CMOS Testing: Part 1

- Introduction
- Fault models
  - Stuck-line (single and multiple)
  - Bridging
  - Stuck-open
- Test pattern generation
  - Combinational circuit test generation
  - Sequential circuit test generation

Outline

- Testing
  - Logic Verification
  - Silicon Debug
  - Manufacturing Test
- Fault Models
- Observability and Controllability
- Design for Test
  - Scan
  - BIST
- Boundary Scan

Testing

- Testing is one of the most expensive parts of chips
  - Logic verification accounts for > 50% of design effort for many chips
  - Debug time after fabrication has enormous cost
  - Shipping defective parts can sink a company
- Example: Intel FDIV bug
  - Logic error not caught until > 1M units shipped
  - Recall cost $450M (!!!)

Logic Verification

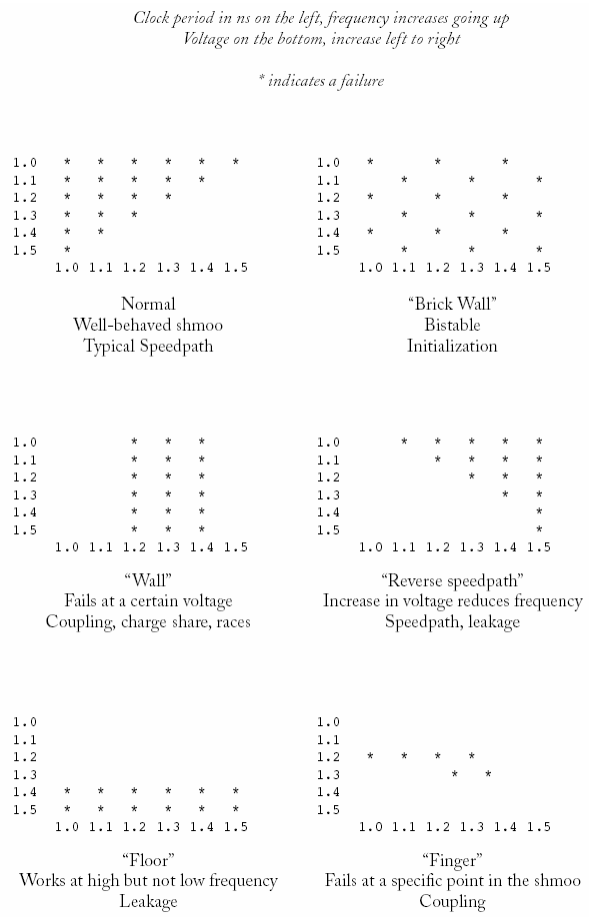
- Does the chip simulate correctly?
  - Usually done at HDL level
  - Verification engineers write test bench for HDL
    - Can't test all cases
    - Look for corner cases
    - Try to break logic design
- Ex: 32-bit adder
  - Test all combinations of corner cases as inputs:
    - 0, 1, 2, 2^{31}-1, -1, -2^{31}, a few random numbers
- Good tests require ingenuity

Silicon Debug

- Test the first chips back from fabrication
  - If you are lucky, they work the first time
  - If not…
- Logic bugs vs. electrical failures
  - Most chip failures are logic bugs from inadequate simulation
  - Some are electrical failures
    - Crosstalk
    - Dynamic nodes: leakage, charge sharing
    - Ratio failures
  - A few are tool or methodology failures (e.g. DRC)
- Fix the bugs and fabricate a corrected chip

Shmoo Plots

- How to diagnose failures?
  - Hard to access chips
    - Picoprobes
    - Electron beam
    - Laser voltage probing
    - Built-in self-test
- Shmoo plots
  - Vary voltage, frequency
  - Look for cause of electrical failures

