EECS 427
Lecture 15: Design for Test (DFT)
Reading: Insert H.3, H.4

Lecture Overview

- Intro to testing
- Designing for testability
  - Ad hoc
  - Scan
  - Self-test
- Single Stuck-at fault models
  - ATPG
  - 5 value logic

Testing is Expensive

- VLSI testers cost ~ $5M
- Volume manufacturing requires large number of testers, maintenance
- Tester time costs are in ¢/sec
- Test cost contributes 20-30% to total chip cost
- The alternative:
  - $1 to find a bad chip and toss it
  - $10 to find a bad IC on circuit board
  - $100 to find bad PC board in a system
  - $1000 to find a bad component in a field system
  - $1000000s to find a recurring bad part in a high-volume system (e.g., Intel floating point divide bug but this was a functional problem and not a manufacturing problem)

Validation and Test of Manufactured Circuits

Goals of Design-for-Test (DFT)

Make testing of manufactured part swift and comprehensive

DFT Mantra

Provide controllability and observability

Components of DFT strategy

- **Provide circuitry to enable test**
- **Provide test patterns that guarantee reasonable coverage**

Goals: Controllability&Observability

- Controllable: Can you easily create the conditions to demonstrate a certain failure?
- Observable: Can you quickly and easily observe the failure and distinguish it from other failures?

- Some tests reduce requirements on distinguishing between failures. Debugging yield problems needs to distinguish.

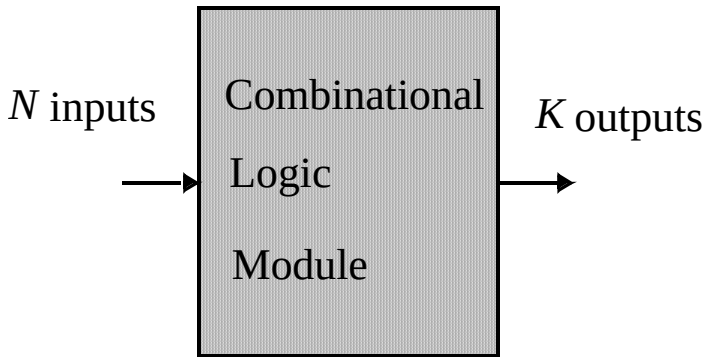
Test Classification

- Diagnostic test
  - Used in chip/board debugging, seeks to find location of faults
- “go/no go” or production test
  - Used in chip production
- Burn-in test
- Parametric test
  - Looks at continuous parameters, rather than discrete
  - Check parameters such as NM, T_{clk}
  - Frequency binning (Intel, AMD) fits here

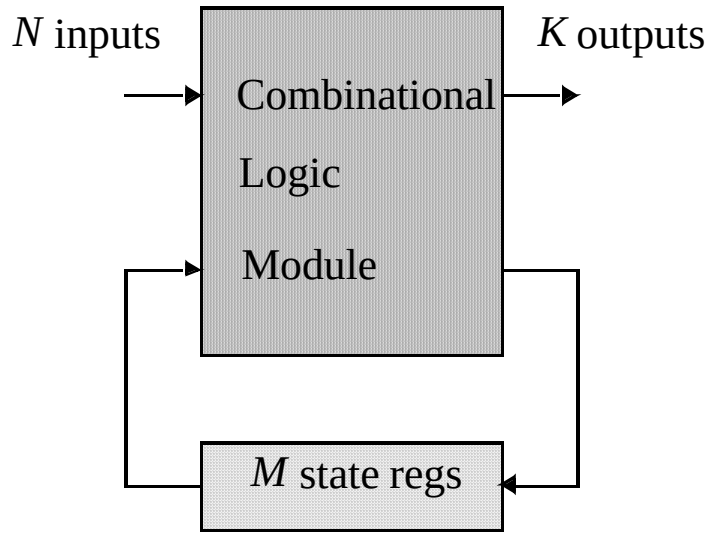
Burn-in or Stress test

- Subject chips to high temperature and increased Vdd while running production tests
- Aimed to catch:
  - ‘Infant mortality’ cases; chips that would have failed quickly after shipping due to major defects

