CPEG 422/622 EMBEDDED SYSTEMS DESIGN

MW 3:35 – 4:50, SHARP 116

Chengmo Yang

chengmo@udel.edu

Evans 201C

LECTURE 14
HARDWARE TEST

OUTLINE

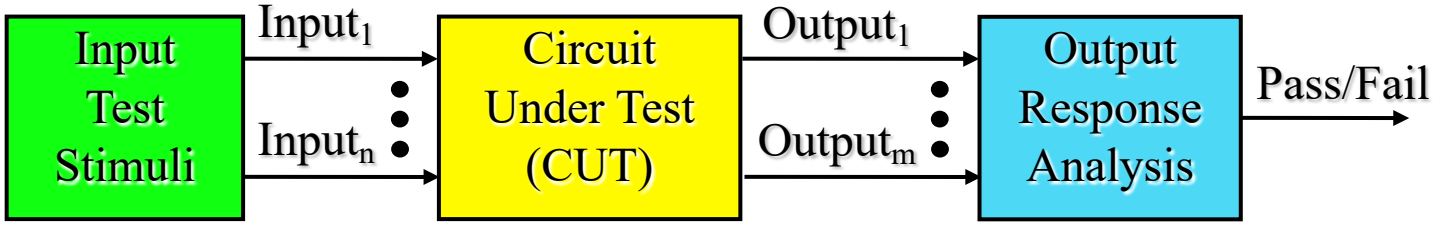
- VLSI test
- Fault models
- Test combinational circuits
- Test sequential circuits

IMPORTANCE OF TESTING

- Moore's Law results from decreasing feature size (dimensions)
  - from 10s of μm to 10s of nm for transistors and interconnecting wires
- Operating frequencies have increased from 100KHz to several GHz
- Decreasing feature size increases probability of defects during manufacturing process
  - A single faulty transistor or wire results in faulty IC
  - Testing required to guarantee fault-free products

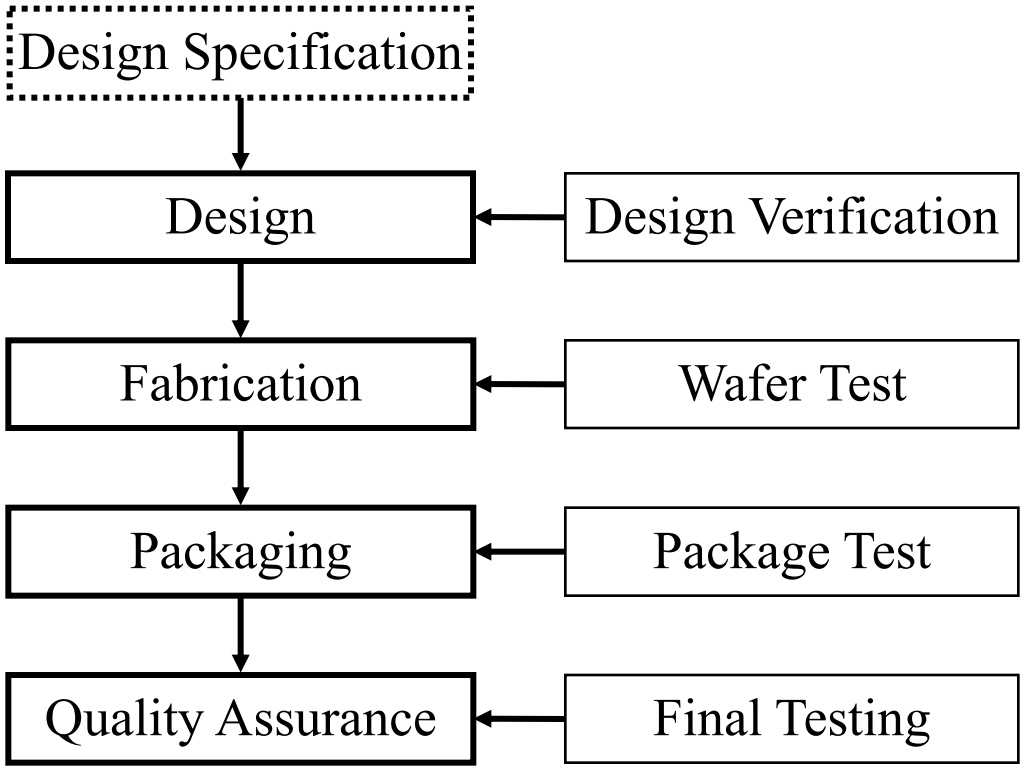
TESTING DURING VLSI LIFE CYCLE

- Testing typically consists of
  - Applying set of test stimuli to
  - Inputs of *circuit under test* (CUT), and
  - Analyzing output responses
    - If incorrect (fail), CUT assumed to be faulty
    - If correct (pass), CUT assumed to be fault-free



TESTING DURING VLSI DEVELOPMENT

- Design verification targets design errors
  - Corrections made prior to fabrication
- Remaining tests target manufacturing defects
  - A defect is a flaw or physical imperfection that can lead to a fault



YIELD AND REJECT RATE

- We expect faulty chips due to manufacturing defects
  - Called yield

$$yield = \frac{number\ of\ acceptable\ parts}{total\ number\ of\ parts\ fabricated}$$

- Undesirable results during testing
  - Faulty chip appears to be good (passes test) ***False negative***
    - Due to inability to achieve 100% fault coverage
  - Good chip appears to be faulty (fails test) ***False positive***
    - Due to poorly designed tests or lack of DFT (design-for-test) tools

TEST GENERATION

- A test is a sequence of test patterns, called test vectors, applied to the CUT whose outputs are monitored and analyzed for the correct response
  - Exhaustive testing – applying all possible test patterns to CUT
  - Functional testing – testing every truth table entry for a combinational logic CUT
    - Neither of these are practical for large CUTs

- Fault coverage is a quantitative measure of quality of a set of test vectors

TEST GENERATION

- Fault coverage for a given set of test vectors

$$fault\ coverage = \frac{number\ of\ detected\ faults}{total\ number\ of\ faults}$$

- 100% fault coverage may be impossible due to undetectable faults

$$fault\ detection\ efficiency = \frac{number\ of\ detected\ faults}{total\ number\ of\ faults - number\ of\ undetectable\ faults}$$

- ***Reject rate*** = 1 – ***yield***$^{(1 - \textit{fault coverage})}$
  - A PCB with 40 chips, each with 90% fault coverage and 90% yield, has a reject rate of 41.9%
    - Or 419,000 defective parts per million (PPM)

TYPES OF VLSI TESTING

- ***On-line testing*** – concurrent with system operation

- ***Off-line testing*** – while system (or portion of) is taken out of service
  - Performed periodically during low-demand periods
  - Used for diagnosis (identification and location) of faulty replaceable components to improve repair time

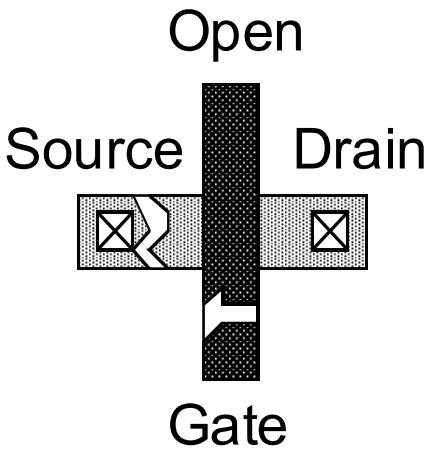
OUTLINE

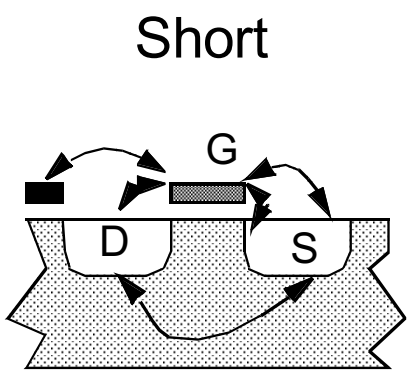
- VLSI test
- **Fault models**
- Test combinational circuits
- Test sequential circuits

