Lecture 10: Logic Diagnosis

Instructor: M. Tahoori

Problem: Fault Diagnosis

- Circuit Under Diagnosis (CUD)
- Test patterns
- A chip with defects inside
- Expected response
- Faulty response
- Not equal!

Question: Where are the fault locations?
### Diagnosis For Yield Improvement

- **Golden Reference Model**
- **Logic Diagnosis**
- **A Set Of Potential Defect Locations**
- **Physical Failure Analysis**
  - Scanning Electronic Microscope (SEM)
  - Focused Ion Beam (FOB)
  - Via void
  - Mouse bite, etc.
- **Defect Mechanisms**
- **Tune the Manufacturing Process or Design for Yield Improvement**

### Design For Diagnosis

- **Complexity Of Diagnosis**
- **Original Design**
  - interface circuitry
- **Separated Logic & Memory**
  - Scan-chain
- **Logic Design With Full-Scan**
- **More Supporting Circuitry**
Possible Assumptions Used in Diagnosis

- Stuck-At Fault Model Assumption
  - The defect behaves like a stuck-at fault
- Single Fault Assumption
  - Only one fault affecting any faulty output
- Logical Fault Assumption
  - A fault manifests itself as a logical error
- Full-Scan Assumption
  - The chip under diagnosis has to be full-scanned

Note: A diagnosis approach less dependent on the fault assumptions is more capable of dealing with practical situations.

Major Approaches

- Cause-Effect Analysis
- Effect-Cause Analysis
- Diagnostic Test Pattern Generation
Terminology

- **Device Under Diagnosis (DUD):** The Failing Chip
- **Circuit Under Diagnosis (CUD):** The Circuit Model
- **Failing Input Vector:** Causes Mismatches

![Gate-level CUD](image)

Cause-Effect Analysis

- **Fault dictionary (pre-analysis of all causes)**
  - Records test response of every fault under the applied test set
  - Built by intensive fault simulation process
- **A chip is diagnosed (effect matching)**
  - By matching up the failing syndromes observed at the tester with the pre-stored fault dictionary
### Fault Dictionary Example

(a) Circuit under diagnosis

<table>
<thead>
<tr>
<th>Circuits</th>
<th>Test vectors in terms of ((a, b, c))</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(v_1)</td>
</tr>
<tr>
<td>fault-free</td>
<td>0</td>
</tr>
<tr>
<td>(f_1)</td>
<td>0</td>
</tr>
<tr>
<td>(f_2)</td>
<td>1</td>
</tr>
<tr>
<td>(f_3)</td>
<td>1</td>
</tr>
<tr>
<td>(f_4)</td>
<td>0</td>
</tr>
<tr>
<td>(f_5)</td>
<td>0</td>
</tr>
</tbody>
</table>

(b) Full-response dictionary

A diagnosis session: traverse from a path from root to a leaf

(c) Diagnostic tree

---

### Terminology: Mismatched Output

Effect-cause analysis does not build fault dictionary. It predicts fault locations by analyzing CUD from mismatch PO’s.
Structural Pruning – Intersection or Union?

(a) Cone intersection.

(b) Cone union when there are multiple faults.

Backtrace Algorithm

- Trace back from each mismatched PO
  - To find out suspicious faulty locations

- Functional Pruning
  - During the traceback, some signals can be disqualified from the fault candidate set based on their signal values.

- Rules
  - (1) At a controlling case (i.e., 0 for a NAND gate): Its fanin signals with non-controlling values (i.e., 1) are excluded from the candidate set.
  - (2) At a non-controlling case (i.e., 1 for a NAND gate): Every fanin signal remains in the candidate set.
**Backtrace Example**

All suspicious fault locations are marked in red.

Target mismatched output

**Diagnostic Test Pattern Generation**

DTPG helps to increase diagnostic resolution

Model for differentiating vector generation

d, stuck-at 1

d, stuck-at 0
Scan Test and Diagnosis

Flush test of scan chains
(pumping random patterns and checking response)

Pass

Pass or Fail?

Fail

Test Combinational Logic

Find failing scan chain(s)
Classify fault types

Scan Chain Diagnosis

Commonly Used Fault Types in Scan Chains

Scan Chain Faults

Functional Faults

Stuck-at Bridging

Setup-Time Violation Fault

Hold-Time Violation Fault

Timing Faults

Slow-To-Rise Fault

Slow-To-Fall Fault

Each fault could be permanent or intermittent.

