CMOS Testing: Part 1

• Introduction
• Fault models
  – Stuck-line (single and multiple)
  – Bridging
  – Stuck-open
• Test pattern generation
  – Combinational circuit test generation
  – Sequential circuit test generation

Need for Testing

• Physical defects are likely in manufacturing
  – Missing connections (opens)
  – Bridged connections (shorts)
  – Imperfect doping, processing steps
  – Packaging
• Yields are generally low
  – Yield = Fraction of good die per wafer
• Need to weed out bad die before assembly
• Need to test during operation
  – Electromagnetic interference, mechanical stress, electromigration, alpha particles
Testing Levels and Test Costs

- Wafer
- Packaged chip
- Board
- System
- Field
- Concurrent checking

- Cost to detect a fault (per chip)
  - Wafer: $0.01-$0.1
  - Packaged chip: $0.1-$1
  - Board: $1-$10
  - System: $10-$100
  - Field: $100-$1000

Manufacturing Testing

Goal: Detect manufacturing defects

Defects: layer-to-layer shorts
discontinuous wires
thin-oxide shorts to substrate or well

Faults: nodes shorted to power or ground (stuck-at)
nodes shorted to each other (bridging)
inputs floating, outputs disconnected (stuck-open)
Testing and Diagnosis

- Testing: Determine if the system (chip, board) is behaving correctly
- Diagnosis: Locate the cause of malfunctioning
Testing : The Buzzwords

- Errors
  - Permanent
  - Intermittent
  - Transient
- Faults
  - Physical
  - Logical
- Test Evaluation
  - Fault coverage
  - Fault simulation
- Types of testing
  - Off-line, on-line
  - Self-test vs external test
  - DC (static) vs AC (at-speed)
  - Edge-pin, guided-probe, bed-of-nails, E-beam, in-circuit

Testing: The Inevitable Acronyms

- System under test
  - UUT: Unit Under Test
  - CUT: Circuit Under Test
  - DUT: Device Under Test
- The tester
  - ATE: Automatic Test Equipment
- Test generation
  - ATPG: Automatic Test Pattern Generation
- Fault Models
  - SSL: Single Stuck-Line
  - MSL: Multiple Stuck-Line
  - BF: Bridging Fault
- DFT: Design for Testability
  - BIST: Built-in self-test
  - LFSR: Linear-Feedback Shift-Register
Fault Models

- Defects are too many and too difficult to explicitly enumerate
- Abstraction (technology independence): presence of physical defect is modeled by changing the logic function (or delay)
- Reduced complexity: distinct physical defects may be represented by the same logical fault
- Generality: tests derived for logical faults may detect vaguely-understood or hard-to-analyze physical defects
- A test pattern detects a fault from the fault model

Single Stuck-Line (SSL) Model

- A single node in the circuit is stuck-at 1 (s-a-1) or 0 (s-a-0).

Fault-free function $z = AB + CD$
Fault-free function $z = AB + CD$
Faulty function $z^f = AB$
Faulty function $z^f = AB + D$

Number of possible stuck-at faults in a circuit with $n$ lines?
Number of faults reduced by finding equivalent classes
SSL Fault Detection

- A test pattern for fault $x \ s-a-d$ is an input combination that 1) places $\overline{d}$ on $x$ (activation), 2) propagates fault effect ($D$ or $\overline{D}$) to primary output

$$D: 1/0, \overline{D}: 0/1$$

Good circuit $\Rightarrow$ Bad circuit

$ABCE = 0011$ is a test pattern for $C \ s-a-0$

Multiple Stuck-Line (MSF) Faults

- More than one line may be stuck at a logic value

Fault: $\{C \ s-a-0, x \ s-a-1\}$

How many MSL fault can there be in a circuit with $n$ nodes?

How to get test patterns for MSL faults?

Fault universe is too large, MSL fault model seldom used, especially since tests for SSL faults cover many MSL faults
Bridging Faults

- Models short circuits, pairs of nodes considered
- Number of bridging faults?
- Feedback vs non-feedback bridging faults

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>z</th>
<th>zf</th>
<th>Wired-AND</th>
<th>Wired-OR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>?</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>?</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

What are the test patterns in this example?

Stuck-Open Faults

Fault-free circuit: \( z = a + b \)
Faulty circuit: \( z^f = a + b + abz' \)

\( z' \): Previous value of \( z \)

Case 1: \( a = b = 1 \), \( z \) pulled down to 0
Case 2: \( a = 1 \), \( b = 0 \), \( z \) retains previous state

A test for a stuck-open fault requires two patterns
{\( ab = 00, ab = 10 \)}
Test Pattern Generation

- Exhaustive testing: Apply $2^n$ pattern to $n$-input circuit
- Not practical for large $n$
- Advantage: Fault-model independent

Fault-Oriented Test Generation Algorithm:

1) Set $x$ to 1: activate fault
2) Justify $D$ on $x$, propagate $D$ to $z$

Set $C$ and $D$ to 1

Example test pattern: $ABCD = 0011$

- Backtracking may be necessary
- Test generation is NP-complete

Sequential Circuit Test Generation

- Difficult problem!
- Exhaustive testing requires $2^{m+n}$ patterns ($2^m$ states and $2^n$ transitions from each state)
- Every fault requires a sequence of patterns

Initializing sequence: drive to known state
Test activation
Propagation sequence: propagate discrepancy to observable output
Sequential Circuit Test Generation

- Iterative-array model (pseudo-combinational circuit)

Assume initial state of flip-flop is not known

Test pattern sequence: \{11X, 011\}

ABC = 11X

ABC = 011

Current time frame