Chapter 10
Testing and Design for Testability
Boonchuay Supmonchaid
Integrated Design Application Research (IDAR) Laboratory
September 22nd, 2004

Goals of This Chapter
- To provide a background on general testing
  - Fault Modeling
  - Test Generation
  - Faults Simulation
  - IDQ Testing
- To present general techniques on Design for Testability
  - Scan-Based
  - Boundary Scan
  - Build-In Self Test (BIST) Techniques

The Testing Problem
- The goal of testing is to detect manufacturing defects in digital ICs/Components as earlier as possible with minimum costs
  - Times to design and test
  - Silicon area to be used
- Costs increase dramatically as faulty components find their way into higher levels of integration.
  - $1 to fix an IC (so throw it out!)
  - $10 to find and replace bad IC on a PC board
  - $100 to find bad PC board in a system
  - $1000 to find bad component in fielded system

Test Classification
- Diagnostic test
  - Used in chip/board debugging
  - Defect localization
- “go/no go” or production test
  - Used in chip production
- Parametric test
  - $x \in [v,i] \text{ versus } x \in [0,1]$
  - Check parameters such as $NM, V, t, T$
Testing Procedure

- Apply a set of test vectors to each device off the manufacturing line and compare outputs to the known good response.
- The optimum test set will detect the greatest number of defects that can be present in a device with the least number of vectors.
- Development of the optimal test set is the most difficult part of the process.
  - A longer test set takes more time to apply to each device ($$$).
- There are a number of different approaches to test set generation.

Types of Test

- Exhaustive Test
  - Applies every possible input vector
  - Guaranteed to detect all detectable faults
  - Impractical in terms of number of tests required

- Functional Test
  - Tests every function of the device
  - Also detects every detectable faults if completed
  - Provides a smaller test set size than exhaustive tests but requires more time for the designers to find optimum test vector

Example: Which Type of Test?

- Consider a 74181 ALU chip - 14 inputs

<table>
<thead>
<tr>
<th>Type of test</th>
<th>Exhaustive</th>
<th>Functional</th>
<th>Modeled fault</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fault coverage</td>
<td>100 %</td>
<td>100 %</td>
<td>100 % (modeled faults)</td>
</tr>
<tr>
<td>Number of test vectors</td>
<td>16384</td>
<td>448</td>
<td>47</td>
</tr>
<tr>
<td>Notes</td>
<td>At 10 MHz, a 16-bit ALU with 38 inputs would take 7.84 hours to test.</td>
<td>There is no algorithmic way to verify that all functional modes have been covered.</td>
<td>Fault model determines the quality (coverage).</td>
</tr>
</tbody>
</table>
Current Testing Practice

- For most ASIC designs, the typical practice is to begin with the designer’s functional test set.
- This set of vectors is fault simulated to determine its fault coverage.
  - If the coverage is too low to be acceptable, more functional vectors can be added to exercise the portions of the circuit where the undetected faults lie.
  - A more efficient approach is to feed the list into an Automatic Test Pattern Generation (ATPG) program to develop the test.
- Another approach used is to apply Design for Test (DFT) or Build-In Self-Test (BIST) Techniques as part of the design process.
  - IBM’s Level Sensitive Scan Design

Fault Attributes

- **Cause**
  - Specification Mistakes
  - Implementation Mistakes
- **Duration**
  - Permanent
  - Transient
- **Value**
  - Determinate
  - Indeterminate
- **Nature**
  - Hardware
  - Software
- **Extent**
  - Local
  - Global

Fault Modeling

- The trade-off in fault modeling is to develop a model that is as simple as possible to use in test generation while detecting as high as possible the percentage of physical defects.
- Most common fault models
  - **Stuck-at faults**
    - Single stuck-at (s-a) faults
    - Multiple stuck-at (m-a) faults
  - **Stuck-open faults**
  - **Bridging faults**
  - **Delay faults**