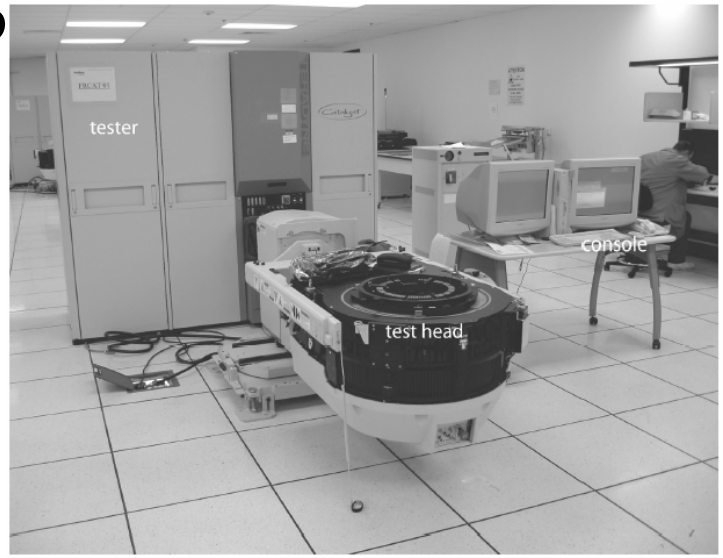


Need for Testing

- Physical defects are likely in manufacturing
  - Missing connections (opens)
  - Bridged connections (shorts)
  - Imperfect doping, processing steps
  - Packaging
- Yields are generally low
  - Yield = Fraction of good die per wafer
- Need to weed out bad die before assembly
- Need to test during operation
  - Electromagnetic interference, mechanical stress, electromigration, alpha particles

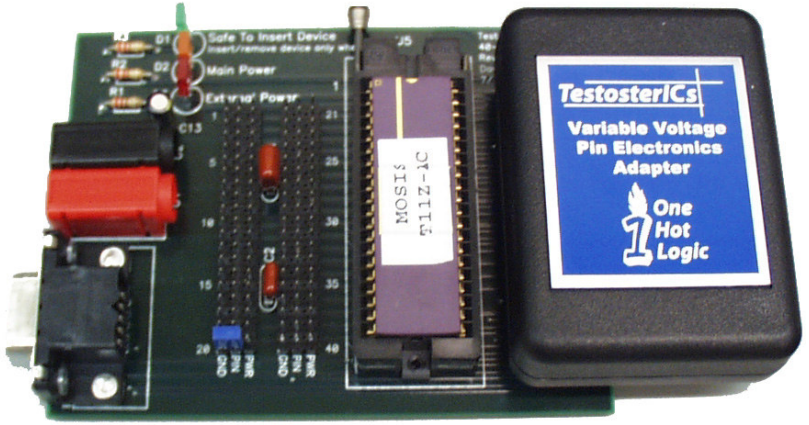
Manufacturing Test

- A speck of dust on a wafer is sufficient to kill chip
- *Yield* of any chip is < 100%
  - Must test chips after manufacturing before delivery to customers to only ship good p
- Manufacturing testers are very expensive
  - Minimize time on tester
  - Careful selection of *test vectors*



Testing Your Chips

- If you don't have a multimillion dollar tester:
  - Build a breadboard with LED's and switches
  - Hook up a logic analyzer and pattern generator
  - Or use a low-cost functional chip tester



Testing Levels and Test Costs

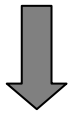
- Wafer
- Packaged chip
- Board
- System
- Field
- Concurrent checking

- Cost to detect a fault (per chip)
  - Wafer: $0.01-$0.1
  - Packaged chip: $0.1-$1
  - Board: $1-$10
  - System: $10-$100
  - Field: $100-1000

Manufacturing Testing

Goal: Detect manufacturing detects

Defects: layer-to-layer shorts
discontinuous wires
thin-oxide shorts to substrate or well



Faults: nodes shorted to power or ground (stuck-at)
nodes shorted to each other (bridging)
inputs floating, outputs disconnected (stuck-open)