Design for Testability



(a) Combinational function



(b) Sequential engine

2^N patterns **2^{N+M} patterns**

***2^N transtions** **Exhaustive test impossible or unpractical**

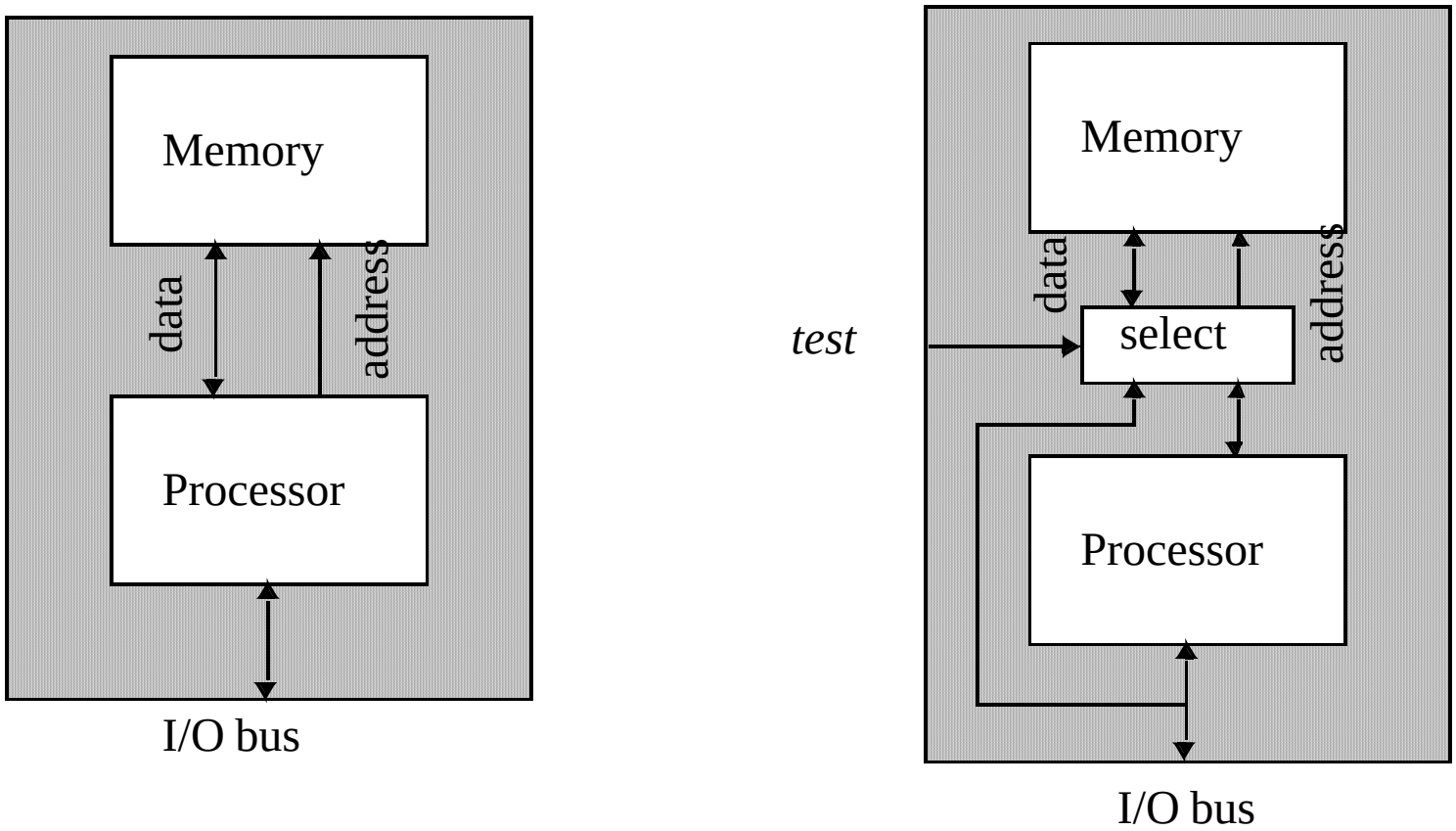
Test Approaches

- Ad-hoc testing
- Scan-based Test
- Self-Test

Problem is getting harder

  - Increasing complexity and heterogeneous combination of modules in system-on-a-chip
  - Larger designs with more inputs mean that less of the design space can be searched