Single Stuck-at Fault Model

- **Assumptions**
  - Only one line in the circuit is faulty at a time
  - The fault is permanent (as opposed to transient)
  - The effect of fault is as if the faulty node is tied to either $V_{CC}$ (s-a-1), or $GND$ (s-a-0) (short circuits!)
  - The function of the gates in the circuit is unaffected by the fault

<table>
<thead>
<tr>
<th>Fault-Free Gate</th>
<th>Fault: s-a-1</th>
<th>Faulty Gate</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Most often used because of its simplicity during test generation and fault simulation
**Single Stuck-at Fault Model II**

- **Advantage**
  - Can be applied at the logic level or module level
  - Reasonable number of faults $2n$ ($n$ = # of circuit nodes)
  - Algorithms for ATPG and fault simulation are well developed and efficient.
  - Single stuck-at fault model can cover about 90% of the possible manufacturing defects in CMOS
    - Source-drain shorts, oxide pinholes, missing features, diffusion contaminants, metallization shorts, etc.
  - Other useful fault models (stuck-open, bridging faults) can be mapped into (sequences of) stuck-at faults

- **Disadvantage**
  - Does not cover all defects in CMOS or other devices

**Stuck-Open Fault Model**

- **Advantages**
  - Covers physical defects not covered by single or multiple stuck-at fault models
  - Can be tested with sequences of stuck-at fault tests
    - In the NOR gate example, apply $AB = 00$ (test for F s-a-0) forcing F to VCC
    - Then apply $AB = 10$ (test for A s-a-0) to force F to GND and observe results.

- **Disadvantages**
  - Requires a larger number of tests (sequence for each fault)
  - Algorithms for ATPG and fault simulation are more complex and less well developed
  - Requires a lower level circuit description, at least for development of the fault list

**Stuck-Open Fault Model**

- **Assumptions**
  - A single physical line in the circuit is broken
  - The resulting unconnected node is not tied to either $V_{CC}$ or GND.

  - **Line Break results in a "memory effect" in the behavior of the circuit**
    - With $AB = 10$, there is no path from either $V_{CC}$ or GND to the output
      - F retains the previous value for some undetermined discharge time.

**Test Generation Process**
Automatic Test Pattern Generation

- The objective is to automatically generate a test for faults in the circuit-under-test
  - Deterministic (or heuristic) - require expertise
  - Random - or more appropriately Pseudorandom
- Major classes of methods
  - Pseudorandom
  - Ad-Hoc
  - Algorithmic
    - D-algorithm
    - PODEM
    - FAN and related algorithms
    - Others …

Pseudorandom Test Generation

- Simply generate an input vector using a pseudorandom number generator and perform fault simulation to determine if it detects the target fault.
- The characteristics of the fault greatly influence how well pseudorandom test generation will work
  - Easy-to-detect faults
  - Hard-to-detect faults
- Typically used in the beginning of the test generation process to remove easy-to-detect faults from the fault list.

Pseudorandom Test Generation II

Typically pseudorandom tests are generated and fault simulated until two or more successive pseudorandom vectors fail to detect any new faults. Then deterministic ATPG processes are used to target the remaining undetected faults.

Ad-Hoc Test Generation

- Uses functional test vectors developed by designers for functional verification and design debugging by
  - Fault simulating to determine fault coverage
  - Determining locations of undetected faults
  - Adding additional functional tests to exercise areas of design with undetected faults
  - Re-fault simulating and repeating until desired fault coverage is achieved
- No special test generation system is required, only fault simulator - use existing vectors and designer expertise.
- Achieving high fault coverage may be difficult and time consuming - especially for synthesized designs
**Fault Table**

To construct fault table of XOR
Insert a test vector, record the response of the circuit, and compare with the known good result $(f_0)$

<table>
<thead>
<tr>
<th>Test Vector</th>
<th>$T_1$</th>
<th>$T_2$</th>
<th>$T_3$</th>
<th>$T_4$</th>
<th>$T_5$</th>
<th>$T_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_1 = 00$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$T_2 = 01$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$T_3 = 10$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$T_4 = 11$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Good/Faulty Circuit Responses
- $f_0$, $f_2$, and $f_6$ are equivalent
- $f_1$ dominates $f_2$ and $f_6$