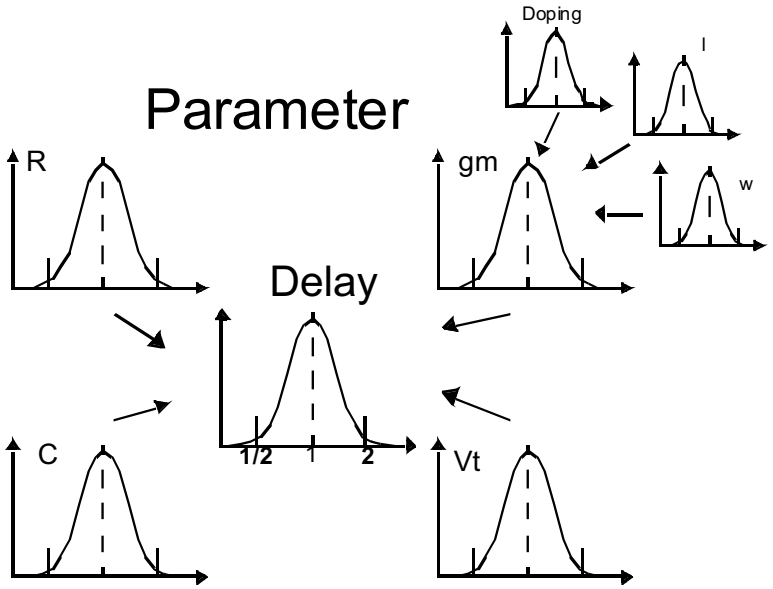
DEFECT AND FAULTS

Fault types: Functional.
Timing.

Abstraction level: Transistor. (layout)
Gate. (netlist)
Macro (functional blocks).

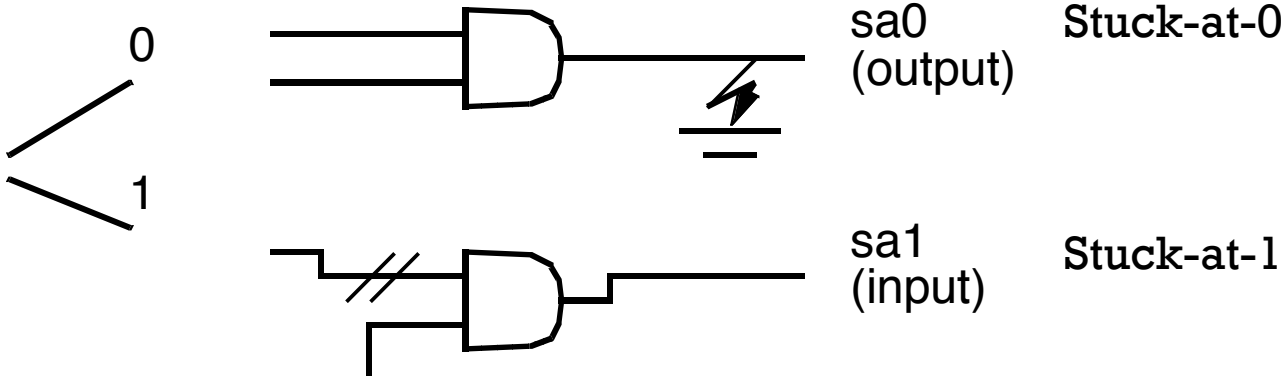




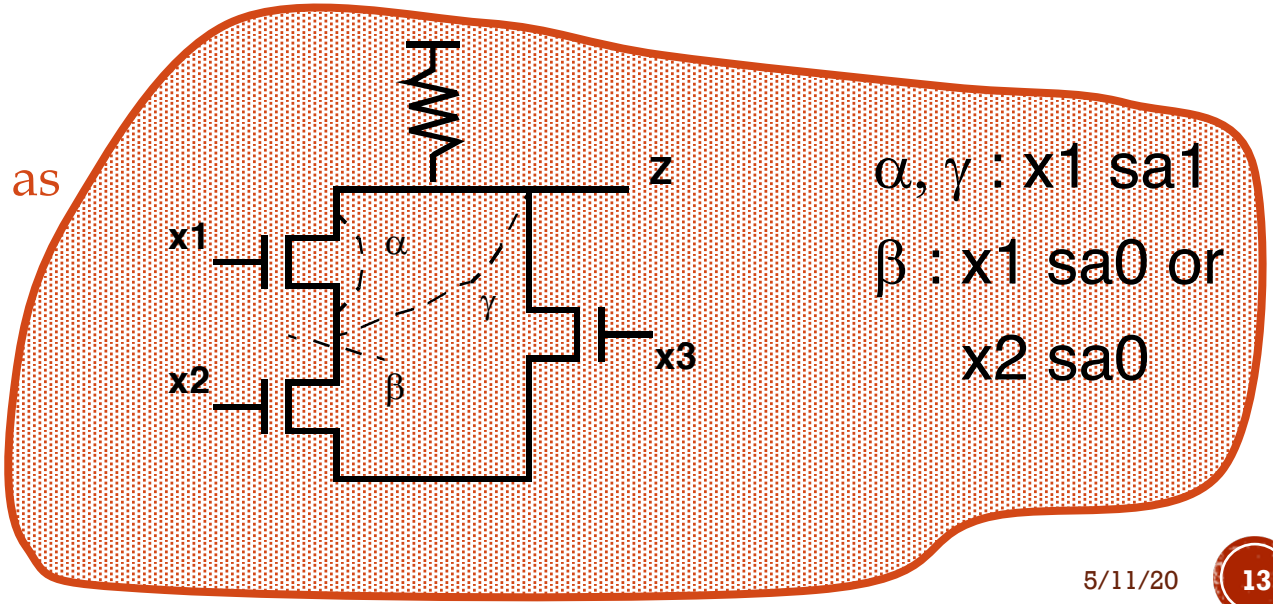


LOGICAL-LEVEL FAULT MODEL

Most Popular - "Stuck - at" model



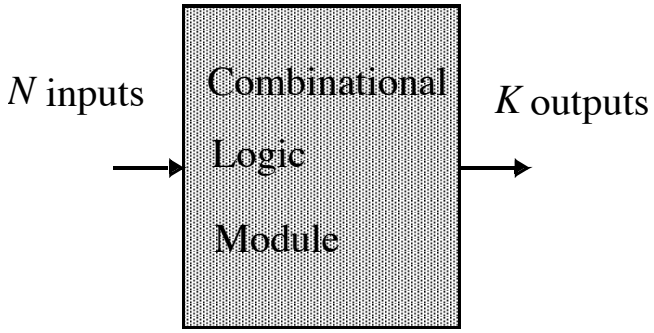
Covers many other occurring faults, such as opens and shorts.



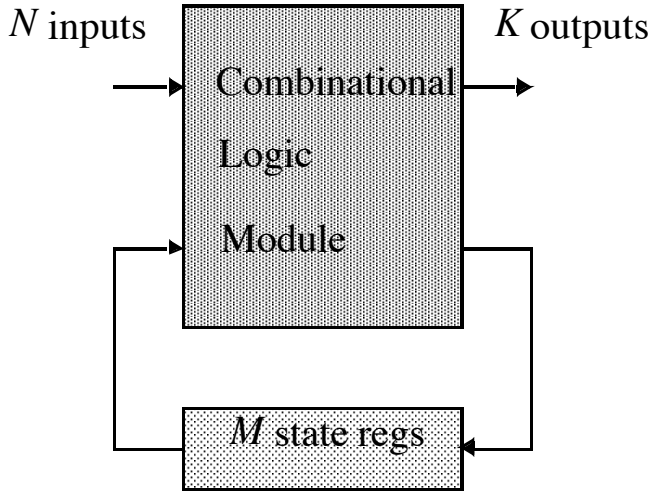
TESTING APPROACHES

1. Exhaustive Algorithm

- For n-input circuit, generate all 2^n input patterns
- Infeasible, unless circuit has < 20 inputs