Ad-hoc Test Example



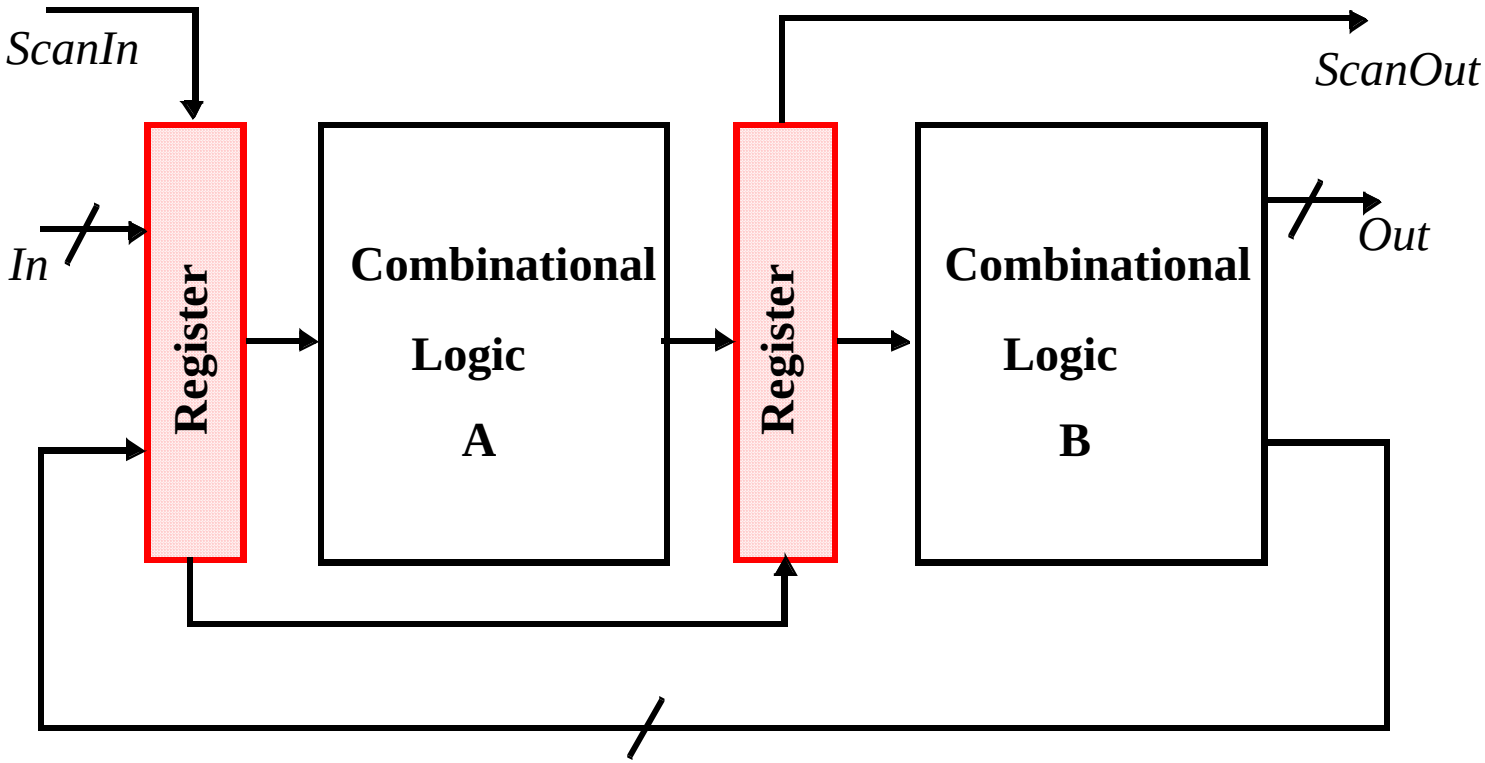
Inserting multiplexers improves testability at expense of additional hardware (and delay during normal operation)

Scan-based Test

Make all registers externally loadable and readable

When testing, register 1 reads from ScanIn. Then block A executes and Register 2 outputs to ScanOut for comparison

