Digital Integrated Circuits
Lecture 19: Design for Testability

Chih-Wei Liu
VLSI Signal Processing LAB
National Chiao Tung University
cwliu@twins.ee.nctu.edu.tw
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 opportunity 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
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
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 parts
- 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
TestosterICs

- Ex: TestosterICs functional chip tester
  - Designed by clinic teams and David Diaz at HMC
  - Reads your IRSIM test vectors, applies them to your chip, and reports assertion failures
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

Stuck-At-1
SA1 Fault

Stuck-At-0
SA0 Fault
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 Example

SA1
- A_3
- A_2
- A_1
- A_0
- n1
- n2
- n3
- Y

SA0

Minimum set:
Test Example

<table>
<thead>
<tr>
<th></th>
<th>SA1</th>
<th>SA0</th>
</tr>
</thead>
<tbody>
<tr>
<td>A3</td>
<td>{0110}</td>
<td>{1110}</td>
</tr>
<tr>
<td>A2</td>
<td>{1010}</td>
<td>{1110}</td>
</tr>
<tr>
<td>A1</td>
<td>{0100}</td>
<td>{0110}</td>
</tr>
<tr>
<td>A0</td>
<td>{0110}</td>
<td>{0111}</td>
</tr>
<tr>
<td>n1</td>
<td>{1110}</td>
<td>{0110}</td>
</tr>
<tr>
<td>n2</td>
<td>{0110}</td>
<td>{0100}</td>
</tr>
<tr>
<td>n3</td>
<td>{0101}</td>
<td>{0110}</td>
</tr>
<tr>
<td>Y</td>
<td>{0110}</td>
<td>{1110}</td>
</tr>
</tbody>
</table>

- Minimum set: \{0100, 0101, 0110, 0111, 1010, 1110\}
Design for Test

- Design the chip to increase observability and controllability

- If each register could be observed and controlled, test problem reduces to testing combinational logic between registers.

- Better yet, logic blocks could enter test mode where they generate test patterns and report the results automatically.
Scan

- Convert each flip-flop to a scan register
  - Only costs one extra multiplexer
- Normal mode: flip-flops behave as usual
- Scan mode: flip-flops behave as shift register

- Contents of flops can be scanned out and new values scanned in
Scannable Flip-flops

(a) Scanning input SI
(b) Scanning input D

(c) Scanning control signals φ_s and φ_d
Built-in Self-test

- Built-in self-test lets blocks test themselves
  - Generate pseudo-random inputs to comb. logic
  - Combine outputs into a syndrome
  - With high probability, block is fault-free if it produces the expected syndrome
Linear Feedback Shift Register
- Shift register with input taken from XOR of state
- Pseudo-Random Sequence Generator

<table>
<thead>
<tr>
<th>Step</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>111</td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>
**Linear Feedback Shift Register**
- Shift register with input taken from XOR of state
- *Pseudo-Random Sequence Generator*

<table>
<thead>
<tr>
<th>Step</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>111</td>
</tr>
<tr>
<td>1</td>
<td>110</td>
</tr>
<tr>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>3</td>
<td>010</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
</tr>
<tr>
<td>5</td>
<td>001</td>
</tr>
<tr>
<td>6</td>
<td>011</td>
</tr>
<tr>
<td>7</td>
<td>111 (repeats)</td>
</tr>
</tbody>
</table>
BILBO

- Built-in Logic Block Observer
  - Combine scan with PRSG & signature analysis

---

**MODE** | **C[1]** | **C[0]**
---|---|---
Scan | 0 | 0
Test | 0 | 1
Reset | 1 | 0
Normal | 1 | 1
Boundary Scan

- Testing boards is also difficult
  - Need to verify solder joints are good
    - Drive a pin to 0, then to 1
    - Check that all connected pins get the values
  - Through-hole boards used “bed of nails”
  - SMT and BGA boards cannot easily contact pins
  - Build capability of observing and controlling pins into each chip to make board test easier
Boundary Scan Example
Boundary Scan Interface

- Boundary scan is accessed through five pins
  - TCK: test clock
  - TMS: test mode select
  - TDI: test data in
  - TDO: test data out
  - TRST*: test reset (optional)

- Chips with internal scan chains can access the chains through boundary scan for unified test strategy.
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)”