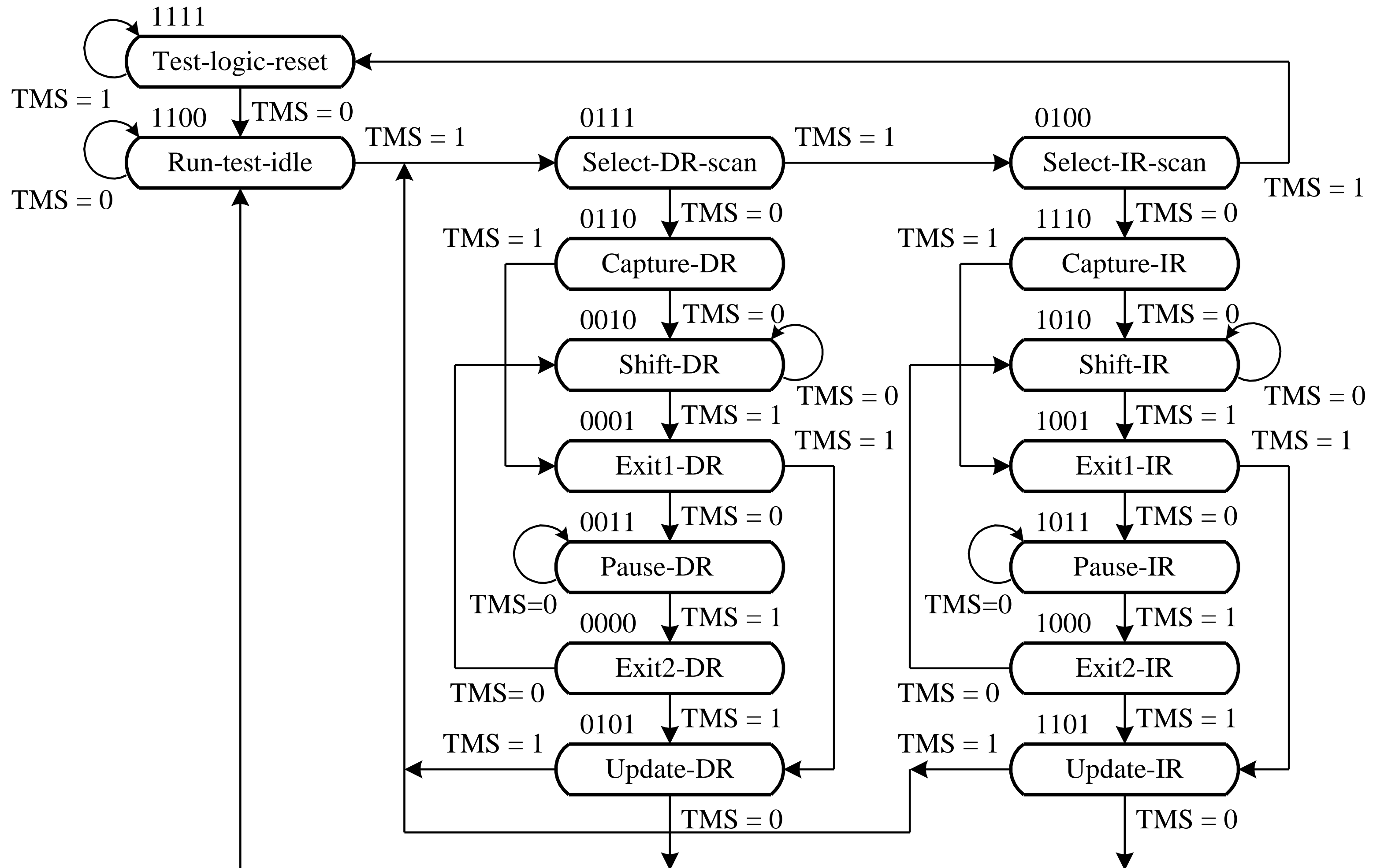
# IEEE 1149.1 --- Control Signals

- Four control signals:
  - TCK (test clock, input) clocks tests into and out of the chip.
  - TDI (test data input, input) input test data into the chip.
  - TDO (test data output, output) output test data out of the chip.
  - TMS (test mode select, input) controls the test operations.
- One optional reset signal
  - TRST_n (test reset) is an optional active-low asynchronous reset signal, used to reset TAP controller externally.