(a) Combinational function



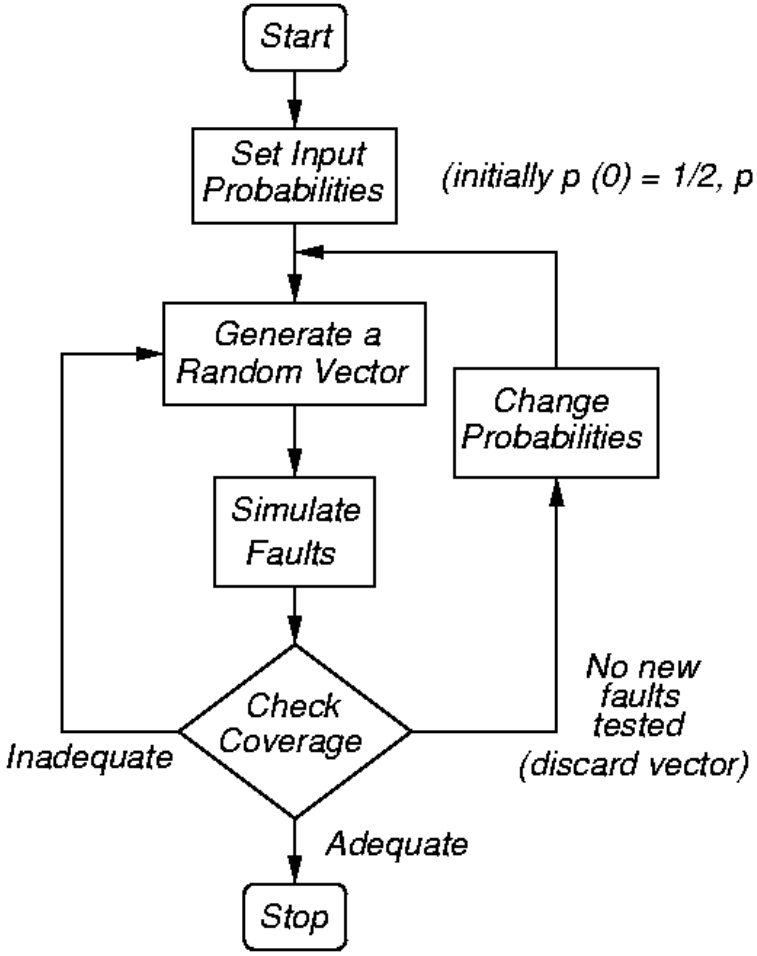
(b) Sequential engine

2^N patterns

2^{N+M} patterns

TESTING APPROACHES

2. **Random Pattern Generation**
   - Flow chart for method
   - Use to get tests for 60-80% of faults, then switch to D-algorithm or other approaches for the rest

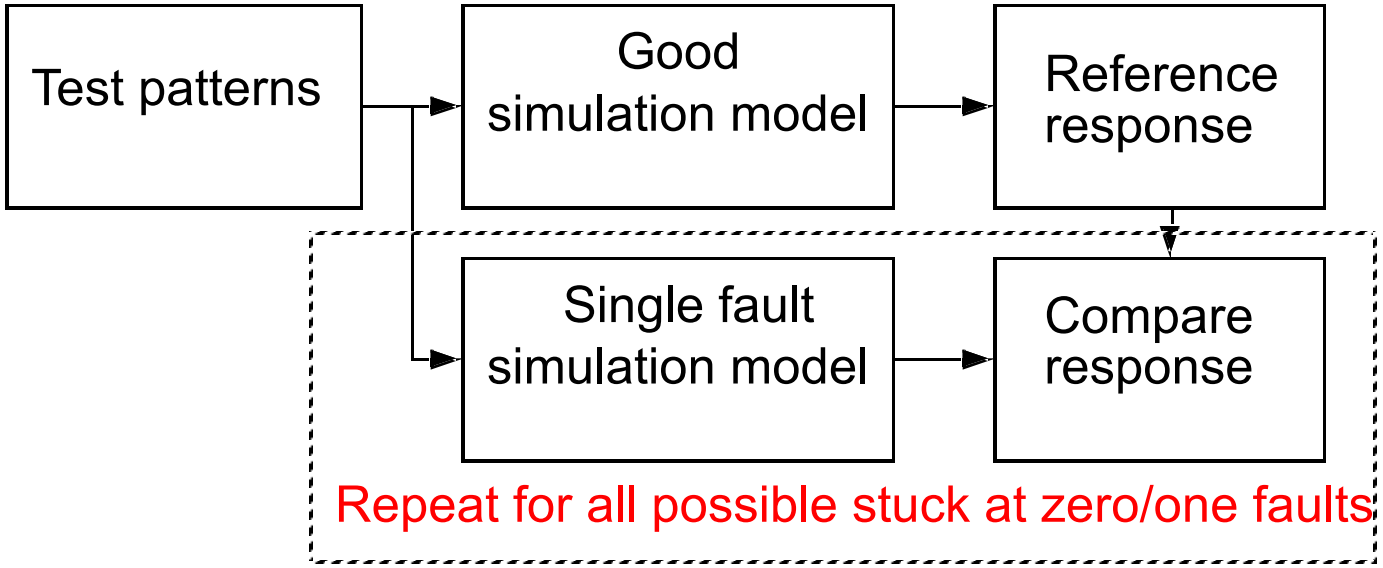


Q: How to estimate fault coverage?

A: Through Fault Simulation

FAULT SIMULATION

Goal: Emulates fault models in CUT and applies test vectors to determine fault coverage

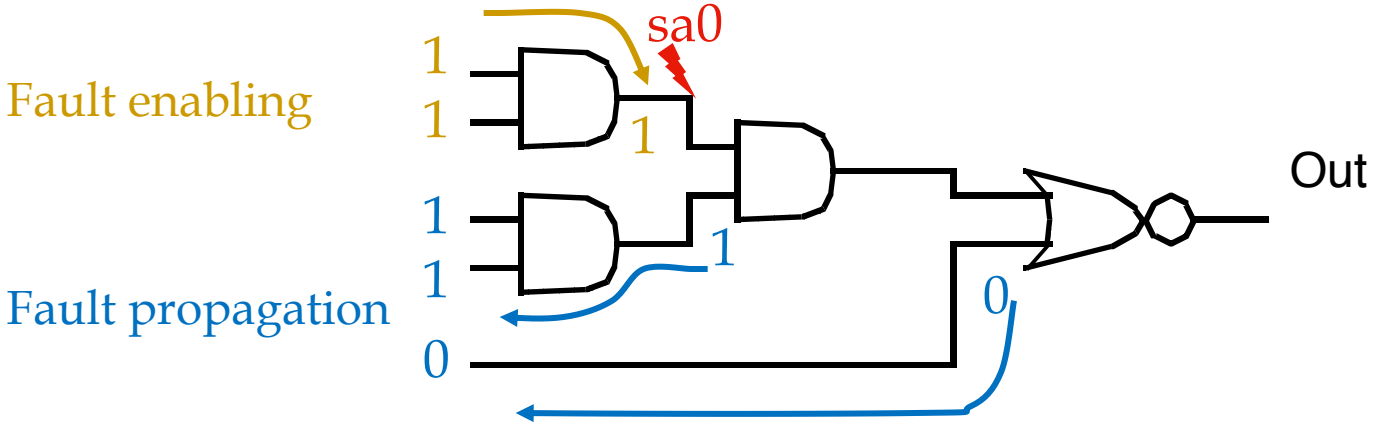


Requires long simulation times!

FAULT SIMULATION

- **Basic Terms:**
  - **Controllability:** the ease of controlling the state of a node
  - **Observability:** the ease of observing the state of a node

- **Goal:** determine input pattern that
  - makes a fault controllable (triggers the fault)
  - makes its impact visible at the output nodes



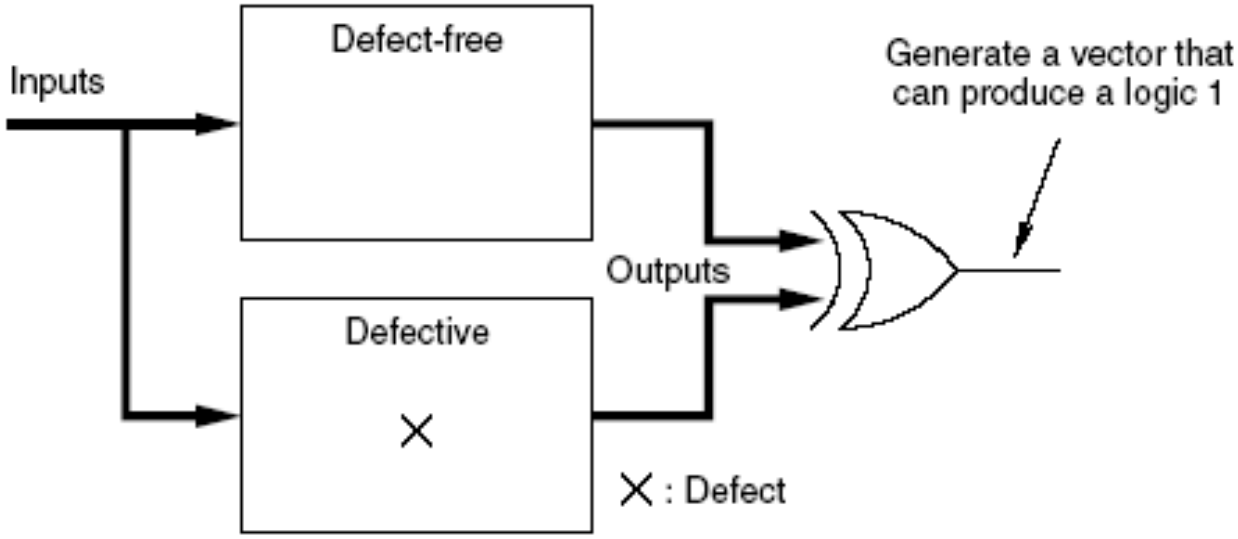
Technique Used: **D-algorithm**

OUTLINE

- VLSI test
- Fault models
- **Test combinational circuits**
- Test sequential circuits