**Fault Table Reduction**

To collapse faults, remove all but one of equivalent columns and all dominating columns

<table>
<thead>
<tr>
<th>Fault Table</th>
<th>Collapsed Fault Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1$</td>
<td>$f_2$</td>
</tr>
<tr>
<td>$T_1$</td>
<td>$T_2$</td>
</tr>
<tr>
<td>$T_4$</td>
<td>$T_5$</td>
</tr>
</tbody>
</table>

Note: an empty row (e.g., $T_1$) is dominated by every row

**Fault Simulation Algorithms**

- Goal: determine the list of faults in a CUT that are detected by a specific test vector
- The general procedure is to simulate the good and faulty circuits and determine if they produce different outputs
- Consists of five specific tasks:
  - Good circuit simulation
  - Fault specification (fault list generation and collapsing)
  - Fault insertion
  - Fault-effect generation and propagation
  - Fault detection and discarding

**Fault Simulation Algorithms II**

- Fault simulation is one of the most widely used of the test technologies presented herein - many efficient algorithms for fault simulation have been developed.
- Major types:
  - Parallel fault simulation
  - Deductive fault simulation
  - Concurrent fault simulation
  - Parallel Pattern Single Fault Propagation (PPSFP)
- Most other work on fault simulation has been in improving the efficiency of these types of fault simulation or actually paralleling the fault simulation algorithms to be run on parallel/distributed computers
**DDQ Testing**

- **DDQ testing** is becoming more prevalent both in research and in new industrial applications.
  - All of the test techniques discussed thus far use voltage measurement techniques.
- **DDQ testing** is based on the physical fact that fault-free CMOS circuits consume VERY LITTLE current in the quiescent state.
  - Quiescent current for MOS devices (IDDQ) is typically in the fA range.
- The presence of faults, under the right conditions, can increase this quiescent current by several orders of magnitude which can be used to detect the fault.

**Advantages of DDQ Testing**

- Test generation is easier
  - Faults must be activated, but not propagated to a Primary Output
- **DDQ testing** can detect defects that are not modeled by the stuck-at model
  - Bridging faults
  - Gate oxide defects
  - Shorts between any two of the four terminals of a transistor
  - Partial defects - defects that do not affect the logic of the circuit, but may affect reliability
  - Some delay faults
  - Some stuck-open faults

**Disadvantages of DDQ Testing**

- Since normal **DDQ** is very low, measurements must be very precise
  - Measurement takes a significant amount of time (1 ms) relative to the voltage measurement techniques
  - Setting **DDQ** threshold for bad devices is hard
- Circuit-under-test must be suitable for **DDQ** testing - certain restrictions must be placed on the design
  - Must contain all static devices (which is slower), i.e., no
    - Dynamic circuitry
    - Pull-ups or pull-downs on I/O buffers
    - Specialized speed optimized circuitry such as RAM sense amps that draw significant static current

---

*DDQ Testing II*

When input goes, PMOS turns on and significant current flows between the source and gate such that IDDQ increases dramatically. However, because the gate output goes high as it should, traditional stuck-at fault testing would not detect this defect.
**$I_{DDQ}$ Fault Modeling**

- There are several fault models that can be used for $I_{DDQ}$ test generation - all of them at the logic or transistor level.
  - Stuck-at Fault Model
    - Drive the faulty node to the value opposite the stuck-at value
  - Transistor Short Model
    - Specific patterns can be derived to test for all possible combinations of shorts between all four terminals
  - Bridging Fault Model
    - Only physically adjacent nodes need to be tested
    - Drive the adjacent nodes to opposite values