Copyright 2016, M. Tahoori
TDSII: Lecture 10
A Stuck-At Fault In the Chain

Effect: A killer of the scan-test sequence

Combinational Logic

input pins

scan-input (SI)

output pins

scan-enable clock

Example: Faulty Syndrome of a Scan Chain

A scan chain

SI (scan input pin)

A faulty flip-flop

SO (scan output pin)

<table>
<thead>
<tr>
<th>Fault Type</th>
<th>Scan-In Pattern</th>
<th>Observed Syndrome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stuck-at-0</td>
<td>1101100110011000</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>Stuck-at-1</td>
<td>1101100110011000</td>
<td>1111111111111111</td>
</tr>
<tr>
<td>Slow-to-Rise</td>
<td>1101110011001100</td>
<td>1000100010001000</td>
</tr>
<tr>
<td>Slow-to-Fall</td>
<td>1101110011001100</td>
<td>1101110111011111</td>
</tr>
</tbody>
</table>

The rightmost bit goes into the scan first
The rightmost bit gets out of the scan first
An underlined bit in the observed image is failing.
Scan Chain Diagnosis Flow

- **Circuit Under Diagnosis**
- **Diagnostic Test Sequence Generator**
- **Diagnostic Test Sequences**
- **Test Application**
- **Fault-Free Observed Images**
- **Faulty FF’s location**
- **Signal Profiling Based Diagnosis Program**
- **Observed Images Of Failing Chip**

**Definition: Snapshot Image**

Def: A snapshot image is the combination of flip-flop values at certain time instance.

Snapshot image: \{(F_1, F_2, F_3, F_4) | (0, 1, 0, 1)\}
Definition: Observed Image

Def: An observed image is the scanned-out version of a snapshot image.

Snapshot image: \(\{(F_1, F_2, F_3, F_4) | (0, 1, 0, 1)\}\)
Observed image: \(\{(F_1, F_2, F_3, F_4) | (0, 0, 0, 1)\}\)

Modified Inject-and-Evaluate Paradigm

Step 1: Scan-in an ATPG pattern
Step 2: Capture the response to FF’s
Step 3: Scan-out and compare

A stuck-at-0 fault is assumed at the output of the 2nd FF from SI
Test Application: Run-and-Scan

Step 1: Apply a test sequence from PI's
Setting up a snapshot image at FF's

Step 2: Scan-out an observed image

The fault location is embedded in the observed image

Summary: Scan Chain Diagnosis

- Hardware Assisted
  - Extra logic on the scan chain
  - Good for stuck-at fault

- Fault Simulation Based
  - To find a faulty circuit matching the syndromes
  - Tightening heuristic $\rightarrow$ upper & lower bound
  - Use single excitation pattern for better resolution

- Profiling-Based Method
  - Locate the fault directly from the difference profiles obtained by run-and-scan test
  - Applicable to bridging faults
  - Use signal processing techniques such as filtering and edge detection
Diagnosis for BIST Logic

Diagnosis in a BIST environment requires
- determining from compacted output responses which test vectors have produced a faulty response (*time information*)
- determining from compacted output responses which scan cells have captured errors (*space information*)

The true fault location inside the logic can then be inferred from the above space and time information using combinational logic diagnosis.

---

Deterministic Masking-Based Diagnosis

(a) STUMP-based BIST architecture

- Core Logic
- MISR (Multiple-Input Signal Analyzer)
- PRPG (Pseudo-Random Pattern Generator)

(b) Scan cell matrix

- Scan chain index (X)
- Scan slice index

Cell partition:
- X = {3,4} (chain set)
- Y = 2 (lower bound)
- Z = 6 (upper bound)
Circuitry to Support Deterministic Masking

PRPG (Pseudo-Random Pattern Generator)

Core Logic

MISR (Multiple-Input Signal Analyzer)

Counter

A Search for Scan Cells Capturing Errors

PRPG (Pseudo-Random Pattern Generator)

Core Logic

MISR (Multiple-Input Signature Register)

(a) Scan cells capturing errors in the fourth scan chain

(b) The search tree

9 BIST sessions

(Y, Z)=(1, 7)

(Y, Z)=(1, 4)

(Y, Z)=(5, 7)

(Y, Z)=(3, 4)

(Y, Z)=(5, 6)

(Y, Z)=(7, 7)

(Y, Z)=(3, 3)

(Y, Z)=(4, 4)