ECE 538
VLSI System Testing
Krish Chakrabarty

Fault Modeling

- Why model faults?
- Some real defects in VLSI and PCB
- Common fault models
- Stuck-at faults
  - Single stuck-at faults
  - Fault equivalence
  - Fault dominance and checkpoint theorem
  - Classes of stuck-at faults and multiple faults
- Transistor faults
- Summary
Why Model Faults?

• I/O function tests inadequate for manufacturing test
• Real defects (often mechanical) too numerous and often not analyzable
• A fault model identifies targets for testing
• A fault model makes analysis possible
• Effectiveness measurable by experiments

Some Real Defects in Chips

• Processing defects
  - Missing contact windows
  - Parasitic transistors
  - Oxide breakdown
  - . . .
• Material defects
  - Bulk defects (cracks, crystal imperfections)
  - Surface impurities (ion migration)
  - . . .
• Time-dependent failures
  - Dielectric breakdown
  - Electromigration
  - . . .
• Packaging failures
  - Contact degradation
  - Seal leaks
  - . . .
Observed PCB Defects

<table>
<thead>
<tr>
<th>Defect classes</th>
<th>Occurrence frequency (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shorts</td>
<td>51</td>
</tr>
<tr>
<td>Opens</td>
<td>1</td>
</tr>
<tr>
<td>Missing components</td>
<td>6</td>
</tr>
<tr>
<td>Wrong components</td>
<td>13</td>
</tr>
<tr>
<td>Reversed components</td>
<td>6</td>
</tr>
<tr>
<td>Bent leads</td>
<td>8</td>
</tr>
<tr>
<td>Analog specifications</td>
<td>5</td>
</tr>
<tr>
<td>Digital logic</td>
<td>5</td>
</tr>
<tr>
<td>Performance (timing)</td>
<td>5</td>
</tr>
</tbody>
</table>


Common Fault Models

- Single stuck-at faults
- Transistor open and short faults
- Memory faults
- PLA faults (stuck-at, cross-point, bridging)
- Functional faults (processors)
- Delay faults (transition, path)
- Analog faults
- For more examples, see Section 4.4 (p. 60-70) of the book.
Fault Models

- Faults must match level of abstraction in terms of
  - Component types
  - Signal values
  - Time units
- **Example**: Logic (gate) level
  - Component types: lines, gates, flip-flops
  - Signal values: 0, 1
  - Time units: gate delays
- **Meaningless concepts at this level**: Voltage, short-circuit, lost message

Behavioral Fault Model

- Defined at highest-level of abstraction, i.e. behavioral specification of system
- Inject various types of faults into Verilog/VHDL descriptions, e.g.,
  - Variable $R$ assumed to be permanently at $V_L$ or $V_H$
  - A *call* to a function may be assumed to fail in such a way that it always returns $V_L$ or $V_H$.
  - The “*for* (CC) {B}” clause fails: either the body {B} is never executed, or always executed irrespective of condition (CC)
Behavioral Faults

- The “switch (id)” clause may fail:
  - All the specified cases are selected
  - None of the specified cases are selected
- An “if (Y) then \{B_1\} else \{B_2\}” construct may fail, e.g.
  - \{B_1\} is never executed and \{B_2\} is always executed
- The assignment “X := Y” may fail, e.g.
  - The value of X remains unchanged
- Must provide adequate coverage of low-level faults

Functional Fault Model

- Ad hoc, geared towards specific functional blocks, e.g. multiplexers, adders, counters
- Faults for a multiplexer:
  - A 0 or 1 cannot be selected on an input line
  - Two inputs are selected simultaneously. Output = ?
- Useful if not too complex for test generation, and provides good coverage of low-level faults
Functional Fault Models

• Example: truth table can change in any arbitrary way
  – Needs exhaustive (verification) testing, all $2^n$ inputs must be applied
  – Pseudoexhaustive testing possible

\[
\begin{align*}
x_1 & \rightarrow & z_1 \\
x_2 & \rightarrow & z_2 \\
\vdots & \rightarrow & \vdots \\
x_n & \rightarrow & z_m
\end{align*}
\]

\[u << n\]
Testing time = $m2^n << 2^n$

Single Cell Fault Model

• Cells can have any implementation
• All possible (combinational) cell faults are allowed; truth table can change in any way
• C-testability: constant number of test patterns, independent of circuit size
  (Ripple-carry adder needs only 8 test patterns for all single stuck-at faults)
Single Stuck-at Fault