**$I_{DDQ}$ Test Generation**

- There are several different methods that can be used to develop $I_{DDQ}$ tests. Most are used in conjunction with voltage (value) testing for the best speed/quality tradeoff.
  - Every Vector $I_{DDQ}$
    - Utilize logic test patterns developed for voltage-sensing test
    - Measure $I_{DDQ}$ after every vector
    - Most useful for first silicon prototype testing
  - Selective $I_{DDQ}$
    - Measure $I_{DDQ}$ measurement on selected subset of entire vector set
    - Run entire functional test at speed, but “pause” after vector selected for $I_{DDQ}$ measurement
  - Supplemental $I_{DDQ}$
    - Add specific set of vectors designed for $I_{DDQ}$ measurement to the end of the full-speed functional test

**$I_{DDQ}$ Fault Modeling II**

For bridging faults and transistor shorts, very high fault coverage is obtained for a very few vectors. This doesn’t mean that this necessarily applies for defect coverage.

**$I_{DDQ}$ Test Measurement Techniques**

- Off-Chip Measurement Unit
  - Built-in current monitors have been proposed, but not yet widely realized
  - A major consideration is not degrading the at-speed performance of the device-under-test

- On-Chip Measurement Unit
**I_DDQ Design for Testability**

- In order to ensure/improve $I_{DDQ}$ testability, several design constraints must be applied to limit good circuit $I_{DDQ}$:
  - Internal Tri-state Buses
    - Short periods of bus contention may be functionally OK, but cause problems with $I_{DDQ}$ testability
    - Design controllers for non-overlapping bus drivers
  - Pull-ups and Pull-downs
    - Pull-up (down) resistors are commonly used in I/O pads
    - Eliminate or use separate power supply for I/O pads

**Design for Testability**

- Design Constraints (continued)
  - Dynamic Circuitry
    - Precharge/discharge type logic typically used for high-speed design
    - Ensure all nodes are precharged on every clock cycle
  - Circuits with Non-Zero Static Current
    - Sense-Amps for memory cells, etc.
    - Avoid or use separate power supply

---

**Design for Testability II**

- Goal: increase the ease with which a device can be tested
  - Increase the controllability and observability of internal points in the circuit
- Three major approaches
  - Ad-hoc DFT techniques
  - Scan-based DFT techniques
  - Self-Test (BIST) techniques
- Problem is getting harder
  - Increasing complexity and heterogeneous combination of modules in system-on-a-chip.
  - Advanced packaging and assembly techniques extend problem to the board level
**Ad-Hoc DFT: Partitioning**

- “Divide and Conquer” approach
  - Physically divide the system into multiple chips or boards
  - On board-level systems use jumper wires to divide subunits
  - Has major performance penalties

**Ad-Hoc DFT: Degating**

- Another technique for separating modules on a chip/board with lower performance penalties

**Ad-Hoc DFT: Test Point Insertion**

- Insert additional lines to control and observe internal nodes
  - Use of multiplexers to reduce increased I/O requirement

**Ad-Hoc DFT: Bus Structured Architecture**

- By forcing the design to be bus structured, internal control and observe points are increased and can be used as test points
Scan-Based DFT

- Goal: make FSMs testable by making the internal state variables controllable and observable
  - This is accomplished by substituting the internal state latches (registers) with scannable latches (registers)

LSSD

- Advantages
  - With LSSD, the testing problem is transformed from one of sequential circuit testing to one of combinational circuit testing
  - By adding controllability/observability to the state variables, LSSD also eases functional testing

- Disadvantages
  - Area overhead
  - Speed overhead - need additional time to latch the next state into the LSSD registers
  - Testing overhead - need additional time to scan in/out test vectors and responses and at-speed testing is not supported
  - Clock generation and distribution is more difficult

Polarity-Hold SRL (Shift-Register Latch)

- Introduced at IBM and set as company policy
  - Official name: “Level Sensitive Scan Design” or LSSD

Scan-Path Register
**Scan-based Test — Operation**

![Scan-based Test Diagram](image)