Fault models

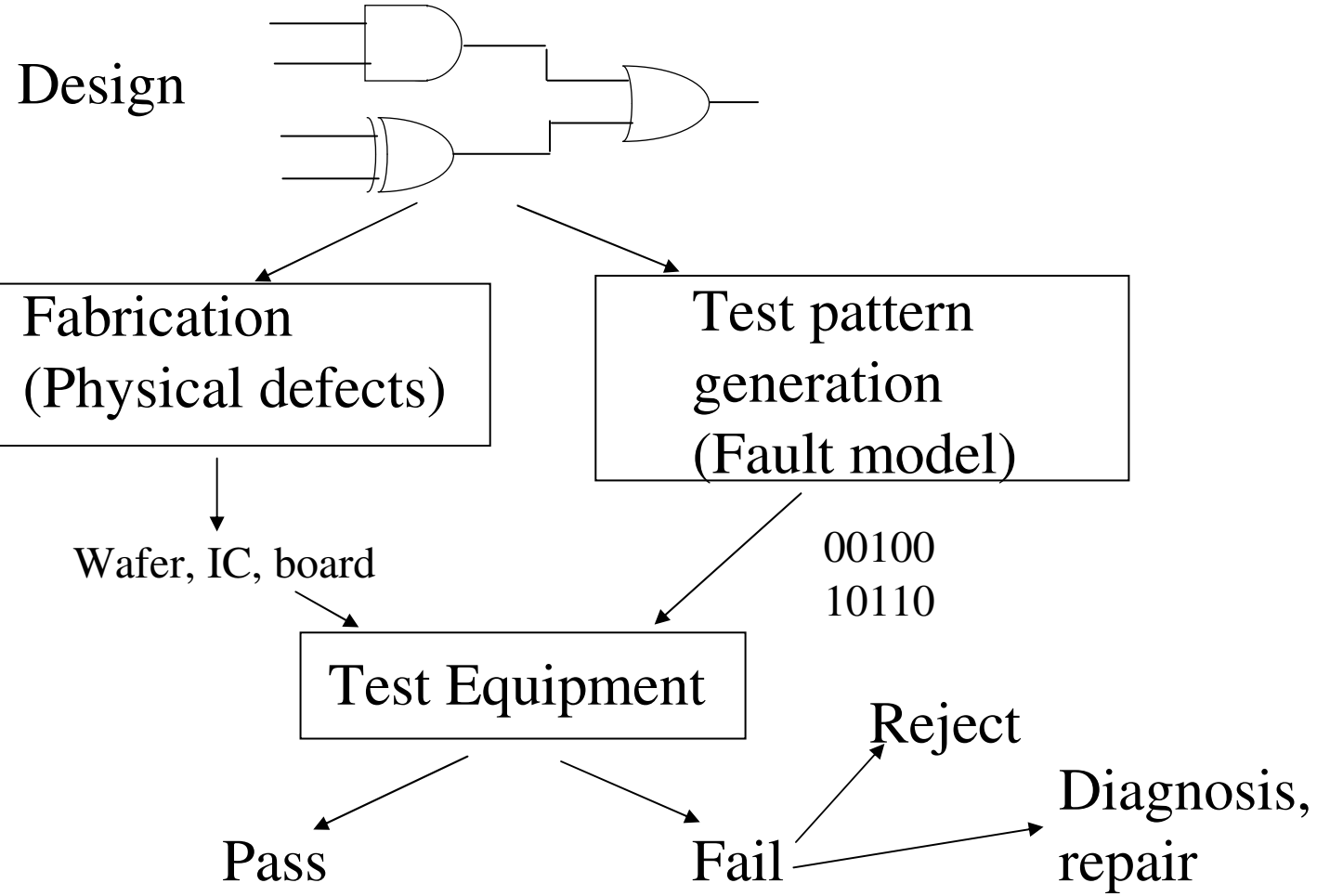
Testing : The Buzzwords

- Errors
  - Permanent
  - Intermittent
  - Transient
- Faults
  - Physical
  - Logical
- Test Evaluation
  - Fault coverage
  - Fault simulation
- Types of testing
  - Off-line, on-line
  - Self-test vs external test
  - DC (static) vc AC (at-speed)
  - Edge-pin, guided-probe, bed-of-nails, E-beam, in-circuit

Testing and Diagnosis

- Testing: Determine if the system (chip, board) is behaving correctly
- Diagnosis: Locate the cause of malfunctioning

