Chapter 1

Introduction
What is this chapter about?

- Introduce fundamental concepts and various aspects of VLSI testing

- Focus on
  - Importance of testing in the design and manufacturing processes
  - Challenges in test generation and fault modeling
  - Levels of abstraction in VLSI testing

- Provide overview of VLSI test technology
Introduction to VLSI Testing

- Introduction
- Testing During VLSI Life Cycle
- Test Generation
- Fault Models
- Levels of Abstraction
- Overview of Test Technology
- Concluding Remarks
Introduction

- Integrated Circuits (ICs) have grown in size and complexity since the late 1950’s
  - Small Scale Integration (SSI)
  - Medium Scale Integration (MSI)
  - Large Scale Integration (LSI)
  - Very Large Scale Integration (VLSI)

- Moore’s Law: scale of ICs doubles every 18 months
  - Growing size and complexity poses many and new testing challenges
Importance of Testing

- Moore’s Law results from decreasing feature size (dimensions)
  - from 10s of µm to 10s of nm for transistors and interconnecting wires
- Operating frequencies have increased from 100KHz to several GHz
- Decreasing feature size increases probability of defects during manufacturing process
  - A single faulty transistor or wire results in faulty IC
  - Testing required to guarantee fault-free products
Importance of Testing

- Rule of Ten: cost to detect faulty IC increases by an order of magnitude as we move from:
  - device → PCB → system → field operation
    - Testing performed at all of these levels
- Testing also used during
  - Manufacturing to improve yield
    - Failure mode analysis (FMA)
  - Field operation to ensure fault-free system operation
    - Initiate repairs when faults are detected
Testing typically consists of

- Applying set of test stimuli to
- Inputs of *circuit under test* (CUT), and
- Analyzing output responses
  - If incorrect (fail), CUT assumed to be faulty
  - If correct (pass), CUT assumed to be fault-free
Testing During VLSI Development

- Design verification targets design errors
  - Corrections made prior to fabrication
- Remaining tests target manufacturing defects
  - A defect is a flaw or physical imperfection that can lead to a fault

VLSI Test Principles and Architectures
**Design Verification**

- Different levels of abstraction during design
  - CAD tools used to synthesize design from RTL to physical level
- Simulation used at various level to test for
  - Design errors in behavioral or RTL
  - Design meeting system timing requirements after synthesis
Yield and Reject Rate

- We expect faulty chips due to manufacturing defects
  - Called yield

- 2 types of yield loss
  - Catastrophic – due to random defects
  - Parametric – due to process variations

- Undesirable results during testing
  - Faulty chip appears to be good (passes test)
    - Called reject rate
  - Good chip appears to be faulty (fails test)
    - Due to poorly designed tests or lack of DFT

\[
yield = \frac{\text{number of acceptable parts}}{\text{total number of parts fabricated}}
\]

\[
\text{reject rate} = \frac{\text{number of faulty parts passing final test}}{\text{total number of parts passing final test}}
\]
Electronic System Manufacturing

- A system consists of
  - PCBs that consist of
    - VLSI devices
- PCB fabrication similar to VLSI fabrication
  - Susceptible to defects
- Assembly steps also susceptible to defects
  - Testing performed at all stages of manufacturing
System-Level Operation

- Faults occur during system operation
- Exponential failure law
  - Interval of normal system operation is random number exponentially distributed
- Reliability
  - Probability that system will operate normally until time $t$
  - Failure rate, $\lambda$, is sum of individual component failure rates, $\lambda_i$

$$P(T_n > t) = e^{-\lambda t}$$

$$\lambda = \sum_{i=0}^{k} \lambda_i$$
System-Level Operation

- Mean Time Between Failures (MTBF)
- Repair time (R) also assumed to obey exponential distribution
  - $\mu$ is repair rate
- Mean Time To Repair (MTTR)
- Fraction of time that system is operating normally called system availability
  - High reliability systems have system availabilities greater than 0.9999
    - Referred to as “four 9s”

\[
MTBF = \int_{0}^{\infty} e^{-\lambda t} dt = \frac{1}{\lambda} \\
P(R > t) = e^{-\mu t} \\
MTTR = \frac{1}{\mu} \\
\text{system availability} = \frac{MTBF}{MTBF + MTTR}
\]
System-Level Testing

- Testing required to ensure system availability
- Types of system-level testing
  - On-line testing – concurrent with system operation
  - Off-line testing – while system (or portion of) is taken out of service
    - Performed periodically during low-demand periods
    - Used for diagnosis (identification and location) of faulty replaceable components to improve repair time
Test Generation