- Three properties define a single stuck-at fault
  - Only one line is faulty
  - The faulty line is permanently set to 0 or 1
  - The fault can be at an input or output of a gate
- Example: XOR circuit has 12 fault sites and 24 single stuck-at faults

![XOR Circuit Diagram]

Test vector for h s-a-0 fault

Single Stuck-Line (SSL) Model

- **Advantages:**
  - Also called stuck-at
  - Matches circuit level, easy to use
  - Moderate number of faults (2n for an n-line circuit)
  - Tests for SSL faults provide good defect coverage (experiments)

- **Disadvantages:**
  - Does not account for timing/delay faults
  - Few physical defects behave like SSL faults
Single Stuck-Line Faults

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

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

SSL Fault Detection

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

$D_1: 1/0$, $D_0: 0/1$

Good circuit
Bad circuit

$ABCE = 0011$ is a test pattern for C s-a-0
### Testing a Gate for SSL Faults

2-input OR gate

\[
\begin{array}{c}
\text{a} \\
\text{b} \\
\end{array}
\quad \rightarrow \quad 
\begin{array}{c}
z \\
\end{array}
\]

2-input NAND gate

\[
\begin{array}{c}
\text{a} \\
\text{b} \\
\end{array}
\quad \rightarrow \quad 
\begin{array}{c}
z \\
\end{array}
\]

<table>
<thead>
<tr>
<th>Fault a/0 a/1 b/0 b/1 z/0 z/1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ab</td>
</tr>
<tr>
<td>00 x x x</td>
</tr>
<tr>
<td>01 x x x</td>
</tr>
<tr>
<td>10 x x x</td>
</tr>
<tr>
<td>11 x</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test ab</th>
</tr>
</thead>
<tbody>
<tr>
<td>00 x</td>
</tr>
<tr>
<td>01 x</td>
</tr>
<tr>
<td>10 x</td>
</tr>
<tr>
<td>11 x</td>
</tr>
</tbody>
</table>

### Equivalence and Dominance

- Faults \( f_1 \) and \( f_2 \) are **equivalent** if and only if (iff) every test pattern for \( f_1 \) is a test pattern for \( f_2 \), and vice versa.

**Example:** \( \{a/1, b/1, z/1\} \) form an **equivalence class**. Equivalent faults give rise to identical faulty functions.

- Fault \( f_1 \) **dominates** fault \( f_2 \) if and only if every test pattern for \( f_1 \) is a test pattern for \( f_2 \) (\( f_2 \) is **dominated** by \( f_1 \)).

**Example:** \( a/0 \) dominates \( z/0 \), \( b/0 \) dominates \( z/0 \)

\[ \Rightarrow f_1 \text{ and are } f_2 \text{ equivalent iff } f_1 \text{ dominates } f_2 \text{ and } f_2 \text{ dominates } f_1 \]

\[ \Rightarrow \text{Significant fault list collapsing is possible (drop dominated & equivalent faults)} \]
Fault Equivalence

• Number of fault sites in a Boolean gate circuit = #PI + #gates + # (fanout branches).
• Fault equivalence: Two faults f1 and f2 are equivalent if all tests that detect f1 also detect f2.
• If faults f1 and f2 are equivalent then the corresponding faulty functions are identical.
• Fault collapsing: All single faults of a logic circuit can be divided into disjoint equivalence subsets, where all faults in a subset are mutually equivalent. A collapsed fault set contains one fault from each equivalence subset.

Equivalence Rules

Diagram showing equivalence rules for AND, NAND, OR, NOR, NOT, WIRE, and FANOUT gates.
Equivalence Example

Faults in green removed by equivalence collapsing

Collapse ratio = \frac{20}{32} = 0.625

Fault Equivalence

- **Theorem**: An n-input NAND, NOR, AND or OR gate has n+2 equivalent classes:
  \{x_1/c\}, \{x_2/c\}, \ldots, \{x_n/c\}, \{z/(c\oplus i)\}, \{z/(c\oplus i), x_1/c, x_2/c, \ldots\}
- Local fault collapsing reduces number of faults by almost 50%
Fault Dominance

- If all tests of some fault $F_1$ detect another fault $F_2$, then $F_2$ is said to dominate $F_1$.
- Dominance fault collapsing: If fault $F_2$ dominates $F_1$, then $F_2$ is removed from the fault list.
- When dominance fault collapsing is used, it is sufficient to consider only the input faults of Boolean gates. See the next example.
- In a tree circuit (without fanouts) PI faults form a dominance collapsed fault set.
- If two faults dominate each other then they are equivalent.