TEST GENERATION

- Generate an input vector that can distinguish the defect-free circuit from the hypothetically defective one



- *Goal*: find efficient set of test vectors with maximum fault coverage
- **Automatic Test Pattern Generation (ATPG):** Algorithms generating sequence of test vectors for a given circuit based on specific fault models
  - E.g., D-algorithm

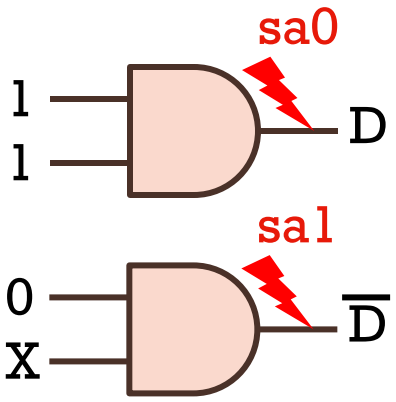
D-ALGORITHM

- Models circuit faults
  - Stuck-at-0
  - Stuck-at-1

- 5-Value Logic
  - 0 – binary 0 in both good and fault circuit
  - 1- binary 1 in both good and fault circuit
  - X – don't care
  - **D** – binary 1 in good circuit, 0 in bad circuit
  - $\overline{\mathbf{D}}$ – binary 0 in good circuit, 1 in bad circuit

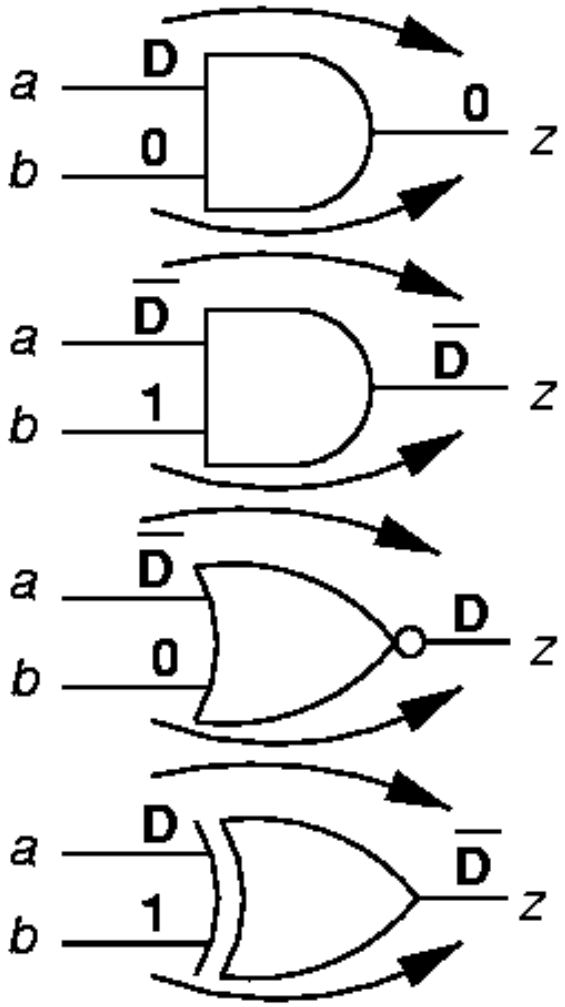
D-ALGORITHM

- ***Primitive D-Cube*** of Failure
  - AND Output sa0: "1 1 D"
  - AND Output sa1: "0 X $\overline{D}$ "
    or "X 0 $\overline{D}$ "
  - Wire sa0: "D"



- ***Propagation D-cube*** – model conditions under which fault effect propagates through gate

FORWARD IMPLICATION



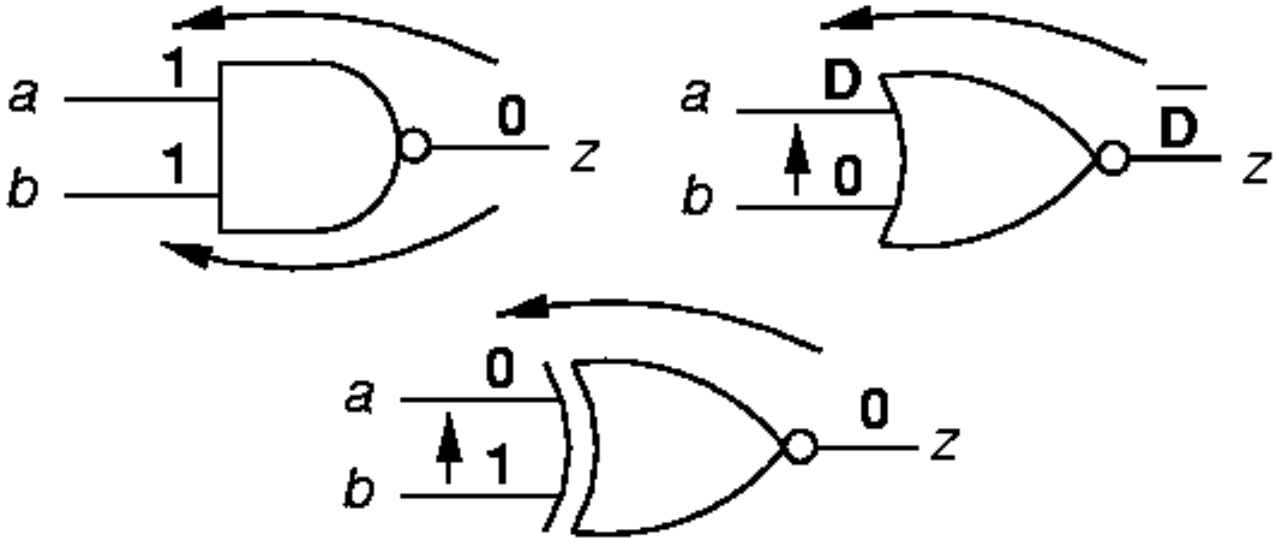
- Results in logic gate inputs that uniquely determine the output

- AND gate forward implication table:

a \ b	0	1	X	D	$\overline{D}$
0	0	0	0	0	0
1	0	1	X	D	$\overline{D}$
X	0	X	X	X	X
D	0	D	X	D	0
$\overline{D}$	0	$\overline{D}$	X	0	$\overline{D}$

