Testing of Digital Systems

Dr. Hao Zheng
Comp. Sci & Eng
U of South Florida

Why Testing?

- Digital System
  - Complex
  - At various levels of abstraction
  - Errors/faults can be introduced easily.
- Need to guarantee its correctness
  - Cost of having bad products in customers’ hands is really high.
- Cost to testing is increasing fast.

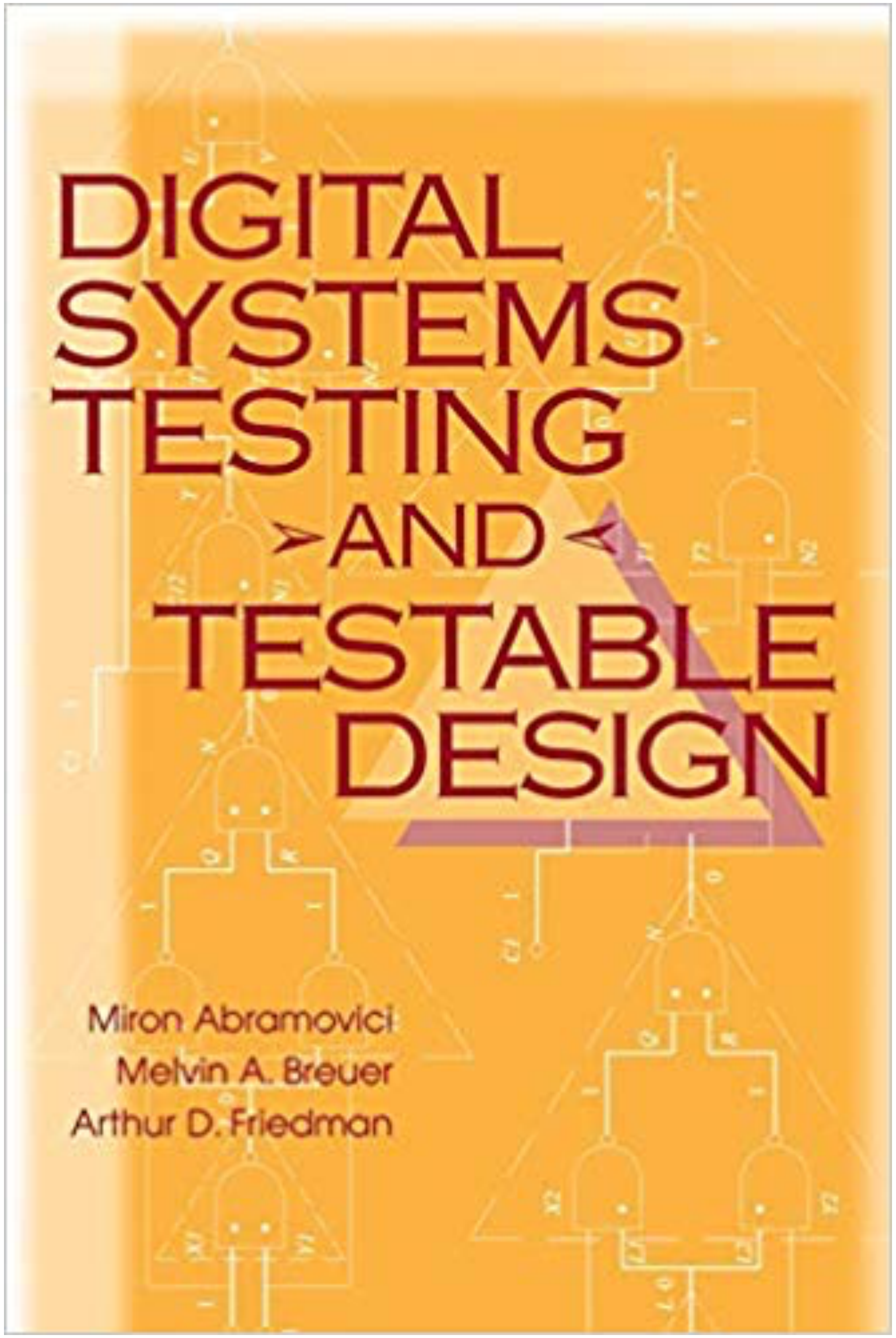
Digital System Testing

- How can we guarantee correctness of a digital system?
  - This course introduces basic concepts and principles for efficient testing.
  - Algorithms and design features for testing
- **Testing** -- Exercise a system and analyze its responses
  - Isolate good ones from faulty ones
- **Diagnosis** -- locate the faults and repair
  - Assume knowledge of internal structure

Prerequisites

- Catalog prerequisites
  - COP 4530 Data Structures
  - CDA 3201 Logic Design
- Highly desirable background
  - COT 4400 Analysis of Algorithms
  - CDA 4213 CMOS VLSI Design
  - Proficient in writing code in C/C++ or any other high level language.