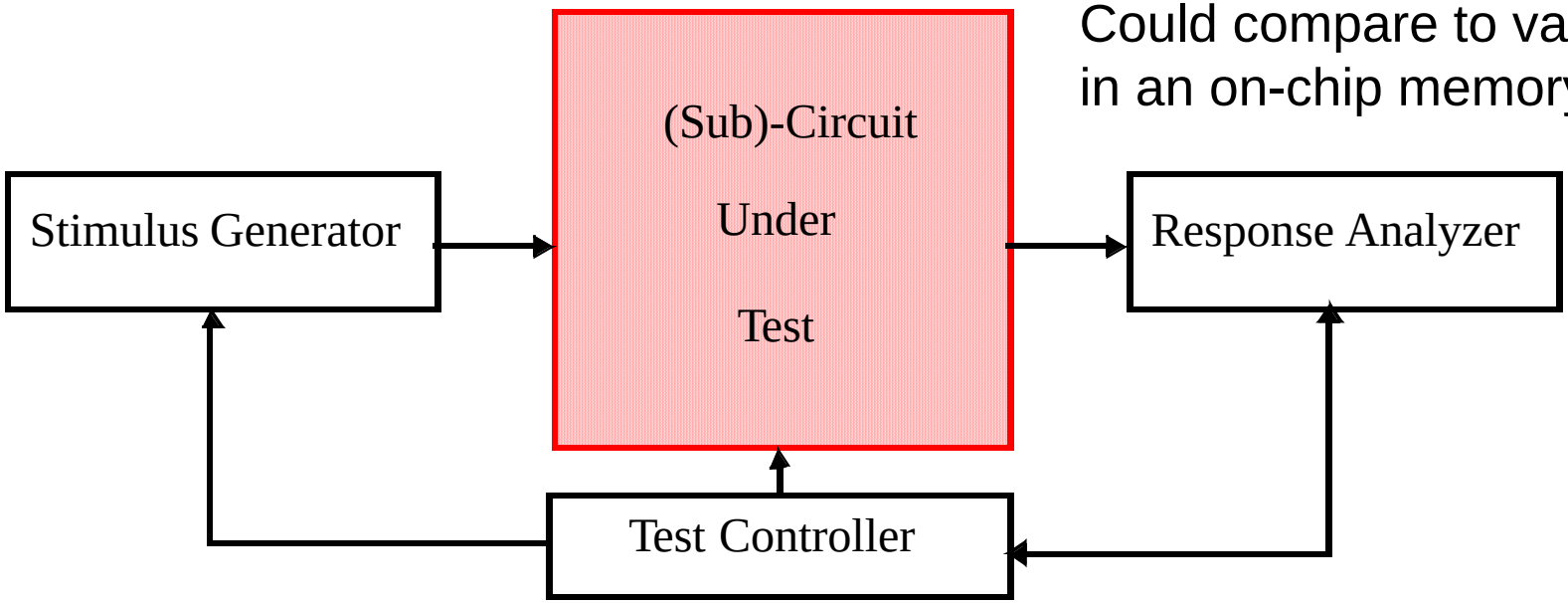
At the same time, Register 1 reads in next test vector



Scan-based Test

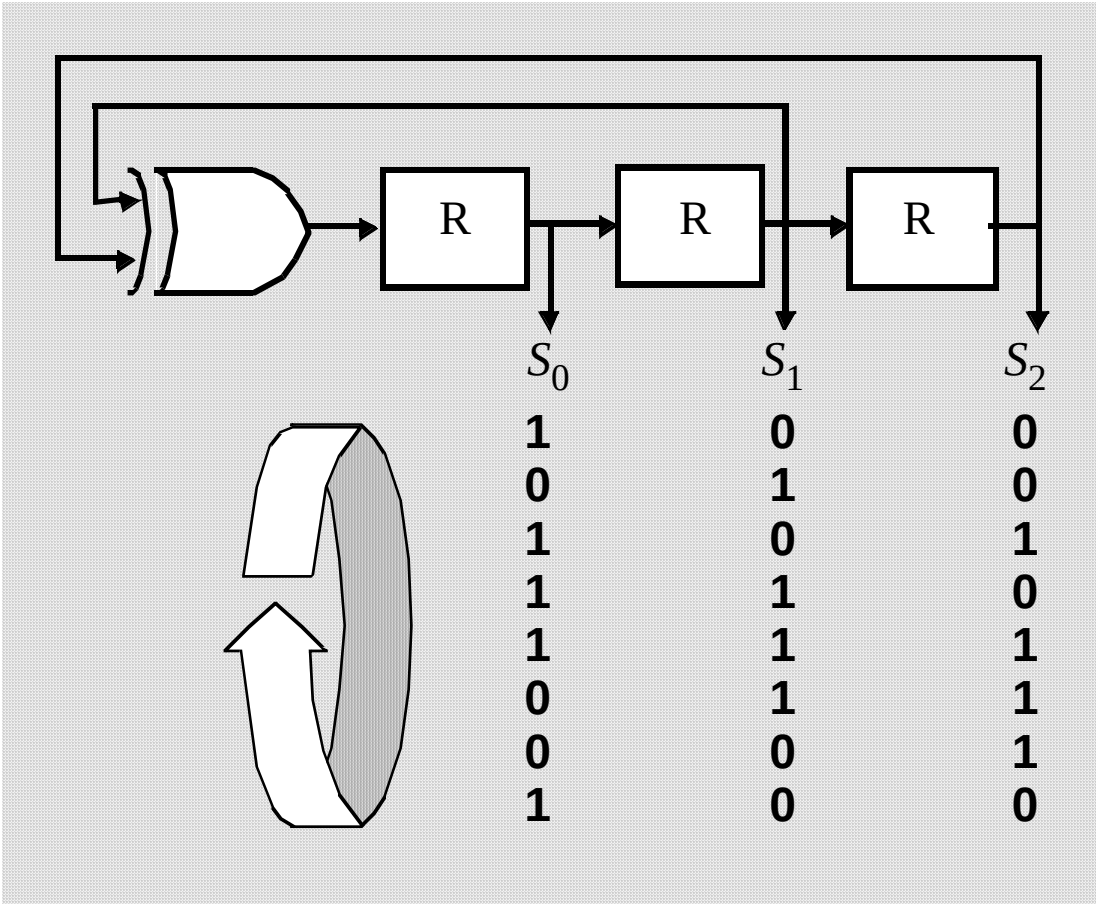
- N+1 clock cycles per test pattern
- 3 pins (scan in, scan out, scan enable)
  - Scan in/out may be shared with other pins
- Can split into multiple (k) chains
  - About (N/k) + 1 cycles per pattern
  - 2k+1 pins