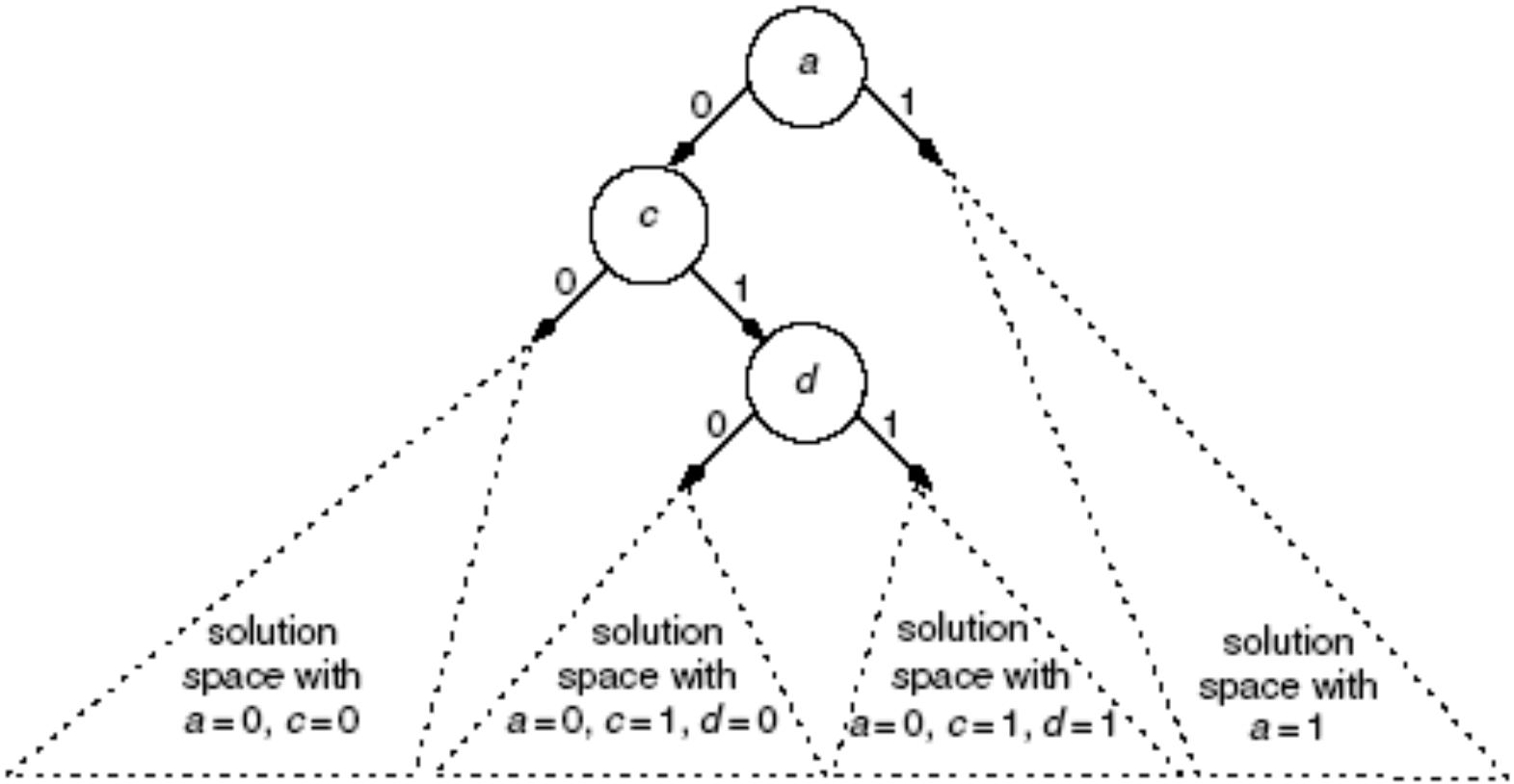
BACKWARD IMPLICATION

- Unique determination of all gate inputs when the gate output and some of the inputs are given



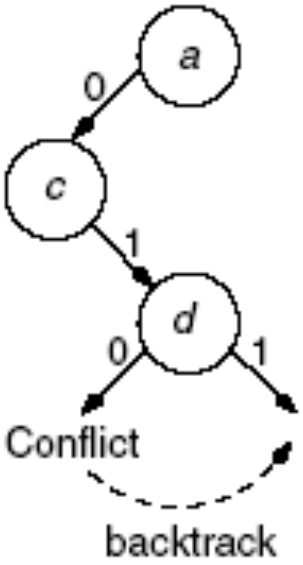
BRANCH-AND-BOUND SEARCH

- The ATPG systematically and implicitly searches the entire search space



BACKTRACKING

- The ATPG searches one branch at a time
- Whenever a conflict (e.g., all D's disappeared) arises, must backtrack on previous decisions

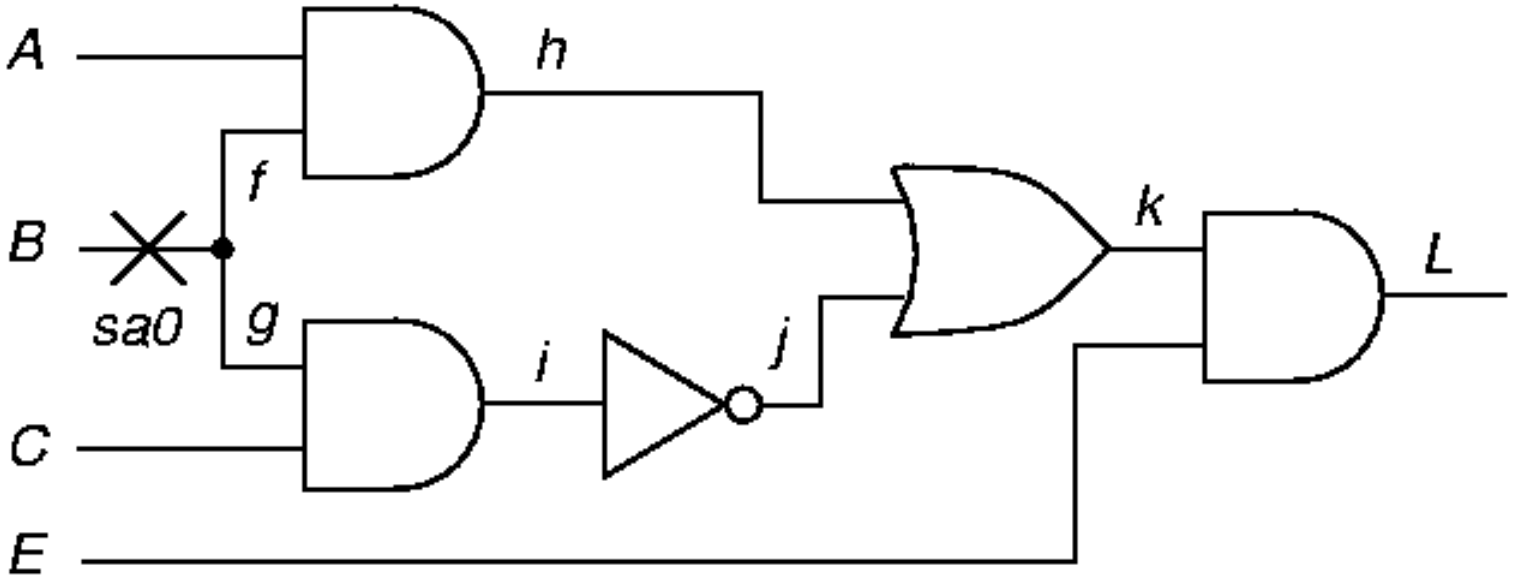


If d=1 also causes a conflict, backtrack to c=0

EXAMPLE

Steps:

1. Fault Sensitization
2. Fault Propagation
3. Line Justification

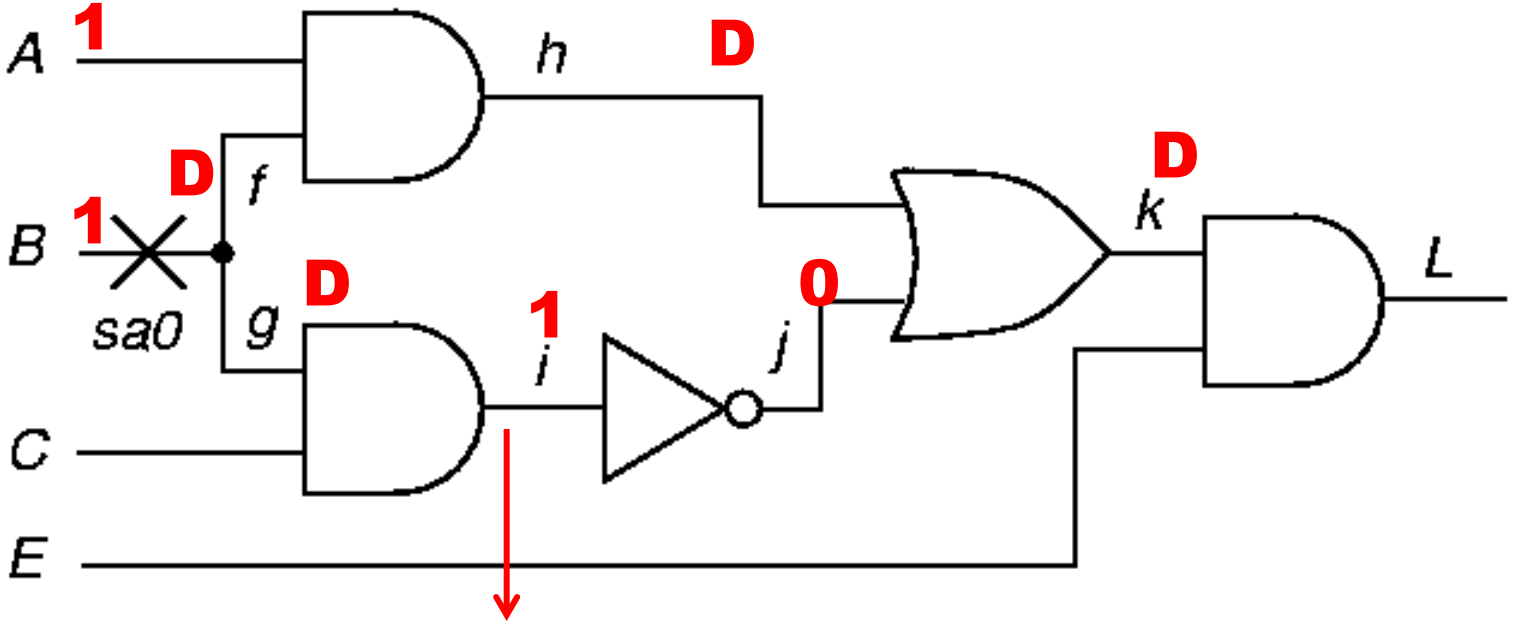


EXAMPLE

Steps:

1. Fault Sensitization
2. Fault Propagation
3. Line Justification

First try path *f – h – k – L*



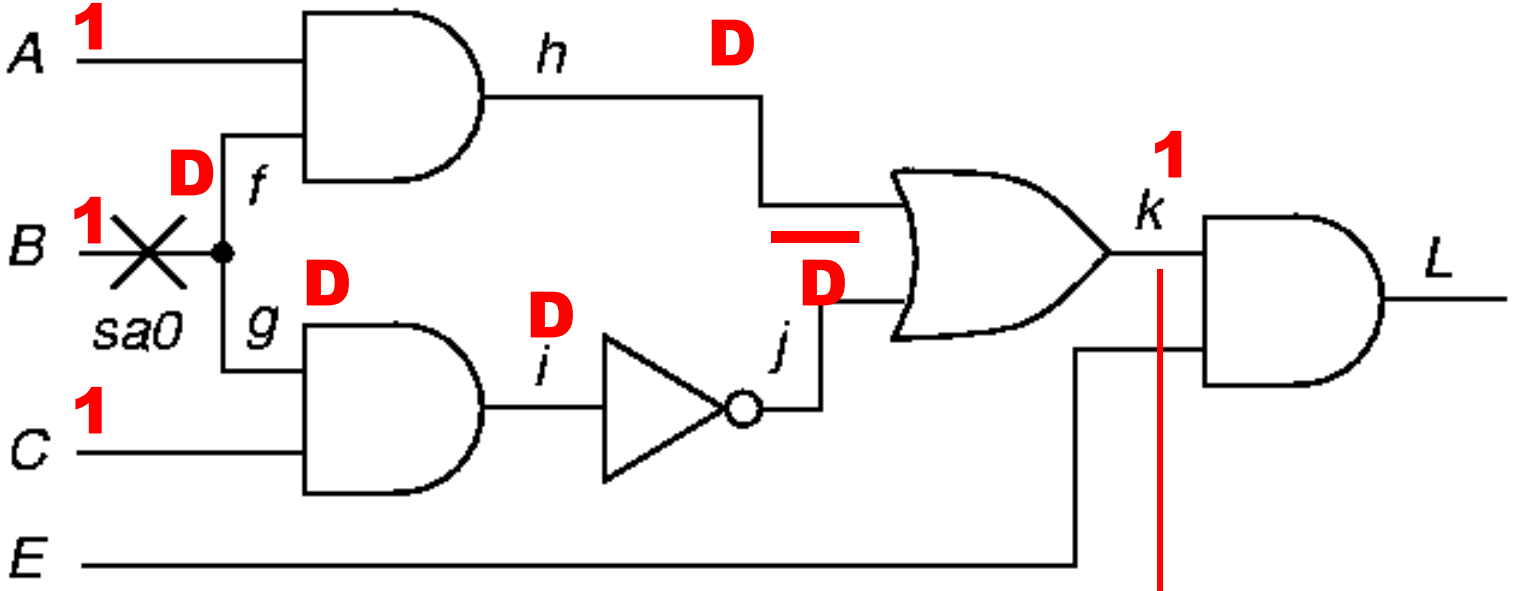
blocked, since there is no way to justify the 1 on *i*

EXAMPLE

Steps:

1. Fault Sensitization
2. Fault Propagation
3. Line Justification

Then try both paths *f* – *h* – *k* – *L* and *g* – *i* – *j* – *k* – *L*



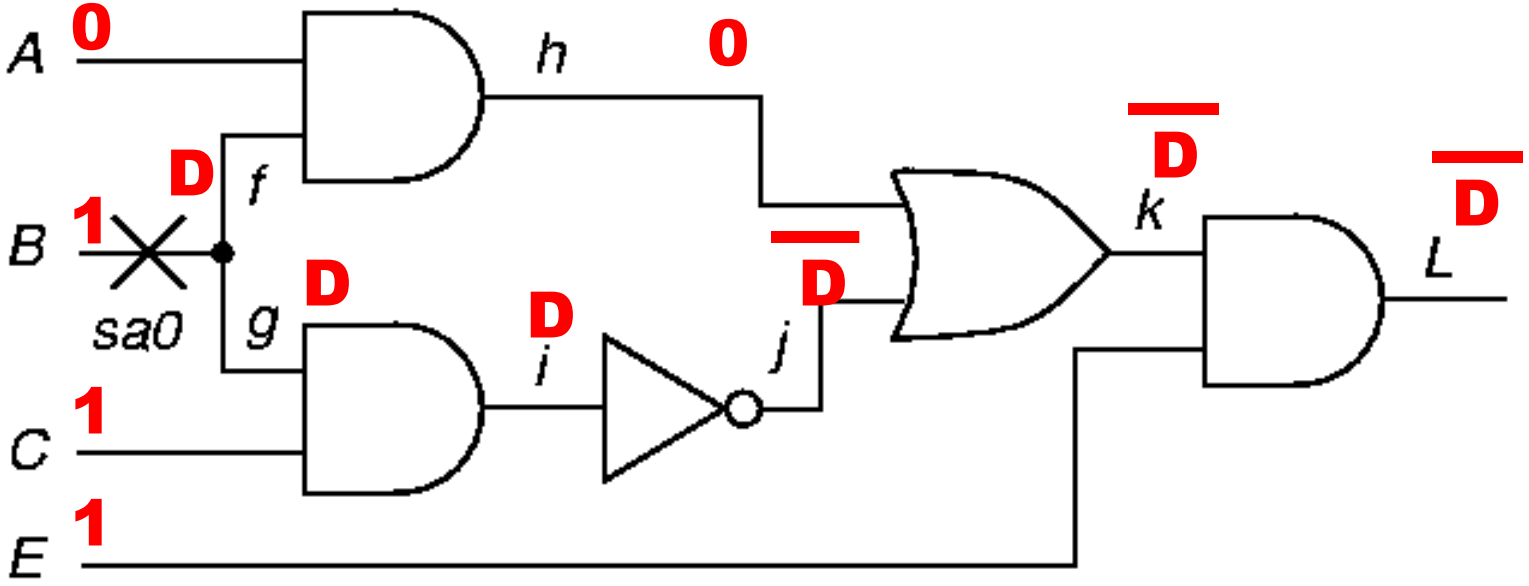
blocked at *k* because *D* disappears (not observable)

EXAMPLE

Steps:

1. Fault Sensitization
2. Fault Propagation
3. Line Justification

Final try: path *g – i – j – k – L*



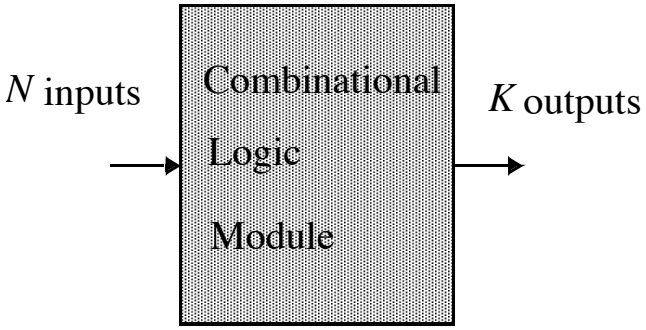
Test Found! Test Vector is 0111

OUTLINE

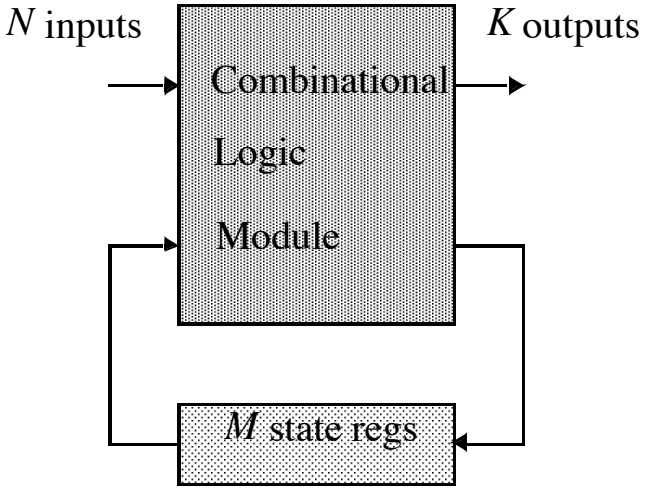
- VLSI test
- Fault models
- Test combinational circuits
- **Test sequential circuits**

SEQUENTIAL CIRCUITS: STATE!

- Combinational circuit testing is relatively easy
- Sequential circuit testing needs to drive sequential elements to specific state to test a fault



(a) Combinational function



(b) Sequential engine

2^N patterns

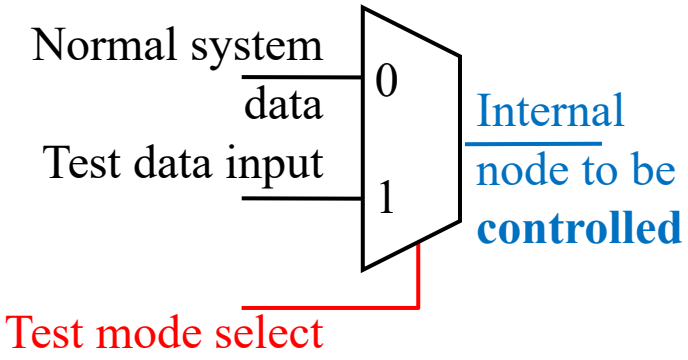
2^{N+M} patterns

DESIGN FOR TESTABILITY (DFT)

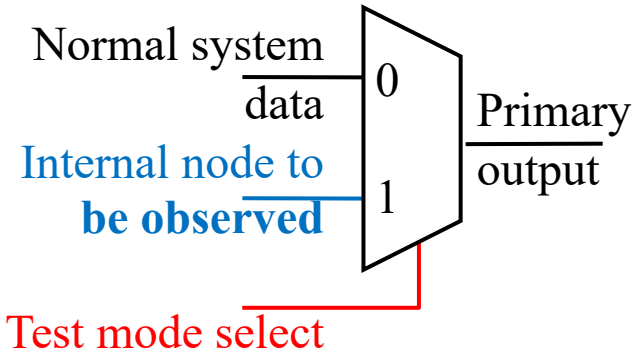
- Generally incorporated in design
- Goal: improve controllability and/or observability of internal nodes of a chip or PCB
- Three basic approaches
  - Ad-hoc techniques
  - Scan design
  - Built-In Self-Test (BIST)

AD-HOC TEST

- Ad-hoc DFT techniques
  - Add internal test points (usually multiplexers) for
    - Controllability
    - Observability
  - Added on a case-by-case basis
    - Primarily targets "hard to test" portions of chip