Textbook


DIGITAL SYSTEMS TESTING AND TESTABLE DESIGN
Miron Abramovici
Melvin A. Breuer
Arthur D. Friedman

Evaluation

Homework/Exams	Weight	Date
Homework	20%	TBA
Exam 1	25%	Around 6th week
Exam 2	25%	Around 13th week
Final Project	30%	Starts after the Spring break

Final Grading Scale

$x < 60\%$	$60\% \leq x < 70\%$	$70\% \leq x < 80\%$	$80\% \leq x < 90\%$	$x \geq 90\%$
F	**D**	**C**	**B**	**A**

Communications

- Canvas
  - Assignments & grades
  - Announcements
- www.cse.usf.edu/~haozheng/teach/psv
  - Lecture slides, reading assignments,
  - Other relevant material

Topics (Tentative)

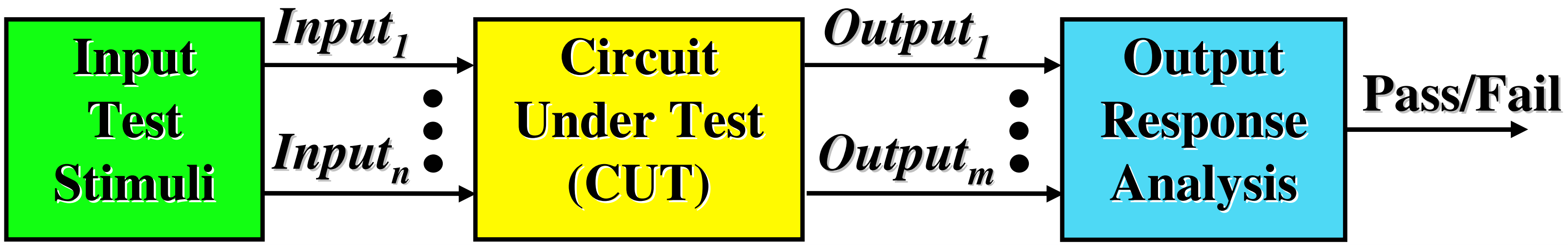
- Circuit modeling/simulation – chapter 2 & 3
- Fault modeling – chapter 4
- Fault simulation – chapter 5
- Testing for single stuck-at fault – chapter 6
- Testing for bridging fault – chapter 7
- Functional testing – chapter 8
- Design for testability – chapter 9
- Built-in self-test – chapter 11
- System-level diagnosis – chapter 15

Levels of Abstraction

- Based on granularities of information
  - System level
  - Processor level
  - Register transfer level
  - *Logic level – focus of this course*
  - Transistor-level
- This course focuses on issues in functional testing
  - Electrical characteristics are ignored.

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 Scenarios

- "Testing" is a general term used for widely different activities & environments
- Examples
  - One or more subsystems testing another by sending and receiving messages
  - A processor testing itself by executing a diagnostic program
  - Automatic Test Equipment (ATE) checking a circuit by applying and observing binary patterns

Errors and Faults

- **Observed error**: instance of incorrect operation
  - Important for testing

- Causes of observed errors
  - **Design Errors**: inconsistent implementation & spec
  - **Fabrication Errors**: due to human errors
  - **Fabrication Defects**: due to imperfect manufacturing
  - **Physical Failures**: due to system operation

Design Errors

- Errors due to incorrect design/understanding/implementation

- Examples
  - Incomplete and inconsistent specifications
  - Incorrect mappings between different levels of design
  - Violations of design rules

Fabrication Errors

- Due to mistakes made during fabrication/assembly sequence

- Examples
  - Wrong components
  - Incorrect wiring
  - Shorts caused by improper soldering

Fabrication Defects

- Errors due to **imperfect** manufacturing process

- Examples
  - Shorts or Opens
  - Improper doping profiles,
  - mask alignment errors