Testing and Diagnosis



Testing: The Inevitable Acronyms

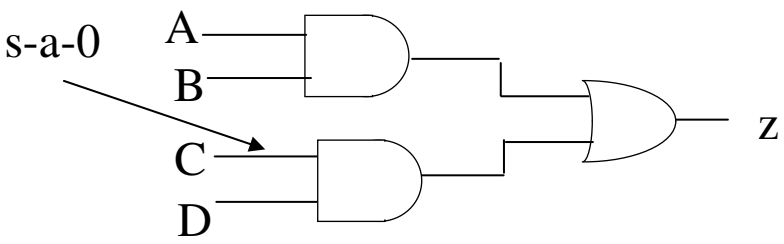
- System under test
  - UUT: Unit Under Test
  - CUT: Circuit Under Test
  - DUT: Device Under Test
- The tester
  - ATE: Automatic Test Equipment
- Test generation
  - ATPG: Automatic Test Pattern Generation
- Fault Models
  - SSL: Single Stuck-Line
  - MSL: Multiple Stuck-Line
  - BF: Bridging Fault
- DFT: Design for Testability
  - BIST: Built-in self-test
  - LFSR:Linear-Feedback Shift-Register

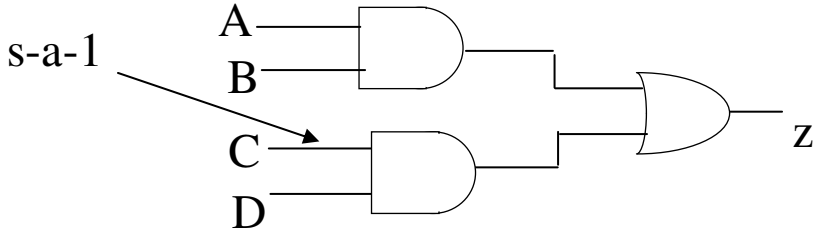
Fault Models

- Defects are too many and too difficult to explicitly enumerate
- Abstraction (technology independence): presence of physical defect is modeled by changing the logic function (or delay)
- Reduced complexity: distinct physical defects may be represented by the same logical fault
- Generality: tests derived for logical faults may detect vaguely-understood or hard-to-analyze physical defects
- A *test pattern* detects a fault from the fault model

Single Stuck-Line (SSL) Model

- A single node in the circuit is stuck-at 1 (s-a-1) or 0 (s-a-0).





Fault-free function z = AB+CD
Faulty function z^f = AB

Fault-free function z = AB+CD
Faulty function z^f = AB+D

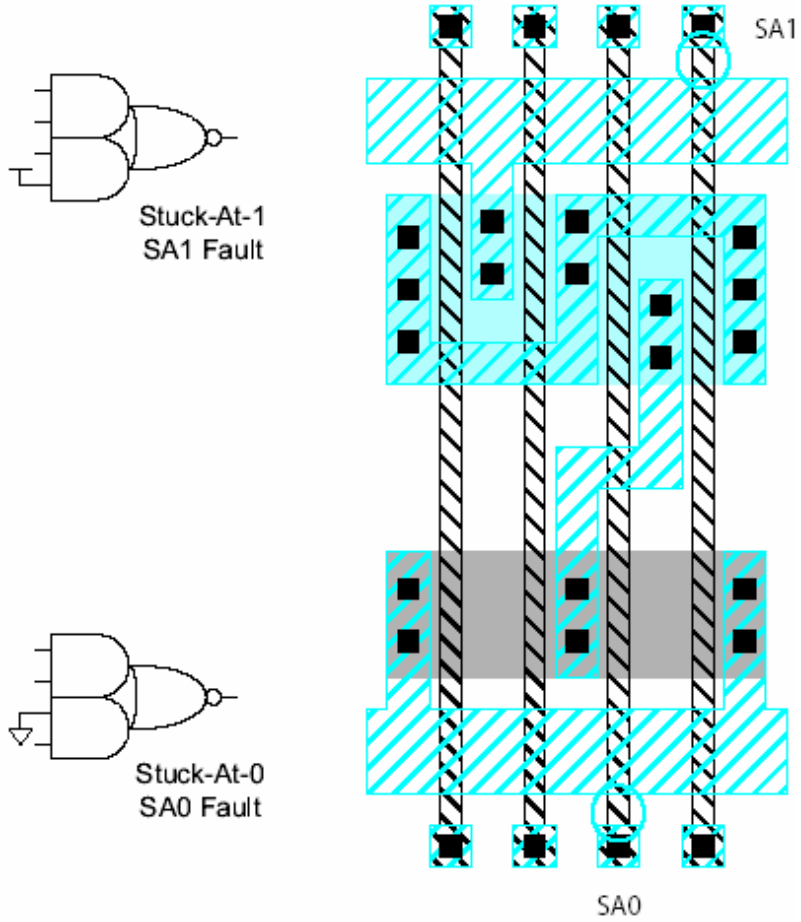
Number of possible stuck-at faults in a circuit with *n* lines?