**Boundary Scan DFT**

- Consists of adding scan registers to the inputs and outputs of the ICs
- Allow for efficient testing at the board level
  - Testing of board-level interconnect
  - Isolation and testing of chips via chip-level BIST or the application of chip-level tests via the test bus
- Requires 4 additional I/O ports - Test Access Port (TAP)
  - TCK - test clock
  - TMS - test mode signal
  - TDI/TDO - serial test data in/out
- Also requires additional logic to control the testing procedure - TAP controller

**Example: Scan-Path Testing**

![Example: Scan-Path Testing Diagram](image)

Partial-Scan can be more effective for pipelined datapaths

**JTAG (IEEE 1149.1)**

![JTAG Diagram](image)

Boundary Scan (JTAG) Chip Architecture
**JTAG (IEEE 1149.1) II**

Boundary Scan (JTAG) Board-level Interconnection

- Printed-circuit board
- Logic
- Packaged IC
- Scan-in
- Scan-out
- Bonding Pad

**Boundary Scan II**

- **Advantages**
  - Area and speed overhead are lower than scan design
  - Boundary-scan can be used to do functional testing/debugging
    - IC internal functional tests
    - IC cluster functional tests
    - IC/cluster emulation - control internal buses and nets
    - Hardware/Software integration tests - use internal scan to load/examine registers, single step, load microcode, etc.

- **Disadvantages**
  - Boundary scan has some area, speed, and testing overhead in the same manner as other scan design

**Built-In Self Test (BIST)**

- BIST is an active technique where the device is designed to test itself (with a little help)

  - Stimulus Generator
  - (Sub)-Circuit Under Test
  - Response Analyzer
  - Test Controller

  Rapidly becoming more important with increasing chip-complexity and larger modules

**Test Pattern Generation for BIST**

- There are several ways that test patterns for BIST can be generated (remember that the device itself is generating the test patterns)

  - Exhaustive Testing - apply all $2^n$ input patterns to a combinational circuit with $n$ inputs
    - Binary counter can be used as a TPG
  - Pseudorandom Testing - generate patterns that appear to be random but are in fact deterministic (repeatable)
    - LFSR used as a TPG
    - Weighted Pseudorandom - weight prob. “1” and “0” differently
    - Adaptive Pseudorandom - modify weight based on the output of fault simulation
  - Pseudoexhaustive Testing - segment device and test each portion exhaustively
**Linear-Feedback Shift Register (LFSR)**

- Basic building blocks
  - Unit delays (D-FFs)
  - Modulo-2 adders
  - Modulo-2 scalar multiplier
- Preserve the principle of superposition (that’s why it is linear)
  - Response to a linear combination of inputs is the linear combination of the responses of the circuit to the individual stimuli

Pseudo-Random* Pattern Generator

*Likelihood of “1” and “0” is 50% but patterns are deterministic/repeatable.

---

**Signature Analysis: Motivation**

- Test patterns for BIST can be generated at-speed by an LFSR with only a clock input
- The outputs of the CUT must be compared to the known good response
- In general, collecting each output response and off-loading it from the CUT for comparison is too inefficient to be practical
- The general solution is to compress the entire output stream into a single signature value
  - And hope that the final results will be unique

---

**Signature Analysis Basic**

- Signature analysis is a compression technique based on the concept of cyclic redundancy checking (CRC)
  - The simplest form of this technique is based on a single LFSR

Counts transitions on single-bit stream
- Compression in time

---

**Build-In Block Observer (BILBO)**

A BILBO is a logic block that can perform different functions, depending on the state of its Mode inputs

<table>
<thead>
<tr>
<th>Mode inputs</th>
<th>Operation mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1</td>
<td>Normal</td>
</tr>
<tr>
<td>0 0</td>
<td>Scan</td>
</tr>
<tr>
<td>1 0</td>
<td>Pattern generation or signature analysis</td>
</tr>
<tr>
<td>0 1</td>
<td>Reset</td>
</tr>
</tbody>
</table>

---

2102-545 Digital ICs
Test Generation Terminology

Test Vector
- An input vector for the circuit-under-test that causes the presence of a fault to be observable at a primary output.