Physical Failures

- Errors occurring during system lifetime due to component wear-out and/or environmental factors

- Examples
  - Electro-migration
  - Temperature/Humidity/Vibration
  - Cosmic radiation and α-particles

Physical Faults

- Design Errors
- Fabrication Errors
- Fabrication Defects
- Physical Failures

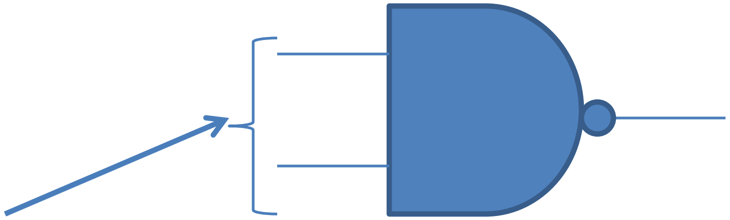
Physical Faults

Stability of Physical Faults:

- Permanent
- Intermittent
- Transient

Physical & Logical Faults

- Physical faults are difficult to handle mathematically.
- Physical faults are represented by the **logical fault**.
  - Example:



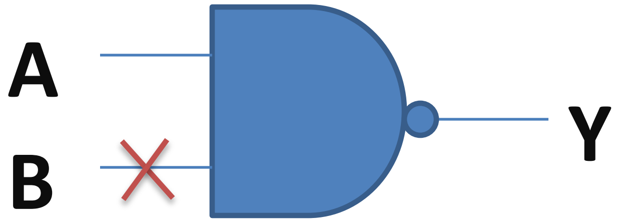
Physical fault
- Inputs are shorted

⇒

Logical fault
- Gate now behaves as an inverter

Fault Model

- If a fault occurs, then we need to *detect* it
- To detect a fault, we should be able to observe the error cause by the fault
- A fault model refers to natures of logical faults.
- Widely used model:
  - A single line **stuck** at a logic value



B Stuck-at-0 →

Modeling & Simulation

- Models are used for pre-fab testing.
- Model of a system
  - Digital computer representation of the system in terms of data structures and/or programs
- Logic Simulation
  - Model exercised by stimulating its inputs with logic values.
  - Evolution of signals over time in response to an applied input sequence.

Test Evaluation

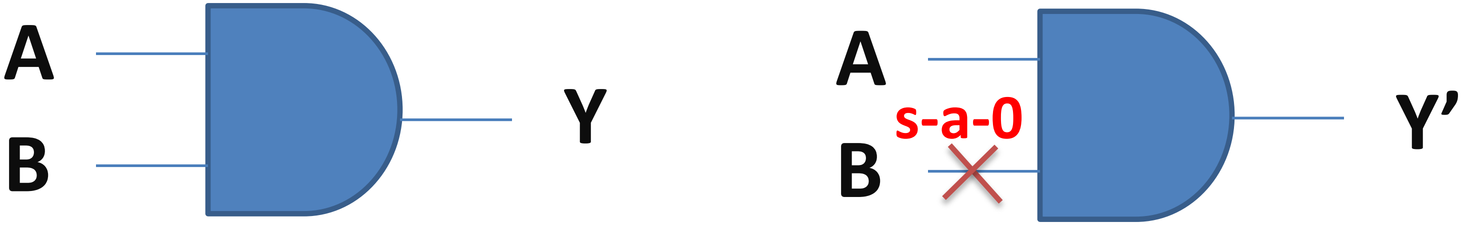
- Objective: determine the quality of a test.

$$\text{fault coverage} = \frac{\text{Number of faults detected}}{\text{Total number of faults}}$$

- Evaluated wrt a fault model`
- Performed via a simulated testing experiment known as **fault simulation.**

Test Evaluation

Question: Assume a fault B s-a-0 (stuck-at-0). Find a test vector (t) that detects this fault.