Number of faults reduced by finding equivalent classes

Stuck-At Faults

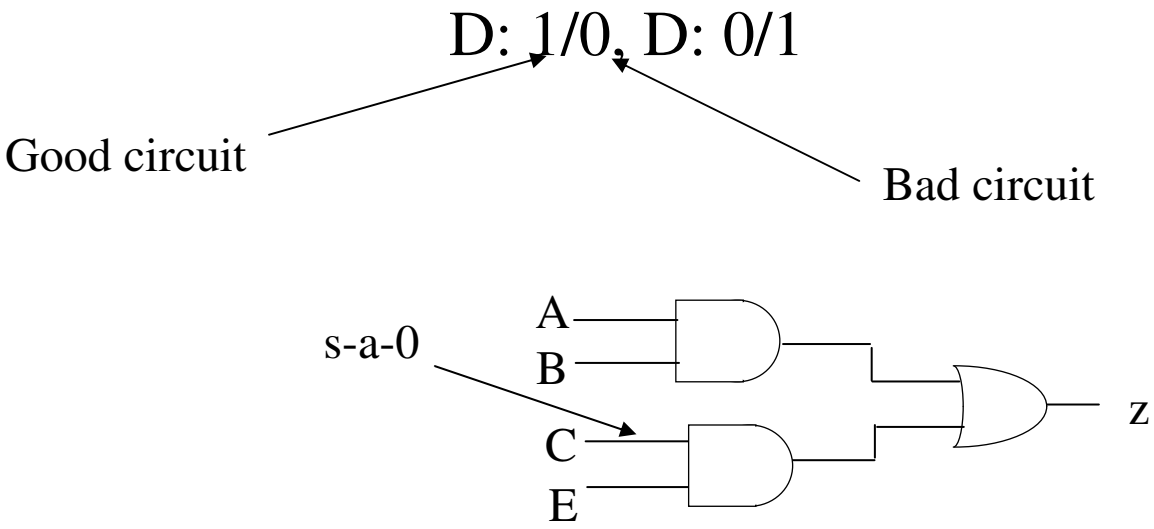
- How does a chip fail?
  - Usually failures are shorts between two conductors or opens in a conductor
  - This can cause very complicated behavior
- A simpler model: *Stuck-At*
  - Assume all failures cause nodes to be “stuck-at” 0 or 1, i.e. shorted to GND or V_{DD}
  - Not quite true, but works well in practice

Examples



SSL Fault Detection

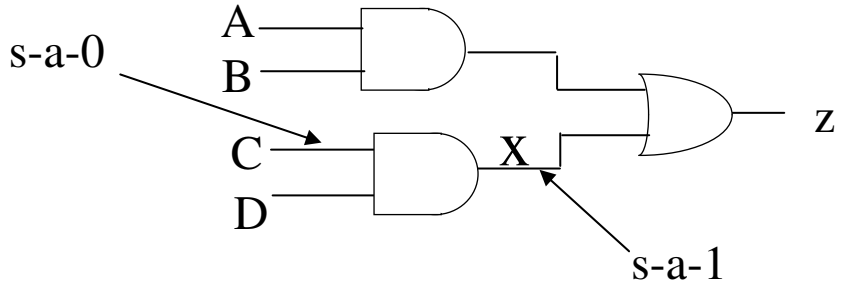
- A test pattern for fault *x* s-a-*d* is an input combination that 1) places $\overline{d}$ on x (activation), 2) propagates fault effect (D or $\overline{D}$) to primary output



ABCE = 0011 is a test pattern for C s-a-0

Multiple Stuck-Line (MSF) Faults

- More than one line may be stuck at a logic value



Fault: {C s-a-0, x s-a-1}

How many MSL fault can there be in a circuit with *n* nodes?

How to get test patterns for MSL faults?

