19. Introduction to Test

• Previous Unit:
  – Circuit pitfalls
  – Noise
  – Reliability

• This Unit:
  – Introduction to testing
  – Logical faults corresponding to defects
  – DFT

Outline

• Testing
  – Logic Verification
  – Silicon Debug
  – Manufacturing Test

• Fault Models
• Observability and Controllability
• Design for Test
  – Scan
  – BIST
• Boundary Scan

Testing

• Testing, testing, testing!

• Three main categories
  – Functionality test or logic verification (before tapeout)
    • Make sure functionality is correct
  – Silicon debug (on first batch of chips from fab)
    • “detective” work
    • You don’t want to mass-produce “bad” chips
  – Manufacturing test (on each mfg’d chip before shipping)
    • You don’t want to ship “bad” chips

Testing a chip can occur at various levels

– Wafer level $0.01-$0.10
– Packaged chip level $0.10-$1
– Board level $1-$10
– System level $10-$100
– Field level $100-$1000

Cost goes up exponentially if fault detected at later stages
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:
    - $0, 1, 2, 2^{16}, -1, -2^{16}$, 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
  - But 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

Cheap Testers

- Tester and test fixtures
  - Can be very expensive (e.g., $1-2M)
- 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
  - Reads test vectors, applies them to your chip, and reports assertion failures
  - A low cost digital VLSI tester
19. Introduction to Test

Stuck-At Faults

• How does a chip fail?
  – Need “fault model”
  – 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

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
19. Introduction to Test

Test Example

- A\_3
- A\_2
- A\_1
- A\_0
- n1
- n2
- n3
- Y

Minimum set:

Test Example

- A\_3
- A\_2
- A\_1
- A\_0
- n1
- n2
- n3
- Y

Minimum set:

Test Example

- A\_3
- A\_2
- A\_1
- A\_0
- n1
- n2
- n3
- Y

Minimum set:
### Test Example

- **A₃**: \{0110\}, \{1110\}
- **A₂**: \{1010\}, \{1110\}
- **A₁**: \{0100\}, \{0110\}
- **A₀**: \{0110\}, \{0111\}
- **n₁**: \{1110\}, \{0110\}
- **n₂**: \{0110\}, \{0100\}
- **n₃**: \{0101\}, \{0110\}
- **Y**: \{\}

**Minimum set:**

- **A₃**: \{0110\}, \{1110\}
- **A₂**: \{1010\}, \{1110\}
- **A₁**: \{0100\}, \{0110\}
- **A₀**: \{0110\}, \{0111\}
- **n₁**: \{1110\}, \{0110\}
- **n₂**: \{0110\}, \{0100\}
- **n₃**: \{0101\}, \{0110\}
- **Y**: \{\}

---

### Test Example

- **A₃**: \{0110\}, \{1110\}
- **A₂**: \{1010\}, \{1110\}
- **A₁**: \{0100\}, \{0110\}
- **A₀**: \{0110\}, \{0111\}
- **n₁**: \{1110\}, \{0110\}
- **n₂**: \{0110\}, \{0100\}
- **n₃**: \{0101\}, \{0110\}
- **Y**: \{\}

**Minimum set:**

- **A₃**: \{0110\}, \{1110\}
- **A₂**: \{1010\}, \{1110\}
- **A₁**: \{0100\}, \{0110\}
- **A₀**: \{0110\}, \{0111\}
- **n₁**: \{1110\}, \{0110\}
- **n₂**: \{0110\}, \{0100\}
- **n₃**: \{0101\}, \{0110\}
- **Y**: \{\}

---

### Test Example

- **A₃**: \{0110\}, \{1110\}
- **A₂**: \{1010\}, \{1110\}
- **A₁**: \{0100\}, \{0110\}
- **A₀**: \{0110\}, \{0111\}
- **n₁**: \{1110\}, \{0110\}
- **n₂**: \{0110\}, \{0100\}
- **n₃**: \{0101\}, \{0110\}
- **Y**: \{\}

**Minimum set:**

- **A₃**: \{0110\}, \{1110\}
- **A₂**: \{1010\}, \{1110\}
- **A₁**: \{0100\}, \{0110\}
- **A₀**: \{0110\}, \{0111\}
- **n₁**: \{1110\}, \{0110\}
- **n₂**: \{0110\}, \{0100\}
- **n₃**: \{0101\}, \{0110\}
- **Y**: \{\}
19. Introduction to Test

Test Example

- \( A_3 \) \( \{0110\} \) \( \{1110\} \)
- \( A_2 \) \( \{1010\} \) \( \{1110\} \)
- \( A_1 \) \( \{0100\} \) \( \{0110\} \)
- \( A_0 \) \( \{0110\} \) \( \{0111\} \)
- \( n_1 \) \( \{1110\} \) \( \{0110\} \)
- \( n_2 \) \( \{0110\} \) \( \{0100\} \)
- \( n_3 \) \( \{0101\} \) \( \{0110\} \)
- \( Y \) \( \{0110\} \) \( \{1110\} \)

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

Scannable Flip-flops

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

![Scan Circuit Diagram](image)

![Scannable Flip-flops](image)
**Built-in Self-test**

- Built-in self-test lets blocks test themselves
  - Generate pseudo-random inputs to combinational logic
  - Combine outputs into a syndrome
  - With high probability, block is fault-free if it produces the expected syndrome

**PRSG**

- **Linear Feedback Shift Register**
  - Shift register with input taken from XOR of state
  - *Pseudo-Random Sequence Generator*

  ![PRSG Diagram]

<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>100</td>
</tr>
<tr>
<td>4</td>
<td>110</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
</tr>
<tr>
<td>7</td>
<td>100</td>
</tr>
</tbody>
</table>

![PRSG Diagram]

<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>100</td>
</tr>
<tr>
<td>4</td>
<td>110</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
</tr>
<tr>
<td>7</td>
<td>100</td>
</tr>
</tbody>
</table>
PRSG

- **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>101</td>
</tr>
<tr>
<td>5</td>
<td>010</td>
</tr>
<tr>
<td>6</td>
<td>001</td>
</tr>
<tr>
<td>7</td>
<td>100</td>
</tr>
</tbody>
</table>

PRSG

- **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>110</td>
</tr>
</tbody>
</table>
19. Introduction to Test

PRSG

• *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>

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-hold 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

BILBO

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

Boundary Scan Example

• *Serial Data In*
• *Serial Data Out*
• *Package Interconnect*
• *IO pad and Boundary Scan Cell*

CHIP A
CHIP B
CHIP C
CHIP D
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)”