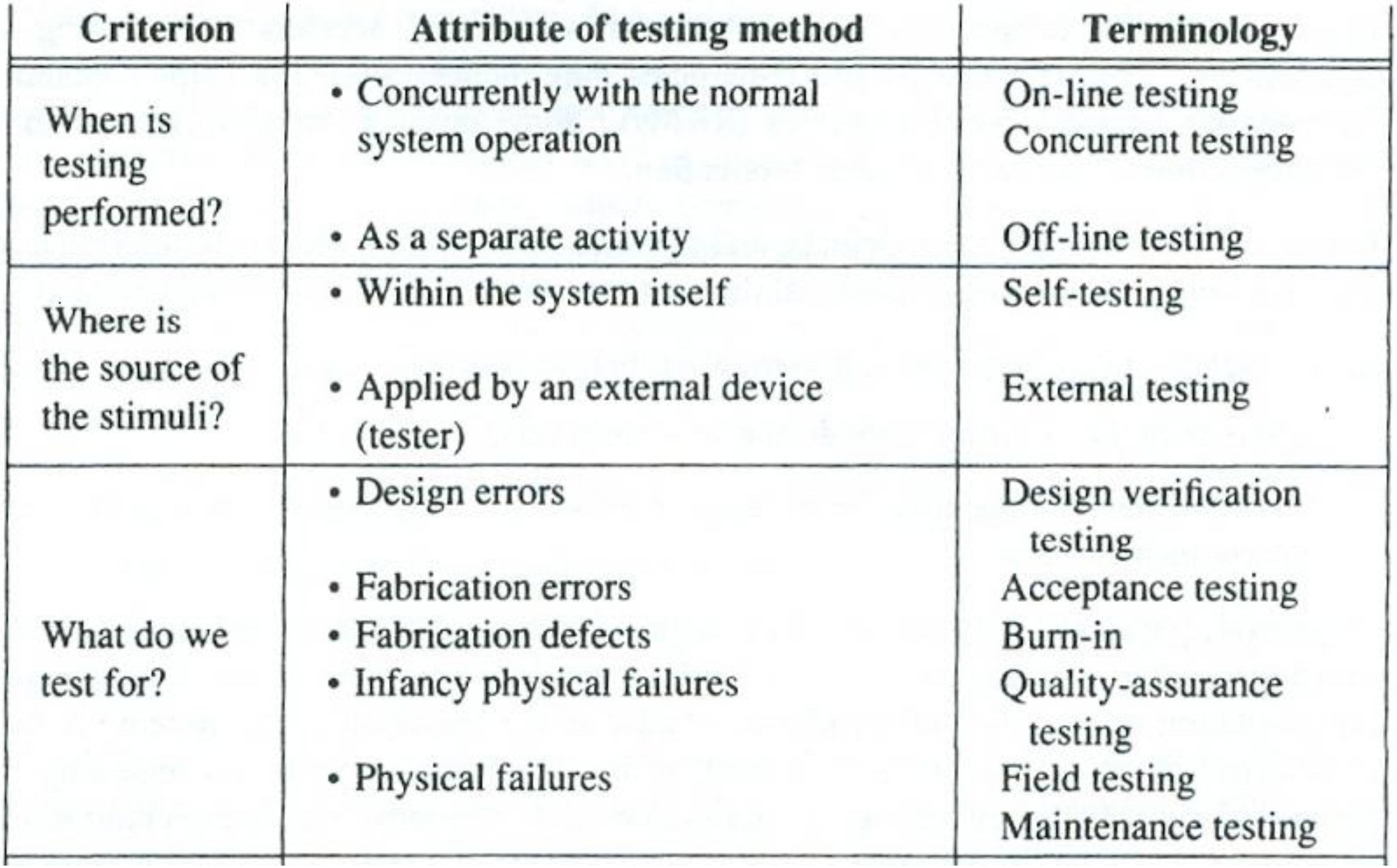


Without fault

With fault

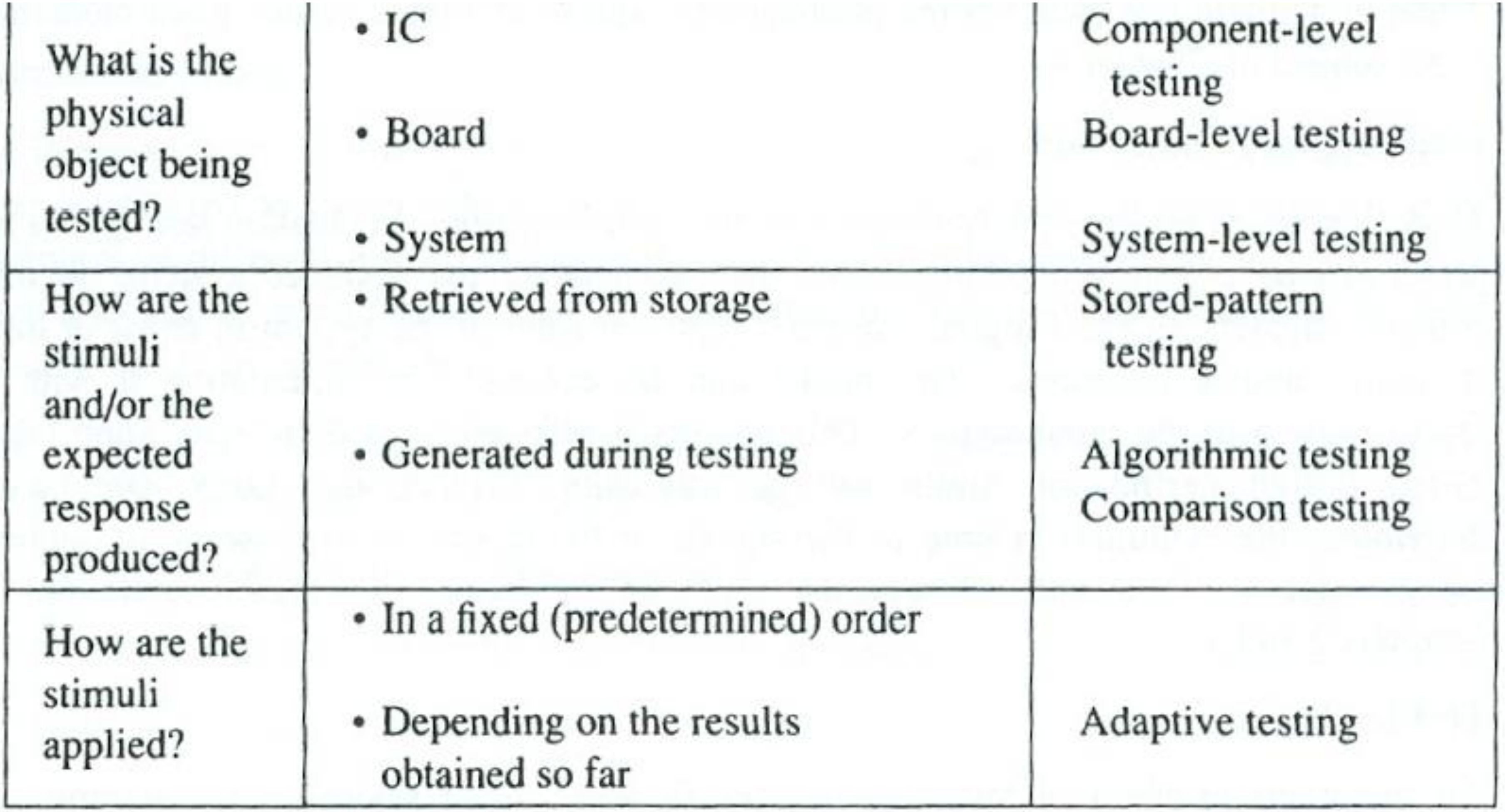
Hint: When you apply the test, the outputs should differ in the presence and absence of the fault