Fault universe is too large, MSL fault model seldom used, especially since tests for SSL faults cover many MSL faults

Observability & Controllability

- *Observability*: ease of observing a node by watching external output pins of the chip
- *Controllability*: ease of forcing a node to 0 or 1 by driving input pins of the chip

- Combinational logic is usually easy to observe and control
- Finite state machines can be very difficult, requiring many cycles to enter desired state
  - Especially if state transition diagram is not known to the test engineer

Test Pattern Generation

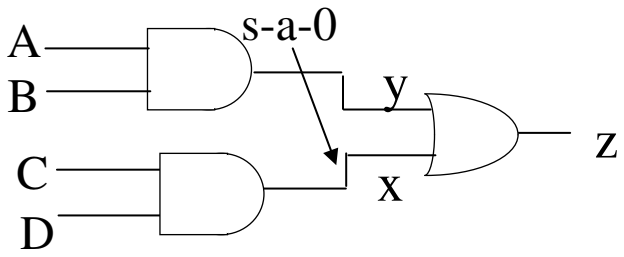
- Manufacturing test ideally would check every node in the circuit to prove it is not stuck.
- Apply the smallest sequence of test vectors necessary to prove each node is not stuck.

- Good observability and controllability reduces number of test vectors required for manufacturing test.
  - Reduces the cost of testing
  - Motivates design-for-test

Test Pattern Generation

- Exhaustive testing: Apply 2^n pattern to n-input circuit
- Not practical for large n
- Advantage: Fault-model independent

Fault-Oriented Test Generation Algorithm:



1) Set x to 1: activate fault
2) Justify D on x, propagate D to z

Set C and D to 1

Set y to 0

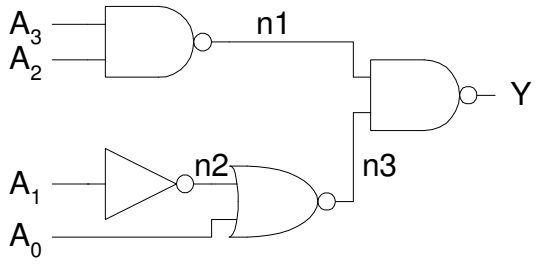
Set either A or B to 0

Example test pattern: ABCD = 0011

- Backtracking may be necessary
- Test generation is NP-complete

Test Example

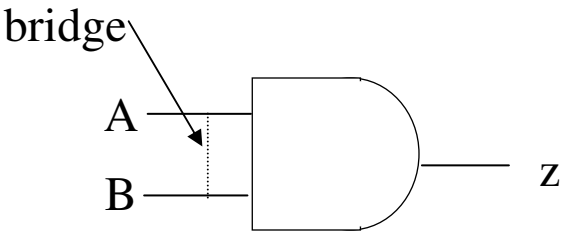
	SA1	SA0
• A_3	{0110}	{1110}
• A_2	{1010}	{1110}
• A_1	{0100}	{0110}
• A_0	{0110}	{0111}
• n1	{1110}	{0110}
• n2	{0110}	{0100}
• n3	{0101}	{0110}
• Y	{0110}	{1110}



- Minimum set: {0100, 0101, 0110, 0111, 1010, 1110}

Bridging Faults

- Models short circuits, pairs of nodes considered
- Number of bridging faults?
- Feedback vs non-feedback bridging faults

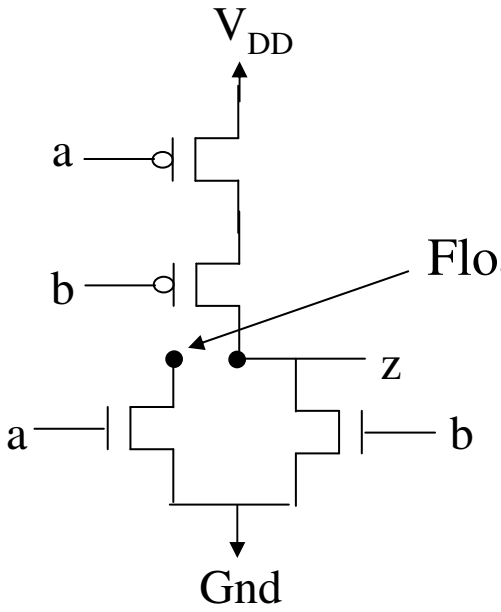


A	B	z	z^f	Wired-AND	Wired-OR
0	0	0	0	0	0
0	1	0	?	0	1
1	0	1	?	0	1
1	1	1	1	1	1

$z^f = ?$

What are the test patterns in this example?

