Lecture 12: Design for Testability
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 (1994)
  - 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
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
Manufacturing Failures

- Metal 1 Shelving
- Metal 5 film particle (bridging defect)
- Open defect
- Spongy Via2 (Infant mortality)
- Metal 5 blocked etch (patterning defect)
- Spot defects “Co” Defect under Gate
- Metal 1 missing pattern (open at contact)

SEM images courtesy Intel Corporation
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
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

<table>
<thead>
<tr>
<th>SA1</th>
<th>SA0</th>
</tr>
</thead>
<tbody>
<tr>
<td>A3</td>
<td></td>
</tr>
<tr>
<td>A2</td>
<td></td>
</tr>
<tr>
<td>A1</td>
<td></td>
</tr>
<tr>
<td>A0</td>
<td></td>
</tr>
<tr>
<td>n1</td>
<td></td>
</tr>
<tr>
<td>n2</td>
<td></td>
</tr>
<tr>
<td>n3</td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td></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) Scannable Flip-flop

(b) Scannable Flip-flop with Scan Paths

(c) Scannable Flip-flop with Separate Clock Phases
ATPG

- Test pattern generation is tedious
- Automatic Test Pattern Generation (ATPG) tools produce a good set of vectors for each block of combinational logic
- Scan chains are used to control and observe the blocks
- Complete coverage requires a large number of vectors, raising the cost of test
- Most products settle for covering 90+% of potential stuck-at faults
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
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></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>

Flops reset to 000
Built-in Logic Block Observer

- Combine scan with PRSG & signature analysis

### Diagram

- **Inputs:**
  - C[0]
  - C[1]
  - SI

- **Outputs:**
  - Q[0]
  - Q[1]
  - Q[2]

- **Logic Blocks:**
  - Flip-flops
  - OR-gates

- **Modes:**
  - **Scan:** C[1] = 0, C[0] = 0
  - **Test:** C[1] = 0, C[0] = 1
  - **Reset:** C[1] = 1, C[0] = 0
  - **Normal:** C[1] = 1, C[0] = 1

- **Additional Components:**
  - PRSG
  - Logic Cloud
  - Signature Analyzer

---

12: Design for Testability

CMOS VLSI Design 4th Ed.
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
Boundary Scan Example

CHIP A

IO pad and Boundary Scan Cell

Serial Data In

CHIP B

Package Interconnect

CHIP C

Serial Data Out

CHIP D

CMOS VLSI Design 4th Ed.
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.
Testing Your Class Project

- Presilicon Verification
  - Test vectors: corner cases and random vectors
  - HDL simulation of schematics for functionality
  - Use 2-phase clocking to avoid races
  - Use static CMOS gates to avoid electrical failures
  - Use LVS to ensure layout matches schematic
  - Don’t worry about timing

- Postsilicon Verification
  - Run your test vectors on the fabricated chip
  - Use a functional chip tester
  - Potentially use breadboard or PCB for full system
TestosterlCs

- TestosterlCs functional chip tester
  - Designed by clinic teams and David Diaz at HMC
  - Reads your test vectors, applies them to your chip, and reports assertion failures
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)”