# IEEE 1149.1 --- Control Signals

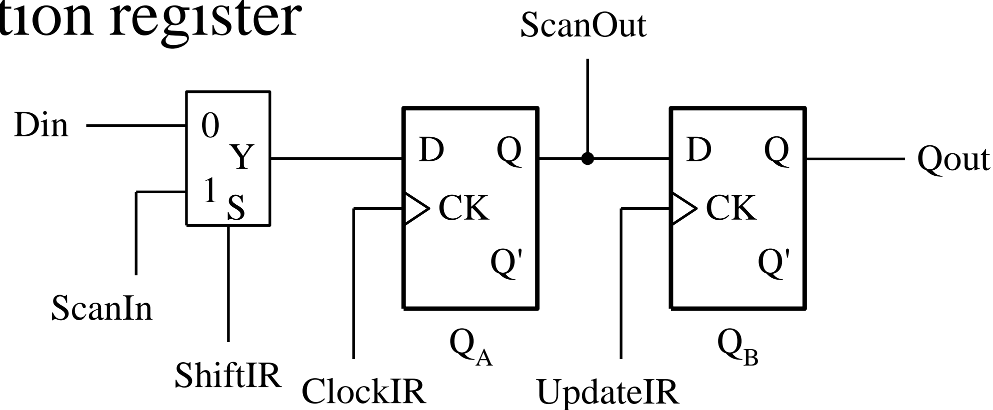
- In the normal mode, TRST_n and TCK are held low and TMS is held high to disable boundary scan.
- To prevent race conditions, inputs are sampled on the positive edge of TCK and output toggle on the negative edge.