Self-test



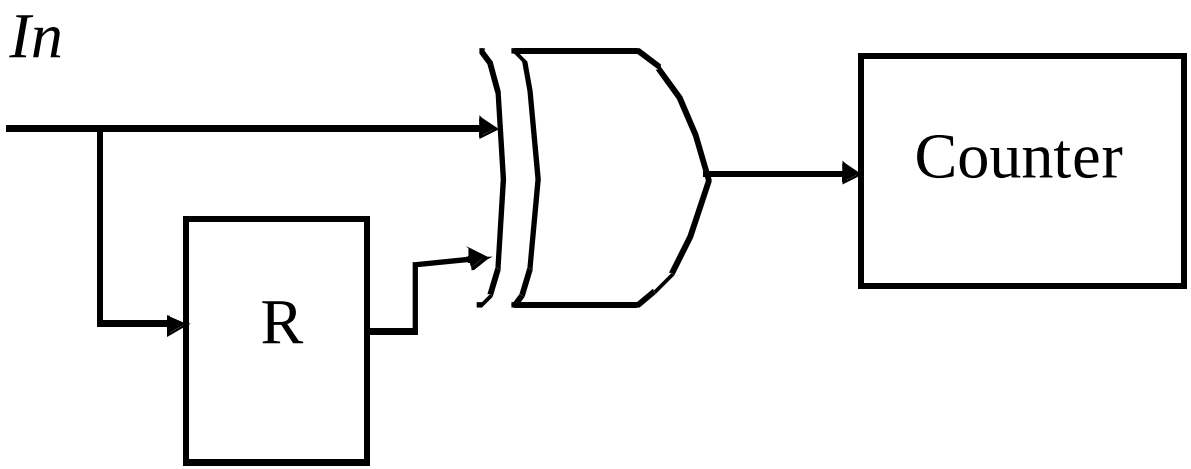
Rapidly becoming more important with increasing chip-complexity, larger modules, higher on-chip vs off-chip frequency. (esp go/no go)

Linear-Feedback Shift Register (LFSR)



Pseudo-Random Pattern Generator

Signature Analysis



Counts transitions on single-bit stream
≡ Compression in time

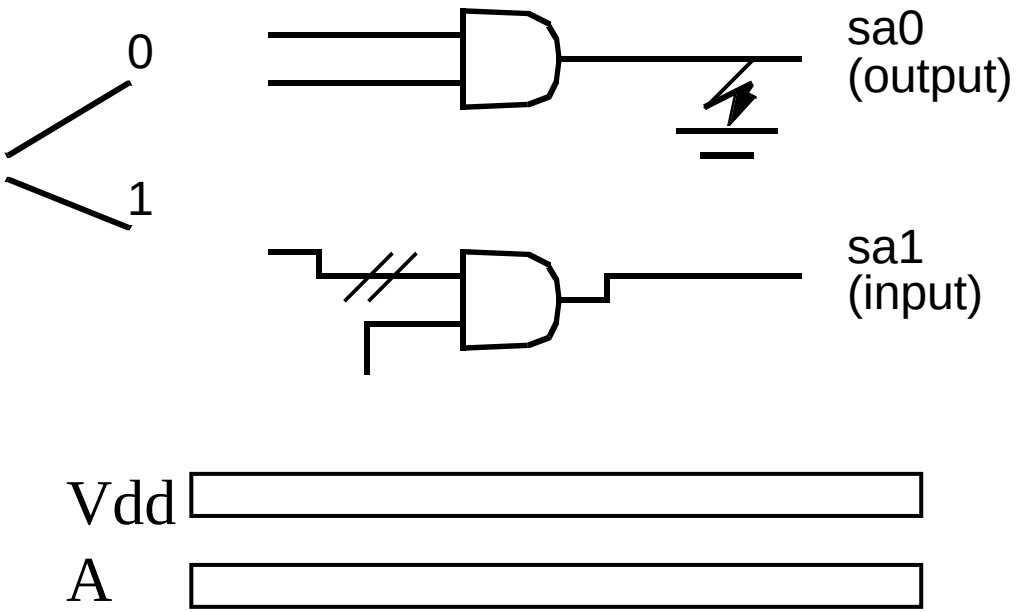
Sort of a parity check – does not guarantee correctness