- A test is a sequence of test patterns, called test vectors, applied to the CUT whose outputs are monitored and analyzed for the correct response
  - Exhaustive testing – applying all possible test patterns to CUT
  - Functional testing – testing every truth table entry for a combinational logic CUT
    - Neither of these are practical for large CUTs
- Fault coverage is a quantitative measure of quality of a set of test vectors
Test Generation

- Fault coverage for a given set of test vectors
  \[ \text{fault coverage} = \frac{\text{number of detected faults}}{\text{total number of faults}} \]

- 100% fault coverage may be impossible due to undetectable faults

\[ \text{fault detection efficiency} = \frac{\text{number of detected faults}}{\text{total number of faults} - \text{number of undetectable faults}} \]

- Reject rate = 1 − yield \(1 - \text{fault coverage}\)
  - A PCB with 40 chips, each with 90% fault coverage and 90% yield, has a reject rate of 41.9%
    - Or 419,000 defective parts per million (PPM)
Test Generation

- **Goal**: find efficient set of test vectors with maximum fault coverage
- Fault simulation used to determine fault coverage
  - Requires fault models to emulate behavior of defects
- A good fault model:
  - Is computationally efficient for simulation
  - Accurately reflects behavior of defects
- No single fault model works for all possible defects
Fault Models

- A given fault model has $k$ types of faults
  - $k = 2$ for most fault models
- A given circuit has $n$ possible fault sites
- Multiple fault model – circuit can have multiple faults (including single faults)
  - Number of multiple fault = $(k+1)^n - 1$
    - Each fault site can have 1-of-$k$ fault types or be fault-free
    - The “-1” represents the fault-free circuit
  - Impractical for anything but very small circuits
- Single fault model – circuit has only 1 fault
  - Number of single faults = $k \times n$
  - Good single fault coverage generally implies good multiple fault coverage
Fault Models

- Equivalent faults
  - One or more single faults that have identical behavior for all possible input patterns
  - Only one fault from a set of equivalent faults needs to be simulated

- Fault collapsing
  - Removing equivalent faults
    - Except for one to be simulated
  - Reduces total number of faults
    - Reduces fault simulation time
    - Reduces test pattern generation time
Stuck-at Faults

- Any line can be
  - Stuck-at-0 (SA0)
  - Stuck-at-1 (SA1)
    # fault types: $k=2$
- Example circuit:
  - # fault sites: $n=9$
  - # single faults = $2 \times 9 = 18$

Truth table for fault-free behavior and behavior of all possible stuck-at faults

<table>
<thead>
<tr>
<th>$x_1 x_2 x_3$</th>
<th>000</th>
<th>001</th>
<th>010</th>
<th>011</th>
<th>100</th>
<th>101</th>
<th>110</th>
<th>111</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>a SA0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a SA1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>b SA0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>b SA1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>c SA0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>c SA1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>d SA0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d SA1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>e SA0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>e SA1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>f SA0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>f SA1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>g SA0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>g SA1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>h SA0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>h SA1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>i SA0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>i SA1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

VLSI Test Principles and Architectures

Ch. 1 - Introduction - P. 20
Stuck-at Faults

- Valid test vectors
  - Faulty circuit differs from good circuit
  - Necessary vectors:
    - 011 detects f SA1, e SA0
    - 100 detects d SA1
    - Detect total of 10 faults
    - 001 and 110 detect remaining 8 faults

<table>
<thead>
<tr>
<th>$x_1x_2x_3$</th>
<th>000</th>
<th>001</th>
<th>010</th>
<th>011</th>
<th>100</th>
<th>101</th>
<th>110</th>
<th>111</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>a SA0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a SA1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>b SA0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>b SA1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>c SA0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>c SA1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>d SA0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d SA1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>e SA0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>e SA1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>f SA0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>f SA1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>g SA0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>g SA1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>h SA0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>h SA1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>i SA0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>i SA1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Stuck-at Faults

- 4 sets of equivalent faults
- \# collapsed faults = $2 \times (P_O + F_O) + G_I - N_I$
  - $P_O$ = \# primary outputs
  - $F_O$ = \# fanout stems
  - $G_I$ = \# gate inputs
  - $N_I$ = \# inverters

Truth table for fault-free behavior
and behavior of all possible stuck-at faults

<table>
<thead>
<tr>
<th>$x_1 x_2 x_3$</th>
<th>000</th>
<th>001</th>
<th>010</th>
<th>011</th>
<th>100</th>
<th>101</th>
<th>110</th>
<th>111</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>a SA0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a SA1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>b SA0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>b SA1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>c SA0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>c SA1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>d SA0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d SA1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>e SA0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>e SA1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>f SA0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>f SA1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>g SA0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>g SA1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>h SA0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>h SA1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>i SA0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>i SA1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Stuck-at Faults