Types of Testing



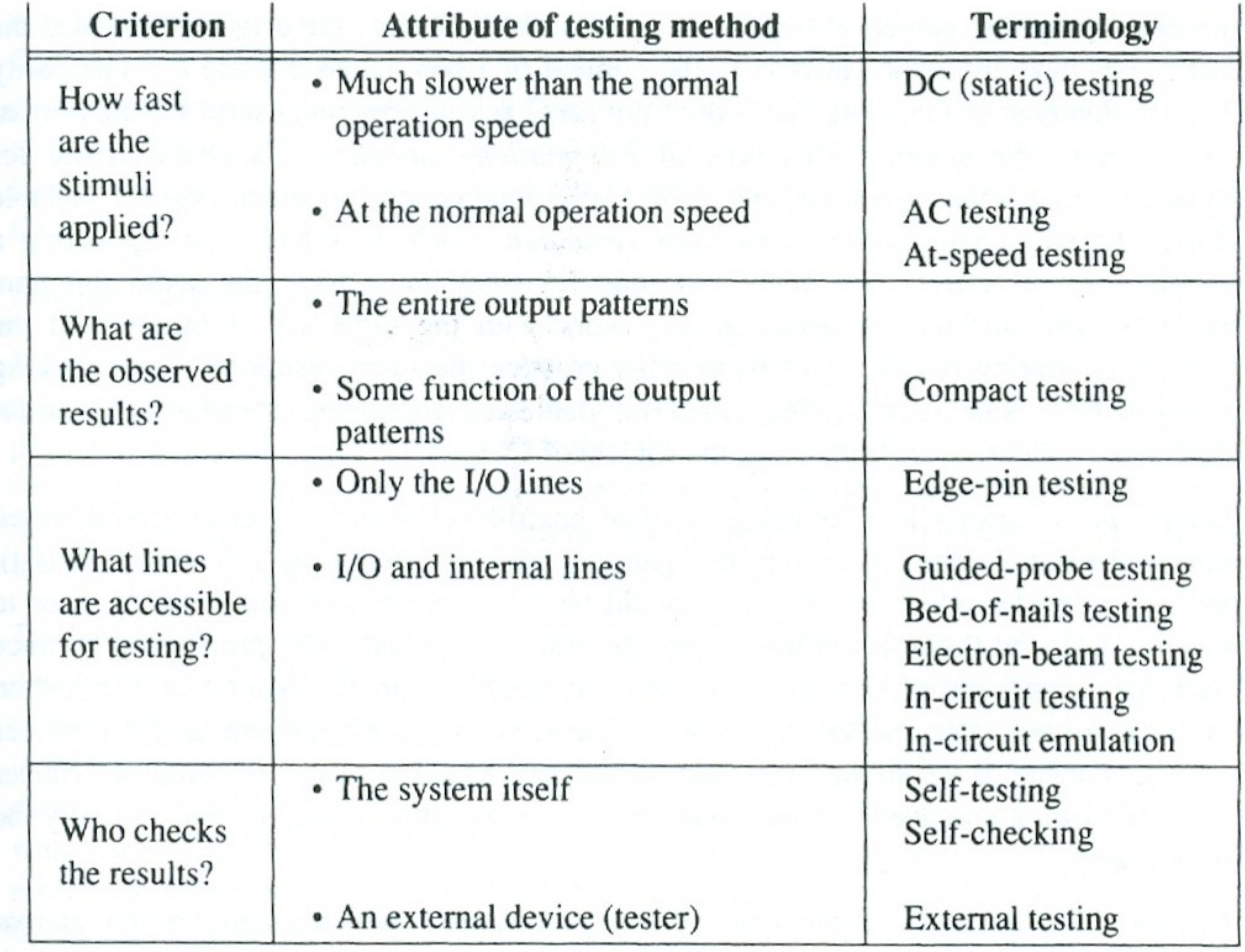
Criterion	Attribute of testing method	Terminology
When is testing performed?	• Concurrently with the normal system operation	On-line testing Concurrent testing
	• As a separate activity	Off-line testing
Where is the source of the stimuli?	• Within the system itself	Self-testing
	• Applied by an external device (tester)	External testing
What do we test for?	• Design errors	Design verification testing
	• Fabrication errors	Acceptance testing
	• Fabrication defects	Burn-in
	• Infancy physical failures	Quality-assurance testing
	• Physical failures	Field testing Maintenance testing

Types of Testing…contd.,



What is the physical object being tested?	• IC • Board • System	Component-level testing Board-level testing System-level testing
How are the stimuli and/or the expected response produced?	• Retrieved from storage • Generated during testing	Stored-pattern testing Algorithmic testing Comparison testing
How are the stimuli applied?	• In a fixed (predetermined) order • Depending on the results obtained so far	 Adaptive testing

Types of Testing…contd.,



Criterion	Attribute of testing method	Terminology
How fast are the stimuli applied?	• Much slower than the normal operation speed	DC (static) testing
	• At the normal operation speed	AC testing At-speed testing
What are the observed results?	• The entire output patterns	
	• Some function of the output patterns	Compact testing
What lines are accessible for testing?	• Only the I/O lines	Edge-pin testing
	• I/O and internal lines	Guided-probe testing Bed-of-nails testing Electron-beam testing In-circuit testing In-circuit emulation
Who checks the results?	• The system itself	Self-testing Self-checking
	• An external device (tester)	External testing

Testing Techniques

- Diagnostic Programs
- In-circuit Emulation
- On-line Testing
- Guided-probe Testing
- Misc Testing