Stuck-Open Faults



Fault-free circuit: $z = \overline{a+b}$

Faulty circuit: $z^f = \overline{a+b} + a\bar{b}\tilde{z}$

$\tilde{z}$: Previous value of z

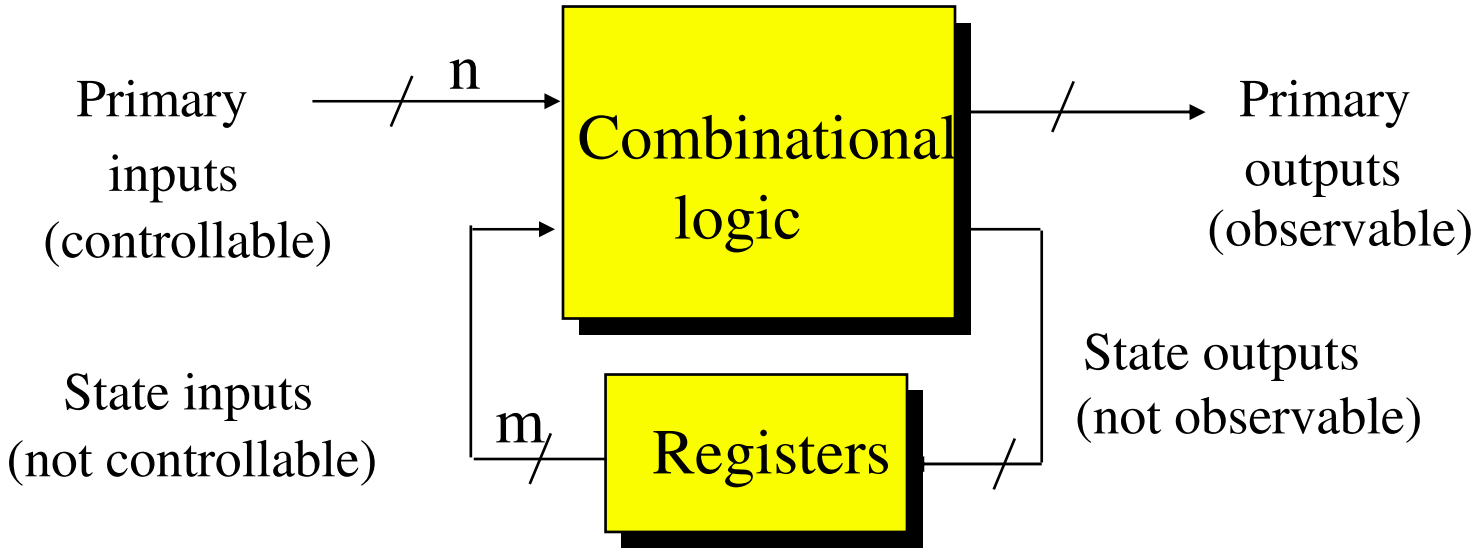
Case 1: a = b = 1, z pulled down to 0

Case 2: a = 1, b =0, z retains previous state

A test for a stuck-open fault requires two patterns

{ab = 00, ab = 10}

Sequential Circuit Test Generation

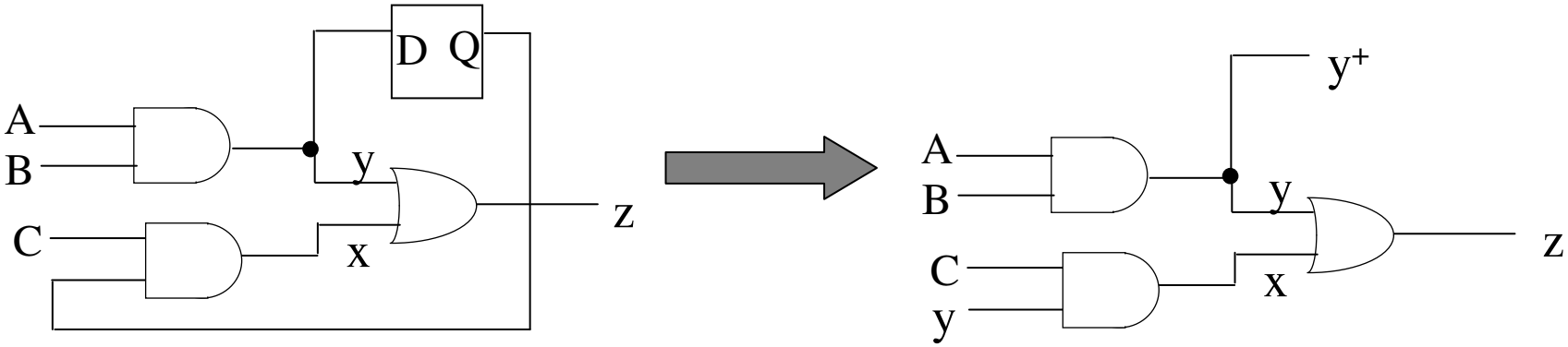


- Difficult problem!
- Exhaustive testing requires 2^{m+n} patterns (2^m states and 2^n transitions from each state)
- Every fault requires a sequence of patterns

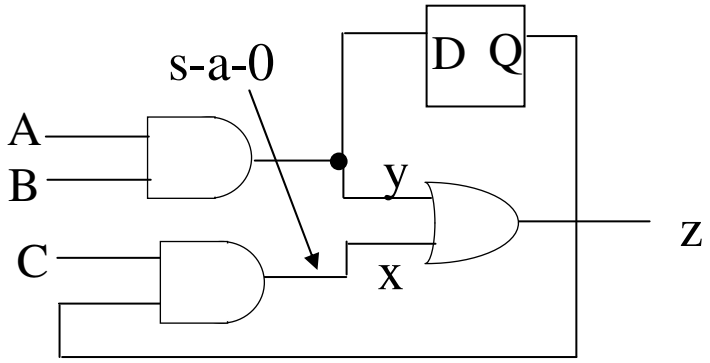