- # collapsed faults = $2 \times (P_O + F_O) + G_I - N_I$
  - $P_O$ = number of primary outputs
  - $F_O$ = number of fanout stems
  - $G_I$ = total number of gate inputs for all gates including inverters
  - $N_I$ = total number of inverters

- For example circuit, # collapsed faults = 10
  - $P_O$ = 1, $F_O$ = 1, $G_I$ = 7, and $N_I$ = 1

- Fault collapsing typically reduces number of stuck-at faults by 50% - 60%
Transistor Faults

- Any transistor can be
  - Stuck-short
    - Also known as stuck-short
  - Stuck-open
    - Also known as stuck-open
  
- Example circuit
  - # fault sites: $n=4$
  - # single faults = $2 \times 4 = 8$

<table>
<thead>
<tr>
<th>$AB$</th>
<th>00</th>
<th>01</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Z$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$N_1$ stuck-open</td>
<td>1</td>
<td>0</td>
<td>last Z</td>
<td>0</td>
</tr>
<tr>
<td>$N_1$ stuck-short</td>
<td>$I_{DDQ}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$N_2$ stuck-open</td>
<td>1</td>
<td>last Z</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$N_2$ stuck-short</td>
<td>$I_{DDQ}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$P_1$ stuck-open</td>
<td>last Z</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$P_1$ stuck-short</td>
<td>1</td>
<td>0</td>
<td>$I_{DDQ}$</td>
<td>0</td>
</tr>
<tr>
<td>$P_2$ stuck-open</td>
<td>last Z</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$P_2$ stuck-short</td>
<td>1</td>
<td>$I_{DDQ}$</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
**Transistor Faults**

- Stuck-short faults cause conducting path from $V_{DD}$ to $V_{SS}$
  - Can be detected by monitoring steady-state power supply current $I_{DDQ}$
- Stuck-open faults cause output node to store last voltage level
  - Requires sequence of 2 vectors for detection
    - 00→10 detects $N_1$ stuck-open

---

2-input CMOS NOR gate

<table>
<thead>
<tr>
<th>$AB$</th>
<th>00</th>
<th>01</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Z$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$N_1$ stuck-open</td>
<td>1</td>
<td>0</td>
<td>last $Z$</td>
<td>0</td>
</tr>
<tr>
<td>$N_1$ stuck-short</td>
<td>$I_{DDQ}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$N_2$ stuck-open</td>
<td>1</td>
<td>last $Z$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$N_2$ stuck-short</td>
<td>$I_{DDQ}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$P_1$ stuck-open</td>
<td>last $Z$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$P_1$ stuck-short</td>
<td>1</td>
<td>0</td>
<td>$I_{DDQ}$</td>
<td>0</td>
</tr>
<tr>
<td>$P_2$ stuck-open</td>
<td>last $Z$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$P_2$ stuck-short</td>
<td>1</td>
<td>$I_{DDQ}$</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Transistor Faults

- # collapsed faults = \(2 \times T - T_S + G_S - T_P + G_P\)
  - \(T\) = number of transistors
  - \(T_S\) = number of series transistors
  - \(G_S\) = number of groups of series transistors
  - \(T_P\) = number of parallel transistors
  - \(G_P\) = number of groups of parallel transistors

- For example circuit, # collapsed faults = 6
  - \(T=4, T_S = 2, G_S = 1, T_P = 2, \) & \(G_P = 1\)

- Fault collapsing typically reduces number of transistor faults by 25% to 35%
**Shorts and Opens**

- **Wires can be**
  - **Open**
    - Opens in wires interconnecting transistors to form gates behave like transistor stuck-open faults
    - Opens in wires interconnecting gates to form circuit behave like stuck-at faults
    - Opens are detected by vectors detecting transistor and stuck-at faults
  - **Short to an adjacent wire**
    - Also known as a bridging fault
Bridging Faults

- Three different models
  - Wired-AND/OR
  - Dominant
  - Dominant-AND/OR

- Detectable by $I_{DDQ}$ testing
Delay Faults and Crosstalk

- Path-delay fault model considers cumulative propagation delay through CUT
  - 2 test vectors create transition along path
  - Faulty circuit has excessive delay
- Delays and glitches can be caused by crosstalk between interconnect
  - due to inductance and capacitive coupling
Pattern Sensitivity and Coupling Faults

- Common in high density RAMs
- Pattern sensitivity fault
  - Contents of memory cell is affected by contents of neighboring cells
- Coupling fault
  - Transition in contents of one memory cell causes change in contents of another cell
Pattern Sensitivity and Coupling Faults