Other Signature schemes

- Build signature into pattern generation : xor result with input, pseudo-random value after n iterations is signature
  - Small, can build into scan chain, less controllability
- Separate LFSR for signature, xor results into LFSR
  - Can use different test pattern generation for more controllability but bigger

Fault Models (H.4.1)

Most Popular – Single "Stuck-at" model



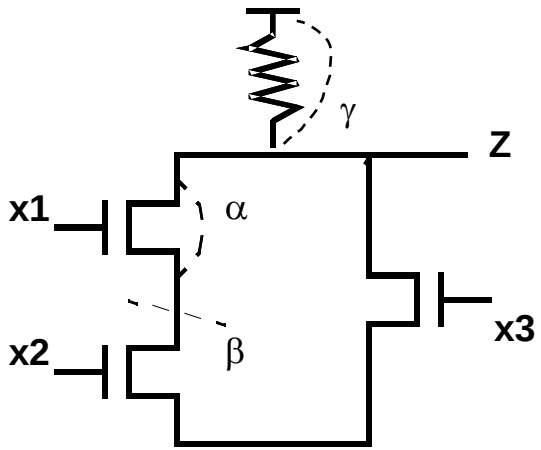
- Fault is permanent
- Effect of fault is that the faulty node is tied to either Vdd or ground
- Gate now functions improperly (which allows for observability)

Pros/Cons of Stuck-at Fault Model

- Advantages:
  - Reasonable # of faults: 2n where n is # of circuit nodes
  - Well-studied
  - ~90% of possible manufacturing defects are covered by this model
    - Source/drain shorts (see next slide), oxide pinholes, missing features, metallization shorts
- Disadvantages:
  - Does not cover all defects found in CMOS circuits

Stuck-at 0/1 Fault Models, cont.

Covers almost all (other) occurring faults, such as opens and shorts.

