Fault Modeling

李昆忠 Kuen-Jong Lee

Dept. of Electrical Engineering
National Cheng-Kung University
Tainan, Taiwan

VLSI Testing Class
Fault Modeling

- Some Definitions
- Why Modeling Faults
- Various Fault Models
- Fault Detection
- Fault Collapsing
Some Real Defects in Chips

- **Processing Faults**
  - missing contact windows
  - parasitic transistors
  - oxide breakdown

- **Material Defects**
  - bulk defects (cracks, crystal imperfections)
  - surface impurities (ion migration)

- **Packaging Failures**
  - Contact degradation
  - Seal leaks

- **Time-Dependent Failures**
  - Dielectric breakdown
  - Electromigration
Defects, Faults, Errors and Failures

- **Defect**: An unintended difference between the implemented hardware and its intended design
- **Fault**: A representation of a “defect” at abstracted functional level
  - May or may not cause a problem
- **Error**: Manifestation of a fault that results in incorrect circuit (system) outputs or states
  - Caused by faults
- **Failure**: Deviation of a circuit or system from its specified behavior
  - Fails to do what it should do
  - Caused by an error

Defect ---> Fault ---> Error ---> Failure
Defect, Fault, and Error

- Example
  - Defect: $b$ short to ground
  - Fault: signal $b$ stuck-at 0
  - Error: $a=1$, $b=1$, output $c=0$ (correct output $c=1$)
  - No Error when $a=0$ or $b=0$
Why Model Faults?

1. Identifies target faults
   - Model faults most likely to occur
2. Limits the scope of test generation
   - Create tests only for the modeled faults
3. Makes analysis possible
   - Associate specific defects with specific test patterns
4. Makes test effectiveness measurable by experiments
   - Fault coverage can be computed for specific test patterns to reflect its effectiveness
Fault Models

- Stuck-At Faults
- Bridging/break Faults
- Transistor Stuck-On/Open Faults
- Functional Faults
- Memory Faults
- Delay Faults
- Transition Faults
- State Transition Faults
Single Stuck-At Faults

Assumptions:

• Only one line is faulty. (Why?)
  • Faulty line permanently set to 0 or 1.
  • Fault can be at an input or output of a gate.
Single Stuck-at Faults

- # single stuck-at fault sites in a Boolean gate circuit
  \[= \#PI + \#\text{gates} + \# \text{(fanout branches)}\]
- Example: A 4-NAND XOR circuit has 12 fault sites (●) and 24 single stuck-at faults

How many faults?
Multiple Stuck-At Faults

- Several stuck-at faults occur at the same time
  - Important in high density circuits
- For a circuit with k lines
  - There are $2^k$ single stuck-at faults
  - There are $3^k-1$ multiple stuck-at faults
- ATPG algorithms for multiple s-a-faults are much more complex and not as well developed
Why Single Stuck-At Faults?

• Complexity is greatly reduced.
  Many different physical defects may be modeled by the same logical single stuck-at fault.

• Single stuck-at fault is technology independent.
  Can be applied to TTL, ECL, CMOS, etc.

• Single stuck-at fault is design-style independent.
  Gate Arrays, Standard Cell, Custom VLSI

• Even when single stuck-at fault does not accurately model some physical defects, the tests derived for these faults may still be effective for these defects.

• Single stuck-at tests cover a large percentage of multiple stuck-at faults.
Bridging Faults

- Two or more normally distinct points (lines) are shorted together
  - Logic effect depends on technology
  - Wired-AND for TTL
    
    \[ \begin{align*}
    \text{A} & \quad \text{B} \\
    \text{f} & \quad \text{g}
    \end{align*} \]
    \[ \Rightarrow \]
    
    \[ \begin{align*}
    \text{A} & \quad \text{B} \\
    \text{f} & \quad \text{g}
    \end{align*} \]
  - Wired-OR for ECL
    
    \[ \begin{align*}
    \text{A} & \quad \text{B} \\
    \text{f} & \quad \text{g}
    \end{align*} \]
    \[ \Rightarrow \]
    
    \[ \begin{align*}
    \text{A} & \quad \text{B} \\
    \text{f} & \quad \text{g}
    \end{align*} \]
  - CMOS?
Feedback Bridging Faults

- Can cause oscillation or latching (additional memory)
- Consequences:
  - The shorted signal lines form wired logic so the original logic function is changed
  - The circuit may become unstable if unwanted feedbacks exist
- Applying opposite values to the signal lines being tested to test these faults
CMOS Transistor Stuck-On

- Transistor stuck-on may cause ambiguous logic level.
  - depends on the relative impedances of the pull-up & pull-down networks
- When input is low, both P and N transistors are conducting causing increased quiescent current, called IDDQ fault.
CMOS Transistor Stuck-OPEN

- Transistor stuck-open may cause output floating
- Can turn the circuit into a sequential one (temporarily keep the previous value)
- Stuck-open faults require two-vector tests: initialization and test vectors

\[ ? = \text{previous state} \]

\[
\begin{align*}
0 & \quad \text{stuck-open} \\
? & \quad \text{01 / 00} \\
10 & \quad \text{memory behavior}
\end{align*}
\]
(Line) Break Faults