- Common in memory cells of high density RAMs
- Pattern sensitivity fault
  - Contents of cell affected by contents of neighboring cells
- Coupling fault
  - Transition in one cell causes change in another cell
- Detected with specific memory test algorithms
  - Background Data Sequence (BDS) used for word-oriented memories

**Notation:**
- \(w_0\) = write 0 (or all 0’s)
- \(r_1\) = read 1 (or all 1’s)
- ↑ = address up
- ↓ = address down
- ⇢ = address either way

<table>
<thead>
<tr>
<th>Test Algorithm</th>
<th>March Test Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>March LR w/o BDS</td>
<td>⇢(w0); ↓(r0, w1); ↑(r1, w0, r0, r0, w1); ↑(r1, w0); ↑(r0, w1, r1, r1, w0); ↑(r0)</td>
</tr>
<tr>
<td>March LR with BDS</td>
<td>⇢(w00); ↓(r00, w11); ↑(r11, w00, r00, r00, w11); ↑(r11, w00); ↑(r00, w11, r11, r11, w00); ↑(r00, w01, w10, r10); ↑(r10, w01, r01); ↑(r01)</td>
</tr>
</tbody>
</table>
Analog Fault Models

- Catastrophic faults
  - Shorts and opens
- Parametric faults
  - Parametric variations in passive and active components cause components to be out of tolerance range
- Active components can sustain defects that affect DC and/or AC operation
Levels of Abstraction

- High levels have few implementation details needed for effective test generation
  - Fault models based on gate & physical levels

Example: two circuits for same specification

- Ckt B test vectors do not detect 4 faults in Ckt A

\[
f(a,b,c) = \sum_m (1,7) + d(3) = ab\overline{c} + abc + X\overline{abc}
\]

\[
f = abc + \overline{ab\overline{c}}
\]

Test Vectors
\{111,110,101,011,010,000\}

\[
f = ab + b\overline{c}
\]

Test Vectors
\{111,101,010,000\}

VLSI Test Principles and Architectures
Overview of VLSI Test Technology

Automatic Test Equipment (ATE) consists of

- Computer – for central control and flexible test & measurement for different products
- Pin electronics & fixtures – to apply test patterns to pins & sample responses
- Test program – controls timing of test patterns & compares response to known good responses
Overview of VLSI Test Technology

- Automatic Test Pattern Generation (ATPG)
  - Algorithms generating sequence of test vectors for a given circuit based on specific fault models

- Fault simulation
  - Emulates fault models in CUT and applies test vectors to determine fault coverage
  - Simulation time (significant due to large number of faults to emulate) can be reduced by
    - Parallel, deductive, and concurrent fault simulation
Overview of VLSI Test Technology

- Design for Testability (DFT)
  - Generally incorporated in design
  - Goal: improve controllability and/or observability of internal nodes of a chip or PCB

- Three basic approaches
  - Ad-hoc techniques
  - Scan design
    - Boundary Scan
  - Built-In Self-Test (BIST)
Design of Testability

- Ad-hoc DFT techniques
  - Add internal test points (usually multiplexers) for
    - Controllability
    - Observability
  - Added on a case-by-case basis
    - Primarily targets “hard to test” portions of chip

---

VLSI Test Principles and Architectures

Ch. 1 - Introduction - P. 37
Design for Testability

- **Scan design**
  - Transforms flip-flops of chip into a shift register
  - Scan mode facilitates
    - Shifting in test vectors
    - Shifting out responses

- **Good CAD tool support**
  - Transforming flip-flops to shift register
  - ATPG
Design for Testability

- Boundary Scan – scan design applied to I/O buffers of chip
  - Used for testing interconnect on PCB
    - Provides access to internal DFT capabilities
  - IEEE standard 4-wire Test Access Port (TAP)

<table>
<thead>
<tr>
<th>TAP pin</th>
<th>I/O</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCK</td>
<td>input</td>
<td>Test clock</td>
</tr>
<tr>
<td>TMS</td>
<td>input</td>
<td>Test Mode Select</td>
</tr>
<tr>
<td>TDI</td>
<td>input</td>
<td>Test Data In</td>
</tr>
<tr>
<td>TDO</td>
<td>output</td>
<td>Test Data Out</td>
</tr>
</tbody>
</table>
Design for Testability

- Built-In Self-Test (BIST)
  - Incorporates test pattern generator (TPG) and output response analyzer (ORA) internal to design
    - Chip can test itself
  - Can be used at all levels of testing
    - Device → PCB → system → field operation
Concluding Remarks

- Many new testing challenges presented by
  - Increasing size and complexity of VLSI devices
  - Decreasing feature size

- This chapter presented introduction to VLSI testing

- Remaining chapters present more details as well as solutions to these challenges