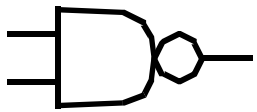


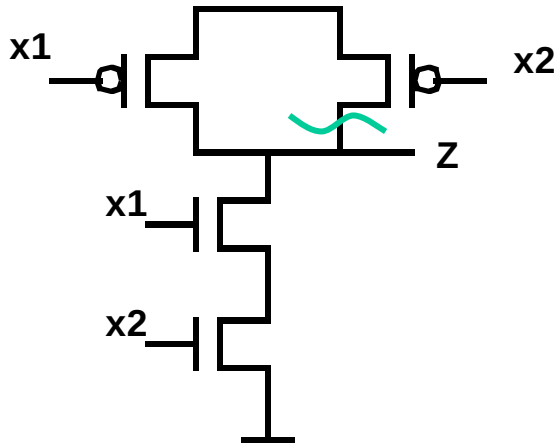
α, γ : x1 sa1

β : x1 sa0 or
x2 sa0

γ : Z sa1

Problem with stuck-at model: CMOS open fault



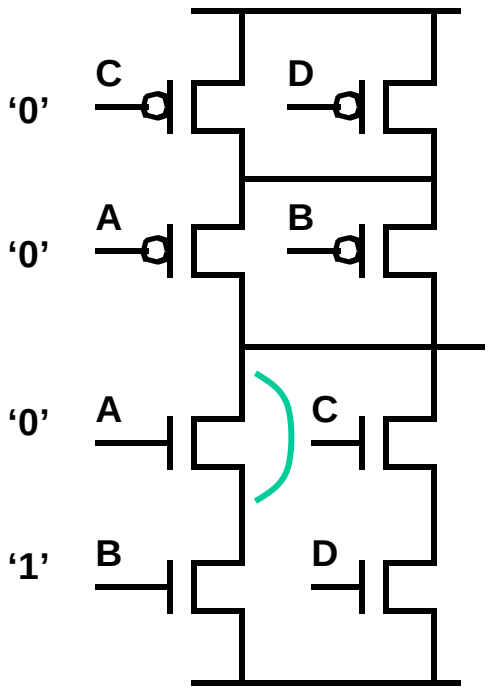


x1	x2	Z
o	x	1
1	1	0
1	0	Z_{n-1}

Sequential effect

Needs two vectors to ensure detection
Less controllable

Problem with stuck-at model: CMOS short fault



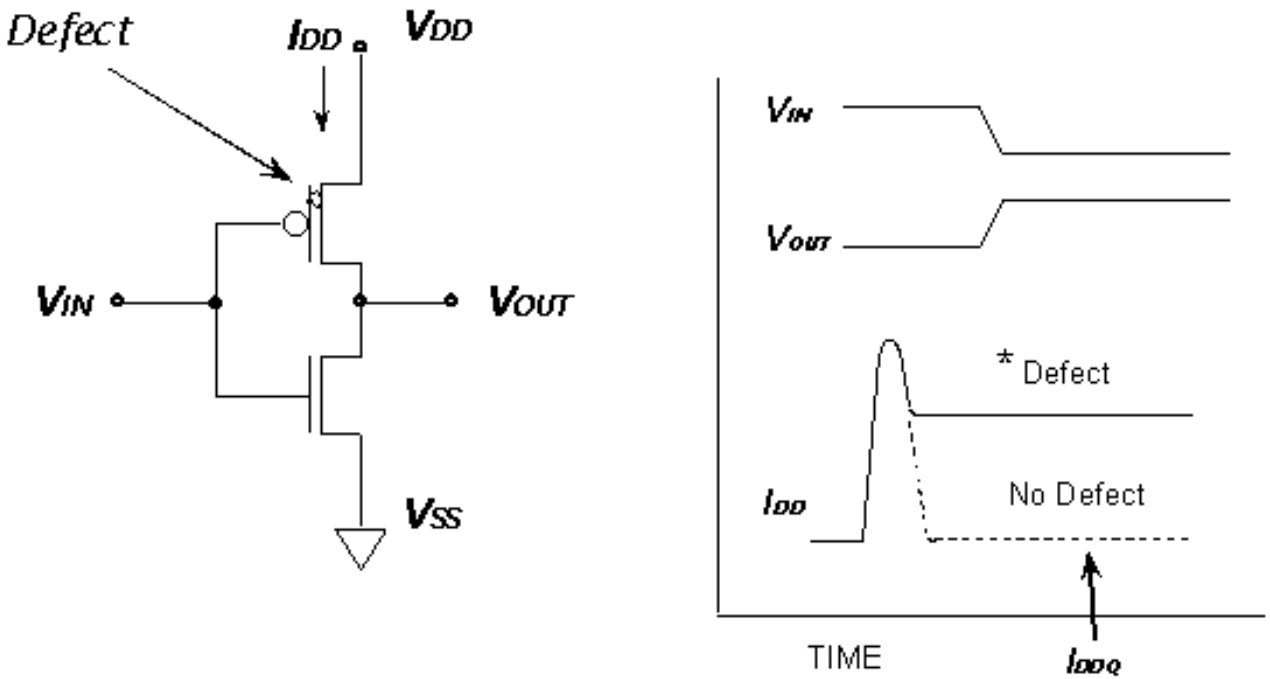
Causes short circuit between Vdd and GND for A=C=0, B=1

Possible approach:
Supply Current Measurement (IDDQ)

IDDQ testing

- Physical defects often lead to large currents flowing, even when the circuit is supposedly in a quiescent state
  - Normally a quiet CMOS circuit will have very little current draw
    - Note this is becoming much less valid with rising leakage currents today, jeopardizing IDDQ testing
- By measuring the quiescent current from the supply voltage, we can assume that a very large value means an error/fault/defect

Example, IDDQ testing



Gate-source short – when Vin is high, there is no current flow but when V_{in} goes low, current jumps