Testing – Diagnostic Programs

- Stimuli – generated within system itself
  - By software or firmware
- Some parts of the system (*hardcore)* must be fault-free
- Can be adaptively applied
- Usually for field or maintenance testing

Testing – In-circuit Emulation

- Remove the need for a fault-free hardcore
- Used for μP-based systems
- μP removed from the board during testing
- Tester emulates the function of μP

On-line Testing

- Stimuli + Response are not known in advance
  - Stimuli are provided by patterns received during the normal operation mode
- Object of interest
  - Some property of the response which should be invariant throughout testing process

  Example 1: Fault-free decoder (only one o/p should be high at any time)

  Example 2: Word Parity
- Self-checking sub-circuits known as **checkers.**

Guided-Probe Testing

- Used for board-level testing
- If errors found, then tester decided which internal lines should be monitored
- Placed a probe on selected lines and re-tested
- Goal is to trace back to source of the error
- Sophisticated testers can monitor
  - A group of lines
  - All accessible internal lines (using a bed-of-nails fixture)

Other Testing Approaches

- **Algorithmic Testing** -- Generation of input patterns during testing; counters, feedback shift registers used to generate patterns
- **Comparison Testing** – expected responses generated from good known copy or by real-time emulation
- **Compact Testing** – Check some function *f(R)* instead of the response R itself.
  - Example: *f = no. of 1's in R*
  - Advantages: simplified and less memory intensive

Diagnosis and Repair

- After an error is observed, its cause is diagnosed.
- Once the cause is understood, the UUT may be repaired.
- Two diagnosis approaches for logical faults
  - Cause-effect analysis
  - Effect-cause analysis

Test Generation (TG)

- A process of determining the stimuli for testing a system.
- Different testing methods require different TG.
- Fault-oriented TG – targeted at certain fault models
  - Easier to automate
- Function-oriented TG – targeted at certain functions
  - May require substantial manual efforts.

Design For Testability

- Digital circuits come with additional circuitry to assist testing and diagnosis.
- Should be considered at early design stage to balance additional cost to the implementation and reduction in testing cost.
- More important as the complexity of modern circuits increases drastically.

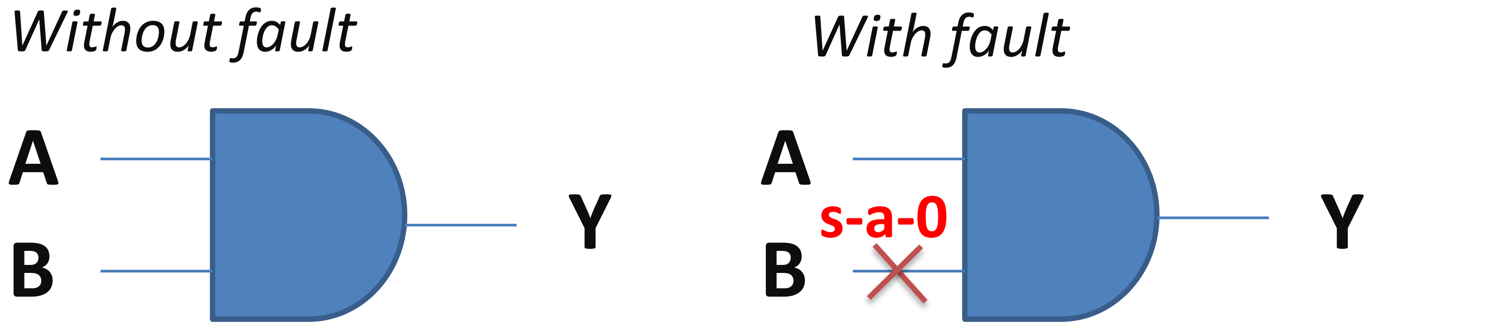
Backup

Main Topics (Tentative)

- Circuit modeling
- Fault modeling & simulation
- Testing for different types of faults
- Functional testing
- Design for testability
- Built-in self test
- System-level diagnosis
- Other contemporary testing issues

Test Vector

Question: Assume a fault B s-a-0 (stuck-at-0). Find a test vector (t) that detects this fault ***and compute its fault coverage.***



1) What is the size (N) of the fault universe ?

2) How many faults (M) can the test vector detect?

3) Fault Coverage = (M/N) =