  Initializing sequence: drive to known state
  Test activation
  Propagation sequence: propagate discrepancy to observable output

Sequential Circuit Test Generation

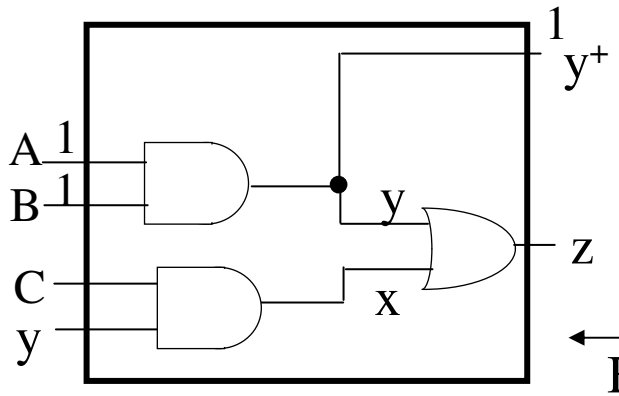
- Iterative-array model (pseudo-combinational circuit)



Sequential Circuit Test Generation

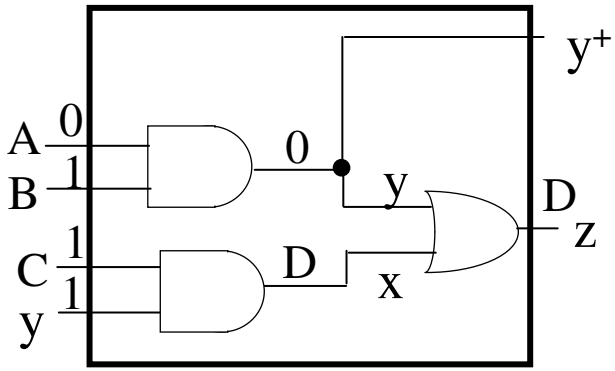


Assume initial state of flip-flop is not known



ABC = 11X

Backward traversal in time



ABC = 011

Current time frame

Test pattern sequence: {11X, 011}

Summary

- Think about testing from the beginning
  - Simulate as you go
  - Plan for test after fabrication
- "If you don't test it, it won't work! (Guaranteed)"