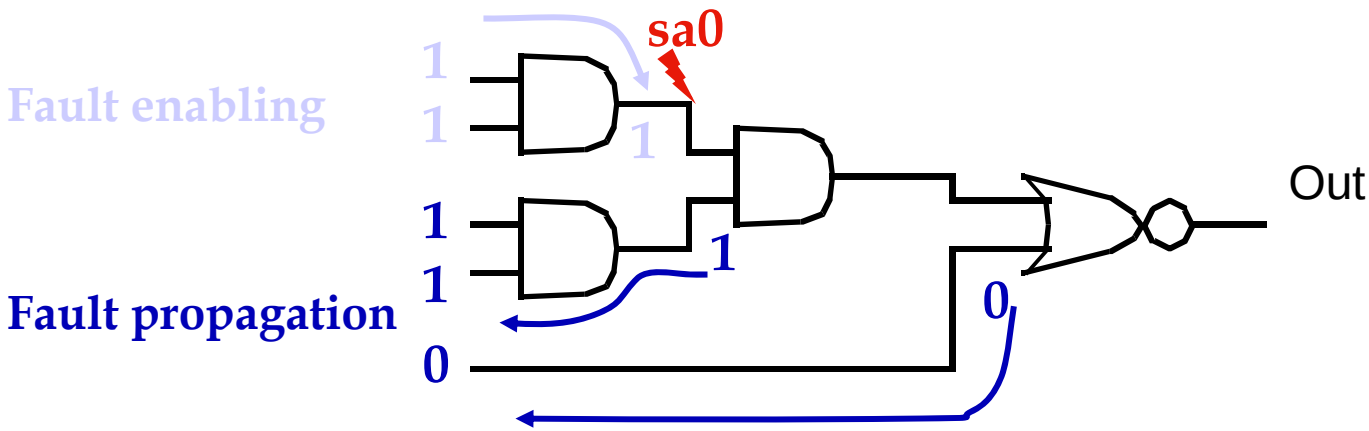
Gate still functions properly but you wouldn't want this to be shipped

Generating and Validating Test-Vectors

- Automatic test-pattern generation (ATPG)
  - for given fault, determine excitation vectors (called test vector) that will propagate error to primary (observable) output
  - majority of available tools: combinational networks only
  - sequential ATPG available from academic research
- Fault simulation
  - determines test coverage of proposed test-vector set
  - simulates correct network in parallel with faulty networks
- Both require adequate models of faults in CMOS integrated circuits

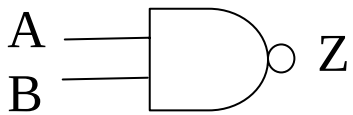
Path Sensitization

Goals: Determine input pattern that makes a fault controllable (triggers the fault, and makes its impact visible at the output nodes)



Techniques Used: D-algorithm, Podem

Stuck-At examples



Possible faults:

1. A sa 0
2. A sa 1
3. B sa 0
4. B sa 1
5. Z sa 0
6. Z sa 1

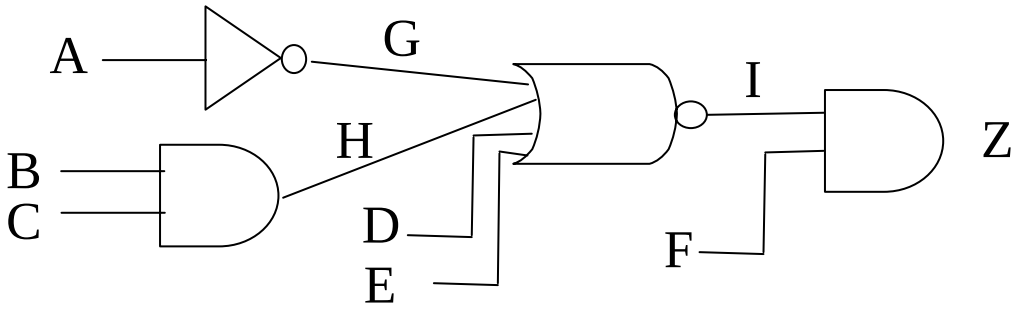
Tests to detect these

A	B	Z	Detects

5 value logic

- Possible values
  - 0, 1, X, D, D’
  - X = could be anything
  - D = should be a 1 but is a 0
  - D’ = should be a 0 but is a 1
- SA0 creates D’, SA1 creates D
- Boolean operators create new algebra (for example:)
  - D and 1 = D
  - D and D’ = 0
  - D’ nor 0 = D
- With this formalism (or others) we can design algorithms to choose test patterns that generate D or D’

ATPG Example



SA 0 fault on node I

Set I to D

Must make this fault observable – propagate to Z

(F=1)

Must make this fault controllable – backtrack to primary inputs

G,H,D,E = 0 : A= 1, B,C=1

ATPG difficulty

- Trees are easy, real combinational circuits are DAGS
- In general case even finding an input case that produces a 1 is NP complete (SAT)
- In practice most circuits are not so difficult and there has (and is) a lot of work possible in ATPG (just like in logic synthesis which is also a hard problem)

Summary

- Testing is an important part of designing integrated circuits
- Many engineers specialize in DFT techniques and are always in demand
- Fault models are abstractions of physical defects and are used to assess their impact on circuit behavior
  - Stuck-at 0/1 are most common
  - Test vectors can be created to determine whether a node is actually stuck at 0 or 1
- Key design for test techniques include:
  - Scan: load data into registers, run through logic, then scan out to compare to expected result
  - Self-test (or built-in self-test BIST): Incorporate everything on-chip which eases testing equipment requirements but requires lots of design effort