- Can be on the line between two gates or within one gate.
- Usually resulting in floating.
- May require two or more patterns to detect a break fault.
Functional Faults

- Fault effects modeled at a higher level than logic for function modules, such as

  -- Decoders
  -- Multiplexers
  -- Adders
  -- RAM
  -- ROM
  -- CPU (instruction set)
  -- Cache memory
Functional Faults of Decoder

\( f(L_i/L_j) \): Instead of line \( L_i \), Line \( L_j \) is selected
\( f(L_i/L_i+L_j) \): In addition to \( L_i \), \( L_j \) is selected
\( f(L_i/L_j+L_k) \): Instead of \( L_i \), \( L_j \) and \( L_k \) are selected
\( f(L_i/0) \): None of the lines are selected
Memory Faults

- **Parametric Faults**
  - Output Levels
  - Power Consumption
  - Noise Margin
  - Data Retention Time

- **Functional Faults**
  - Stuck-at Faults in Address Register, Data Register, and Address Decoder
  - Cell Stuck Faults
  - Adjacent Cell Coupling Faults
  - Pattern-Sensitive Faults
Memory Faults (Cont.)

- Pattern-sensitive faults: the presence of a faulty signal depends on the signal values of the nearby points
  - Most common in DRAMs

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>x</td>
<td>b</td>
</tr>
<tr>
<td>0</td>
<td>a</td>
<td>0</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\text{a=b=0} & \quad \rightarrow \quad x=0 \\
\text{a=b=1} & \quad \rightarrow \quad x=1
\end{align*}
\]

- Adjacent cell coupling faults
  - Pattern sensitivity between a pair of cells
Delay Fault Model

- **Assumption**
  - Some physical defects, such as process variations, make some delays in the CUT greater than some defined limits

- **Two delay fault models are typically used**
  - **Gate delay fault model** (a local delay fault model)
  - **Path delay fault model** (a global delay fault model)
Gate-Delay-Fault

- Slow to rise, slow to fall
  - \( \bar{x} \) is slow to rise when channel resistance R1 is abnormally high
Gate-Delay-Fault

- Disadvantage:

  Delay faults resulting from the sum of several small incremental delay defects may not be detected.
Path-Delay-Fault

• Propagation delay of the path exceeds the clock interval.

• Disadvantage:
  The number of paths grows exponentially with the number of gates.
Transition Fault Model

- Assumes a large/gross delay is present at a circuit node
- Irrespective of which path the effect is propagated, the gross delay defect will be late arriving at an observable point
- Most commonly used in industry for timing-related faults
  - Simple and number of faults linear to circuit size
  - Also needs 2 vectors to test
- Node x slow-to-rise (x-STR) can be modeled simply as two stuck-at faults
  - First time-frame: x/1 needs to be excited
  - Second time-frame: x/0 needs to be excited and propagated
Transition Fault Properties

- Lemma: a transition fault may be launched robustly, non-robustly, or neither
- Example: STR at output of OR gate
State Transition Graph

• Each state transition is associated with a 4-tuple: (source state, input, output, destination state)
A fault causes a single state transition to a wrong destination state.
Fault Detection

- A test (vector) \( t \) detects a fault \( f \) iff \( z(t) \not= z_f(t) = 1 \)
  - \( t \) detects \( f \) \( \iff \) \( z_f(t) \neq z(t) \)

- Example

\[
\begin{align*}
\text{The test 001 detects } f \text{ because } z_1(001) &= 0 \text{ while } z_{1f}(001) = 1
\end{align*}
\]
Sensitization

• A test $t$ that detects a fault $f$
  - Activates $f$ (or generate a fault effect) by creating different $v$ and $v_f$ values at the site of the fault
  - Propagates the error to a primary output $w$ by making all the lines along at least one path between the fault site and $w$ have different $v$ and $v_f$ values

• A line whose value in the test changes in the presence of the fault $f$ is said to be sensitized to the fault $f$ by the test

• A path composed of sensitized lines is called a sensitized path
Detectability

- A fault $f$ is said to be detectable if there exists a test $t$ that detects $f$; otherwise, $f$ is an undetectable fault.
- For an undetectable fault $f$, for all input $x$

$$z_f(x) = z(x)$$

- No test can simultaneously activate $f$ and create a sensitized path to a primary output.
Undetectable Fault

- **G₁ output stuck-at-0 fault is undetectable**
  - Undetectable faults do not change the function of circuit
  - The related circuit can be deleted to simplify the circuit
Undetectable Fault

- The presence of an undetectable fault $f$ may prevent the detection of another fault $g$, even when there exists a test which detects the fault $g$.

- Example:
  - A detectable fault $a$ s-a-0 becomes undetectable under the presence of a undetectable fault $c$ s-a-1.

In fact, $Z = AB + ABC = AB$
Thus the circuit can be simplified.