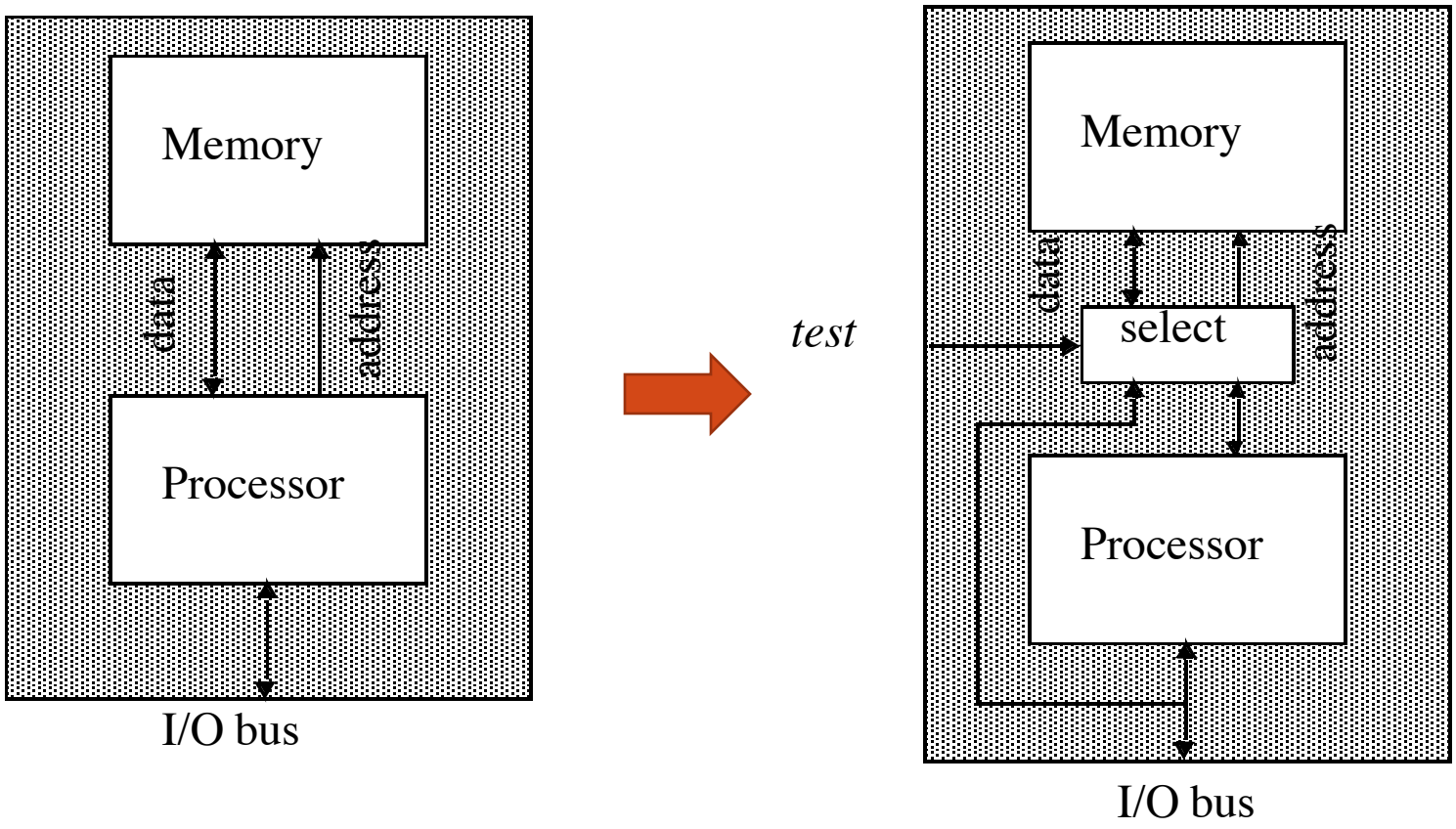


controllability test point



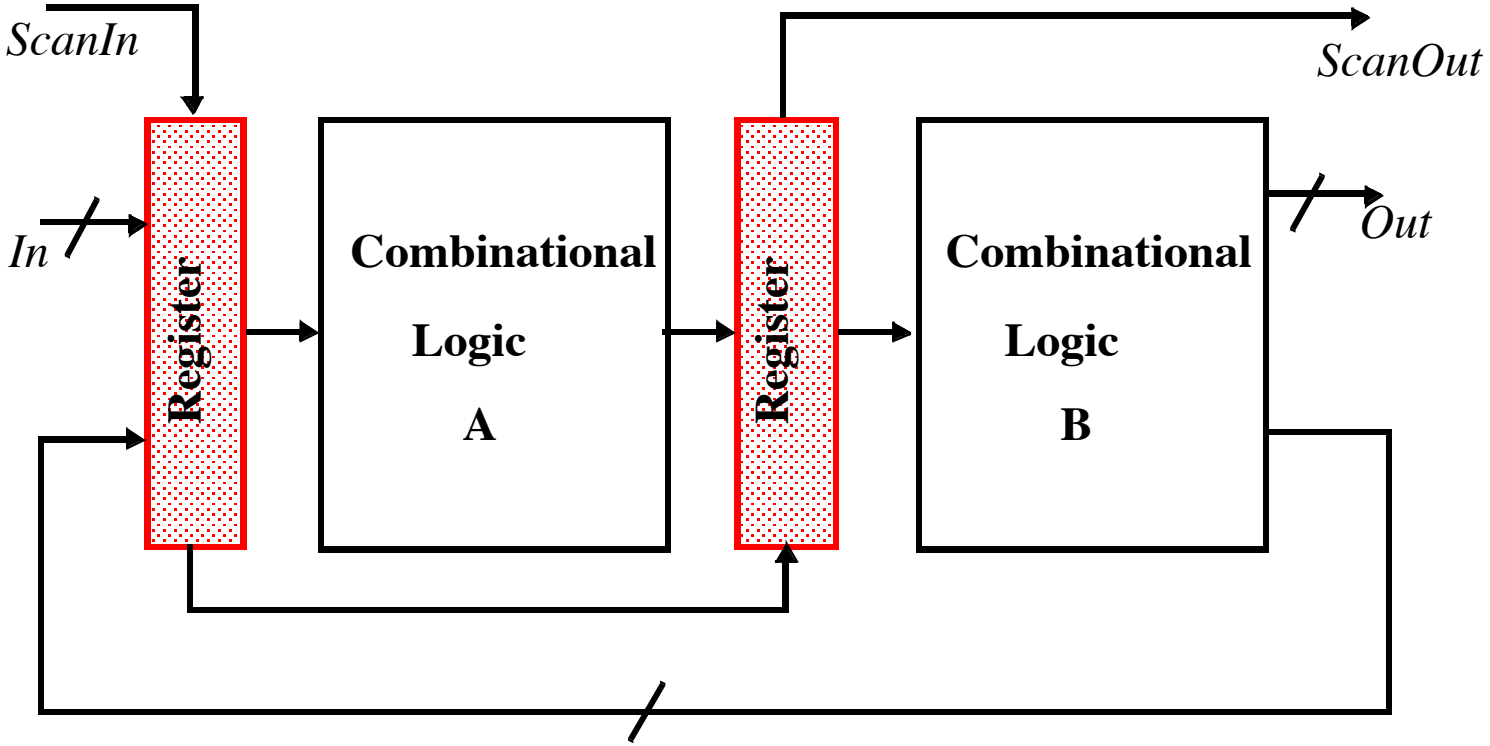
observability test point

AD-HOC TEST



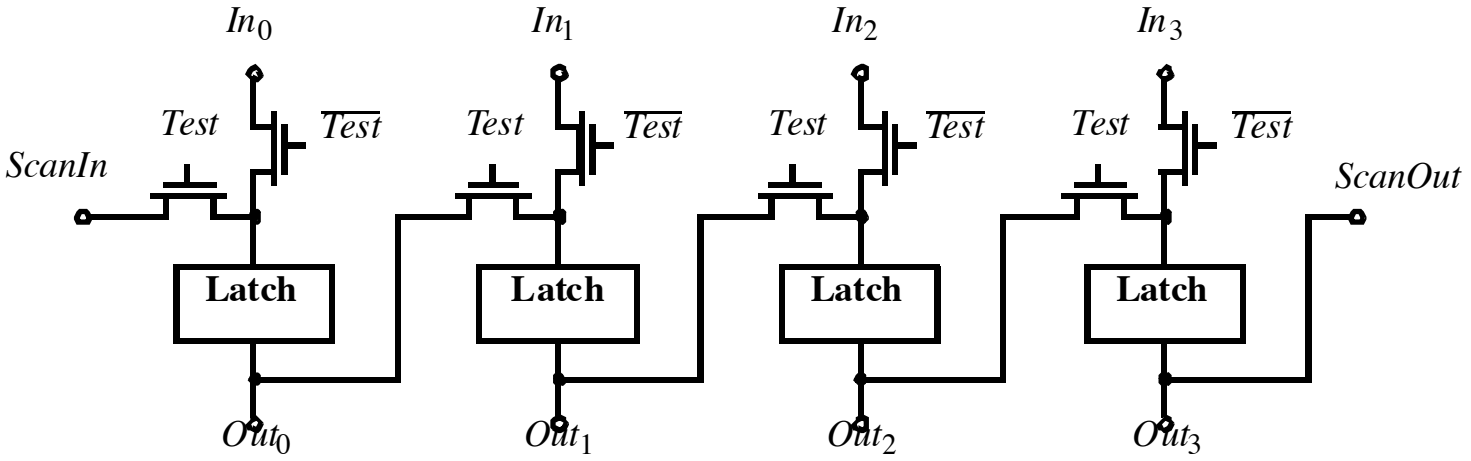
Inserting multiplexer improves testability

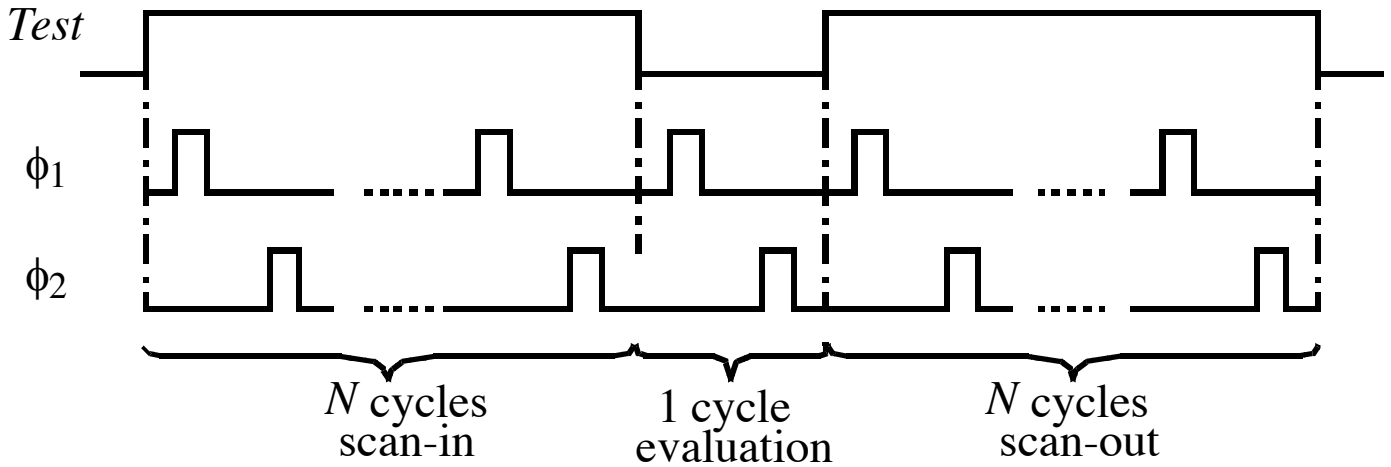
SCAN-BASED TEST



- Connect all flip-flops in the design as a chain (called scan chain).
- In test mode, first scan desired values into those flip-flops, then apply the value, and finally scan out the response.

SCAN-BASED TEST — OPERATION





SCAN ARCHITECTURES

- Full-Scan Design
  - All or almost all storage element are converted into scan cells and combinational ATPG is used for test generation
- Partial-Scan Design
  - A subset of storage elements are converted into scan cells and sequential ATPG is typically used for test generation
- Random-Access Scan Design
  - A random addressing mechanism, instead of serial scan chains, is used to provide direct access to read or write any scan cell

BUILT-IN SELF-TEST (BIST)

- Incorporates test pattern generator (TPG) and output response analyzer (ORA) internal to design
  - Chip can test itself
- Typically use a linear-feedback shift register (LFSR) to generate random test patterns
- Can be used at all levels of testing
  - Device → PCB → system → field operation