# IEEE 1149.1 --- The State Machine

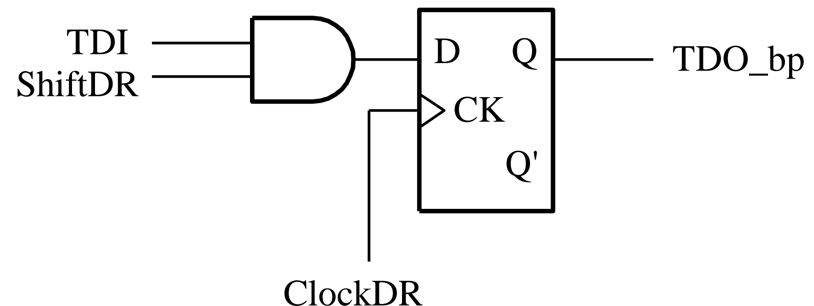


# IEEE 1149.1 --- Instruction and Bypass Registers

- Instruction register cell
  - Shift register
  - Instruction register

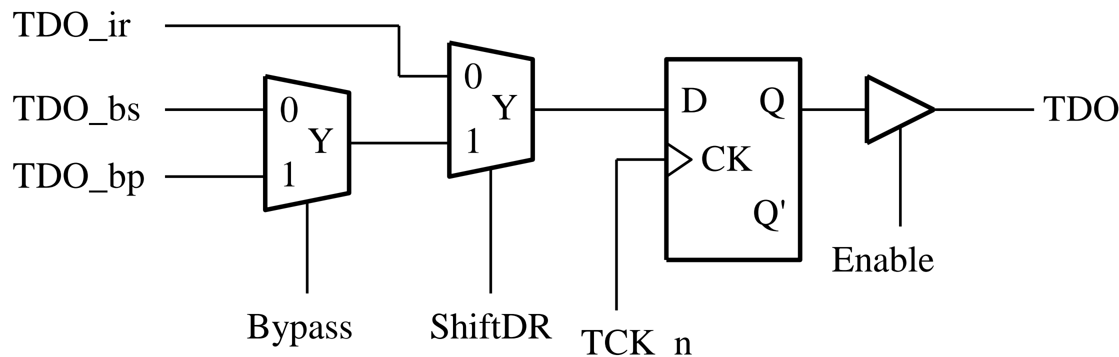



- Bypass register structure



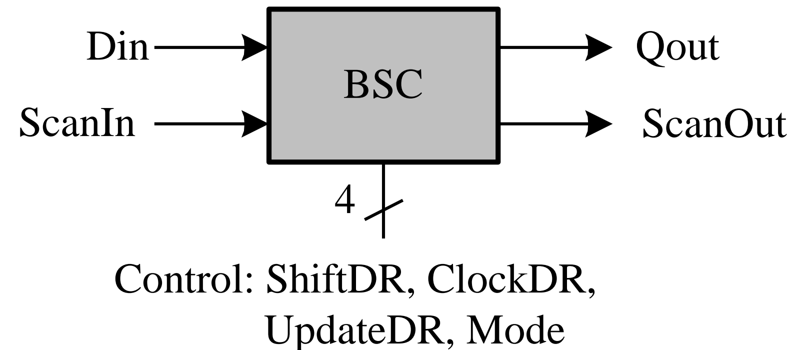
# IEEE 1149.1 --- A TDO Driver

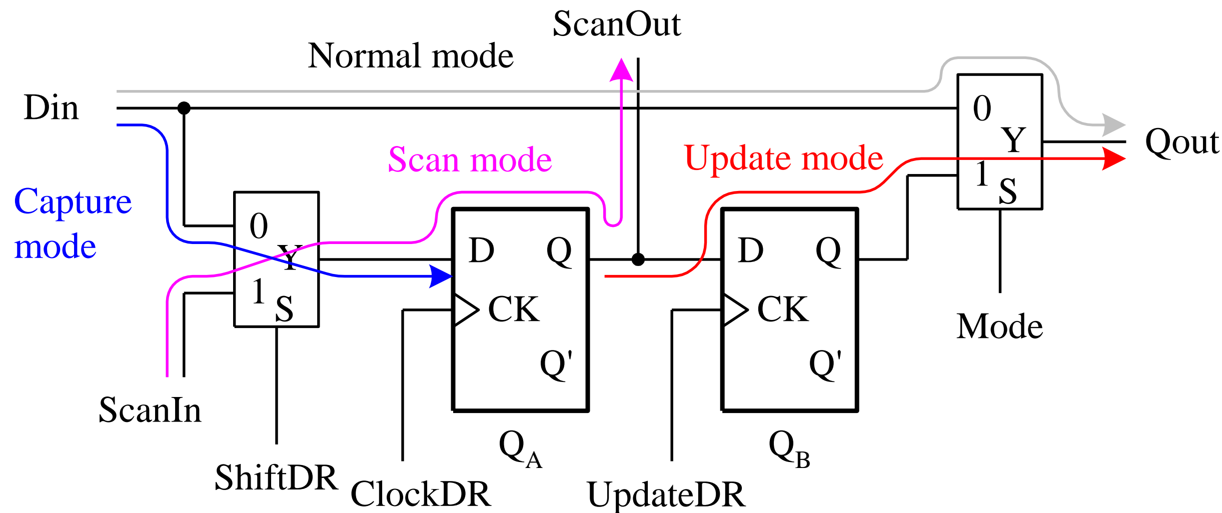
- TDO Driver
  - TMS is sampled at the positive edge of TCK.
  - TDO is sampled at the negative edge of TCK.



# IEEE 1149.1 --- A Boundary Scan Cell

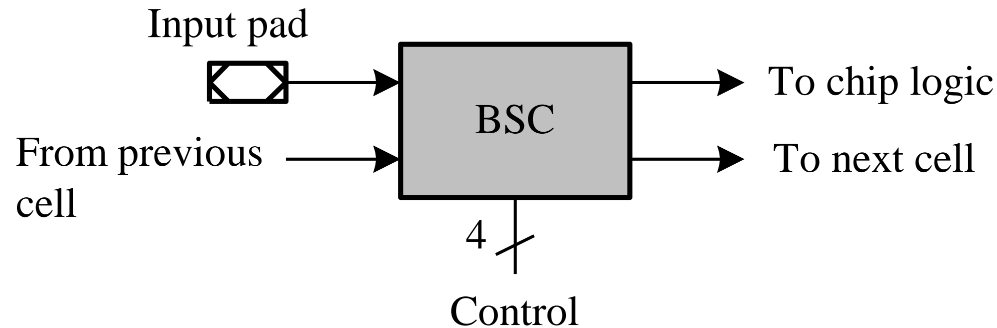
- Operation modes of boundary scan cell (BSC):
  - Normal mode
  - Capture mode
  - Scan mode
  - Update mode