In general, identifying undetectable faults can lead to simplification of CUT
Test Set

- Complete detection test set: A set of tests that detect any detectable faults in a class of faults
- The quality of a test set is measured by fault coverage
- Fault coverage: Fraction of faults that are detected by a test set
- The fault coverage can be determined by fault simulation
  - >95% is typically required for single stuck-at fault model in a complex system such as a CPU
Fault collapsing

- Fault equivalence
- Fault dominance
- Checkpoint theory
Fault Equivalence

- A test $t$ distinguishes between faults $\alpha$ and $\beta$ if
  \[ z_\alpha(t) \neq z_\beta(t) \]

- Two faults, $\alpha$ & $\beta$ are said to be equivalent in a circuit, iff the function under $\alpha$ is equal to the function under $\beta$ for any input combination (sequence) of the circuit.
  - $z_\alpha(t) = z_\beta(t)$ for all $t$
  - No test can distinguish between $\alpha$ and $\beta$
  - Any test which detects one of them detects both faults
Fault Equivalence

- AND gate: all $s-a-0$ faults are equivalent
- OR gate: all $s-a-1$ faults are equivalent
- NAND gate: all the input $s-a-0$ faults and the output $s-a-1$ faults are equivalent
- NOR gate: all input $s-a-1$ faults and the output $s-a-0$ faults are equivalent
- Inverter: input $s-a-1$ and output $s-a-0$ are equivalent
  input $s-a-0$ and output $s-a-1$ are equivalent

Example: Three faults shown are equivalent
Equivalence Fault Collapsing

- $n+2$ instead of $2n+2$ faults need to be considered for an $n$-input gate.
Fault Dominance

- A fault $\beta$ is said to dominate another fault $\alpha$ in an irredundant circuit, iff every test (sequence) for $\alpha$ is also a test (sequence) for $\beta$.
- Notation: $\beta \rightarrow \alpha$
  - No need to consider fault $\beta$ for fault detection
- When two faults $f_1$ and $f_2$ dominate each other, then they are equivalent

- Detect A $sa1$:
  $$z(t) \oplus z_f(t) = (CD + CE) \oplus (D + CE) = 1 \implies (C = 0, D = 1)$$
- Detect C $sa1$:
  $$z(t) \oplus z_f(t) = (CD + CE) \oplus (D + E) = 1 \implies (C = 0, D = 1) \text{ or } (C = 0, E = 1)$$
- Similarly, C $sa1 \rightarrow A sa1$, C $sa1 \rightarrow B sa1$, C $sa0 \rightarrow A sa0$, C $sa0 \rightarrow B sa0$
Fault Dominance

- AND gate: Output s-a-1 dominates any input s-a-1
- NAND gate: Output s-a-0 dominates any input s-a-1
- OR gate: Output s-a-0 dominates any input s-a-0
- NOR gate: Output s-a-1 dominates any input s-a-0
- Dominance fault collapsing: The reduction of the set of faults to be analyzed based on dominance relation
Fault Dominance

• **Detect A sa1:** \[ z(t) \oplus z_f(t) = (CD \oplus CE) \oplus (D \oplus CE) = D \oplus CD = 1 \]
  \[ \Rightarrow (C = 0, D = 1) \]

• **Detect C sa1:** \[ z(t) \oplus z_f(t) = (CD \oplus CE) \oplus (D \oplus E) = 1 \]
  \[ \Rightarrow (C = 0, D = 1) \text{ or } (C = 0, E = 1) \]

• **Similarly**
  - C sa1 → A sa1
  - C sa1 → B sa1
  - C sa0 → A sa0
  - C sa0 → B sa0
Fault Collapsing

- For each $n$-input gate, we only need to consider $n+1$ faults.
**Prime Faults**

- $\alpha$ is a prime fault if every fault that is dominated by $\alpha$ is also equivalent to $\alpha$

  - **Representative Set of Prime Faults (RSPF)**
    - A set that consists of exactly one prime fault from each equivalence class of prime faults
    - True minimal RSPF is difficult to find
Why Fault Collapsing?

- Memory & CPU-Time saving

To ease the burden for test generation and fault simulation in testing

<table>
<thead>
<tr>
<th># of total faults</th>
<th># of equivalent faults</th>
<th># of prime faults</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>60%</td>
<td>40%</td>
</tr>
</tbody>
</table>
Fault Collapsing for Combinational Circuit

- 30 total faults $\Rightarrow$ 12 prime faults
Checkpoint Theorem

- Primary-input & Fanout-Branches

\( \Rightarrow \) a sufficient and necessary set of checkpoints in irredundant combinational circuits

- In fanout-free combinational circuits, primary inputs are the set of checkpoints

- Any test set which detects all signal (multiple) stuck faults on checkpoints will detect all signal (multiple) stuck faults
Fault Collapsing

- The set of checkpoint faults can be further collapsed by using equivalence and dominance relation
- Example

- 10 checkpoint faults
- a s-a-0 ↔ d s-a-0, c s-a-0 ↔ e s-a-0
  b s-a-0 → d s-a-0, b s-a-1 → d s-a-1
- 6 faults are enough