Dominance Example

A dominance collapsed fault set
Testing Simple Gates

1) N-input AND, NAND gate: n +1 patterns

\[
\begin{array}{c}
0 \ 1 \ 1 \ \ldots \ 1 \\
1 \ 0 \ 1 \ \ldots \ 1 \\
1 \ 1 \ 0 \ \ldots \ 1 \\
\vdots \\
1 \ 1 \ 1 \ \ldots \ 0 \\
1 \ 1 \ 1 \ \ldots \ 1 \\
\end{array}
\]

1) N-input OR, NOR gate: n +1 patterns

\[
\begin{array}{c}
1 \ 0 \ 0 \ \ldots \ 0 \\
0 \ 1 \ 0 \ \ldots \ 0 \\
0 \ 0 \ 1 \ \ldots \ 0 \\
\vdots \\
0 \ 0 \ 0 \ \ldots \ 1 \\
0 \ 0 \ 0 \ \ldots \ 0 \\
\end{array}
\]

n-input XOR gate: 2 test patterns (n odd), 3 test patterns (n even)

Independent of n!

Testing a Circuit

- Controllability problem: Apply test patterns to inputs of the CUT that produces desired pattern at fault site (fault activation)
- Observability problem: Propagate an error from fault site to observable primary output
**Multiple Stuck-at (Stuck-Line) Faults**

- A multiple stuck-at fault means that any set of lines is stuck-at some combination of (0,1) values.
- The total number of single and multiple stuck-at faults in a circuit with $k$ single fault sites is $3^k - 1$.
- A single fault test can fail to detect the target fault if another fault is also present, however, such masking of one fault by another is rare.
- Statistically, single fault tests cover a very large number of multiple faults.

**Switch-Level Fault Models**

*Stuck-open Faults*

- Fault-free circuit: $z = a + b$
- Faulty circuit: $z^f = a + b + ab\bar{z}$

$\bar{z}$: 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\}

**Initialization vector**

**Test vector**
**Stuck-Open Example**

Vector 1: test for $A$ s-a-0 (Initialization vector)

Vector 2 (test for $A$ s-a-1)

*Two-vector s-op test can be constructed by ordering two s-at tests*

---

**Switch-Level Fault Models**

*Stuck-on Fault:* Transistor permanently on

- $T_1$ stuck-on: (1,1) possible test pattern
  Output for $T_2, T_2, T_3$ turned on should be 1

- $T_4$ stuck-on: (1,0) possible test pattern
  Output for $T_2, T_2, T_3$ turned on should be 0

⇒ Only one of these faults can be detected

- Limitation of voltage monitoring techniques
- Current monitoring techniques needed
**Geometric Fault Models**

*Bridging faults*: Derived from circuit layout

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

![Diagram of a bridge](image)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>$z_f$</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>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

$z_f = ?$

What are the test patterns in this example?

---

**Bridging Fault Models**

- Tests for resistive shorts between two normally unconnected nets
- Closest fault-model to real defects
- Bridging-faults are caused by manufacturing defects

![Diagram of bridging faults](image)

modeled as wired-AND, wired-OR or resistive shorts
Bridging Faults

Wired-OR or WIRED-AND?

abc = 110 creates intermediate voltage at intermediate nodes

Feedback vs Non-Feedback BF

Non-feedback bridging fault

Feedback bridging fault
Delay Faults

- Affect propagation delay of the circuit, circuit fails at high speeds
- More important for high-speed circuits
- Gate delay fault (GDF): slow 1-to-0 or 0-to-1 transition at a gate output
- Path delay fault (PDF): exists a path from a primary input to primary output that is slow to propagate a 0-to-1 or 1-to-0 transition
Types of GDFs

• Gross GDFs (G-GDF): gate delay defect size is greater than system clock period
  – DFs in all paths going through faulty gate, hence catastrophic
  – Also called transition faults (TFs)
• Small GDFs (S-GDF): delay defect size is smaller than system clock period
  – Detectable if causes PDF in at least one path through the gate

Delay Fault Terminology

• Two-pattern tests
• Test is robust if independent of delays in the rest of the circuit, else non-robust
• Robust test (RT) is a hazard-free robust test (HFRT) if no hazards can occur on tested path, irrespective of gate delays
• Non-hazard-free robust tests allow hazards on side inputs
Summary

• Fault models are analyzable approximations of defects and are essential for a test methodology.
• For digital logic single stuck-at fault model offers best advantage of tools and experience.
• Many other faults (bridging, stuck-open and multiple stuck-at) are largely covered by stuck-at fault tests.
• Stuck-short and delay faults and technology-dependent faults require special tests.
• Memory and analog circuits need other specialized fault models and tests.