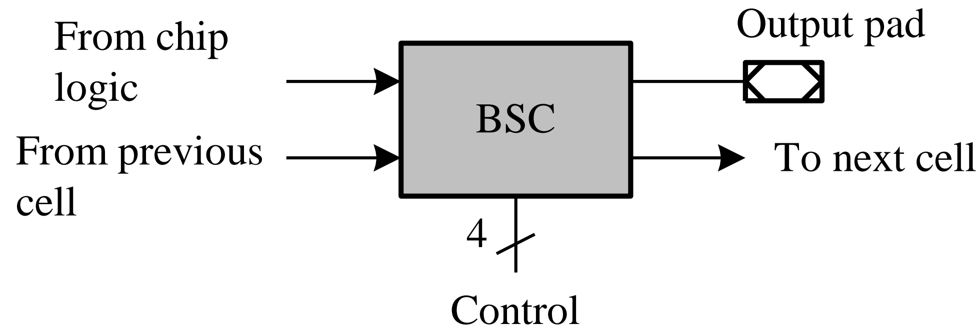



# IEEE 1149.1 --- A Boundary Scan Cell

- Boundary scan input pad configuration

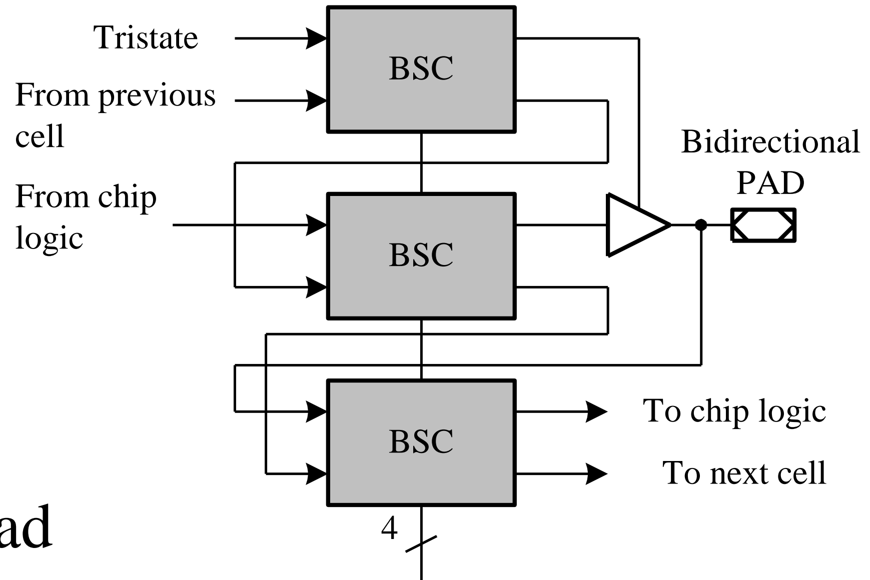



- Boundary scan output pad configuration

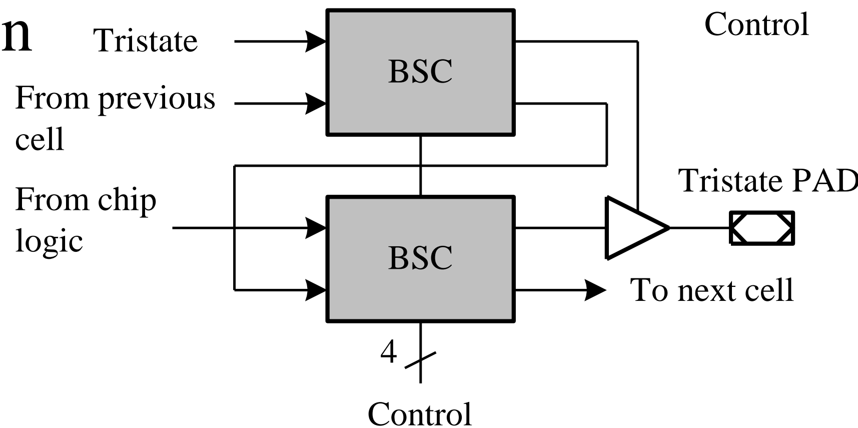


# IEEE 1149.1 --- A Boundary Scan Cell

- Boundary scan bidirectional pad configuration




- Boundary scan tri-state pad configuration



# IEEE 1149.1 --- A Complete Example