Automatic Test Pattern Generation (ATPG)
- The process of generating a test pattern for a specific fault using some type of algorithm.

Detected Fault
- A fault for which a valid test vector has been generated.

Undetected Fault
- A fault for which a test vector has not been generated.

Redundant Fault
- A fault for which no test pattern exists (because of redundant logic in the circuit).

Fault Coverage
- The percentage of total faults for which test patterns have been generated:

Fault Coverage = \( \frac{\text{Number of Detected Faults}}{\text{Total Number of Faults in the CUT}} \times 100\% \)

Fault Efficiency
- The percentage of faults that either are detected or PROVEN redundant (usually used to measure the effectiveness of a test generator):

Fault Efficiency = \( \frac{\text{Number of Detected Faults} + \text{Number of Redundant Faults}}{\text{Total Number of Faults in the CUT}} \times 100\% \)

Controllability
- A testability metric that measures the difficulty in driving a node to a specific value.

Observability
- A testability metric that measures the difficulty in propagating the value on a node to a primary output.

Testability Measure
- A metric that attempts to determine how difficult it will be to generate a test for a specific line in the circuit:
  - Provides feedback to the designer on testability without actually performing test generation.
  - Assists in the test generation process.
  - Is based on controllability and observability.
Test Generation Terminology IV

**Sensitization**
- The process of driving the circuit to a state where the fault causes an actual erroneous value in the device at the point of the fault
  - For example, for single stuck-at faults, driving the node to the value opposite the stuck-at value

**Propagation**
- The process of driving the circuit to a state where the error becomes observable at the primary outputs

**Justification**
- The process of determining the input combination necessary to drive an internal circuit node to a specified value (consistency)

Fault Simulation Terminology

**Good Circuit**
- A logic model of the CUT without any faults inserted

**Faulty Circuit**
- A logic model of the CUT with one or more fault models inserted

**Fault Specification**
- Defining the set of modeled faults and performing fault collapsing

**Fault insertion**
- Selecting a subset of faults to be simulated and creating the data structures to indicate the presence of the faults

Fault Simulation Terminology II

**Equivalent Fault**
- Two faults $f_i$ and $f_j$ are equivalent if there is no test that will distinguish between them

**Dominant Fault**
- A fault $f_i$ dominates a fault $f_j$ if every test that detects $f_i$ also detects $f_j$

**Fault Collapsing**
- The process of reducing the fault set by removing equivalent (and dominated) faults

Built-In Self Test Terminology

**Built-In Self Test (BIST)**
- The capability of a chip, board, or system to test itself
  - To achieve the goal of BIST, the design may
    - Incorporate extra devices necessary for the test
    - Use existing parts/components already available.

**Built-In Test Equipment (BITE)**
- The hardware/software incorporated into a unit to provide DFT or BIST

**On-Line BIST**
- BIST in which testing occurs during normal operation
**Built-In Self Test Terminology II**

**Concurrent On-Line BIST**
- A form of on-line BIST in which testing occurs simultaneously with normal function

**Non-Concurrent On-Line BIST**
- A form of on-line BIST where testing is carried out while the system is in an idle state

**Off-Line BIST**
- BIST in which testing occurs when the system is not in its normal operation

**Functional Off-Line BIST**
- Off-line BIST that uses tests based on the functional behavior of the circuit-under-test

---

**Built-In Self Test Terminology III**

**Structural Off-Line BIST**
- Off-line BIST that uses tests based on the functional behavior of the circuit-under-test

**Pseudo Random Pattern Generator (PRPG)**
- A multi-output device that generates pseudorandom output patterns
  - Usually implemented with a Linear Feedback Shift Register (LFSR)

**Multiple-Input Signature Register (MISR)**
- A multi-input device that compresses a series of input patterns into a (pseudo) unique signature