Fault Simulation

Sungho Kang

Yonsei University
Outline

- Introduction
- Parallel
- Deductive
- Concurrent
- PPSFP
- Critical Path Tracing
- PROOFS
- PARIS
- Fault Sampling
- Statistical Fault Analysis
- Hardware Acceleration
- Hierarchical Fault Simulation
- Conclusion
Fault Simulation

Introduction

Fault Simulation

Fault-free circuit

Faulty circuit

Test Application

Test Set

Fault List

Fault insertion

POs

PIs

+ 1 0

Detected

Undetected

Summary

Fault dictionary

Simulation Output

Test Program

Simulation Output
Application

- Evaluating Test Vectors
- Generating Tests
- Constructing Fault Dictionaries
- Analyzing Circuits under Presence of Faults
Application: Evaluating Test Vectors

- **Fault Coverage**
  - Ratio of the number of faults detected by a test vector to the number of total faults
  - Represents approximation of defect coverage which is the probability that test vectors detect physical defects

\[
\text{Defect Level} = \frac{BP}{GP + BP}
\]
Application: Evaluating Test Vectors

Introduction

- Quality of Test
  - \( Y \): Yield
  - \( DL \): Defect Level
  - \( d \): defect coverage
  - \( DL = 1 - Y \cdot 1^{d} \)

- Consider a 0.5 yield process
  - To achieve 0.01 defect level, 99% coverage is required
  - To achieve 80% coverage, 0.2 defect level
Application: Generating Tests

- Use fault simulation first
- For remaining faults, use deterministic algorithm

![Graph]

- **I**: Functional or Random Patterns
- **II**: Deterministic Patterns
Application: Fault Dictionaries

- **Fault Dictionary**
  - Stores the output response of every faulty circuit corresponding to a fault
  - Stores the signature

- **Computing response before testing**

- **Post-test Diagnosis**
  - Isolating a reduced set of plausible fault
  - Simulating only these faults to identify the actual fault
Application: Analyzing Circuits

- Important in high-reliability systems
- A fault can induce races and hazards not present in the fault-free circuit
- A faulty circuit may oscillate or enter a deadlock state
- A fault can inhibit the proper initialization of a sequential circuit
- A fault can transform a combinational circuit into a sequential circuit or a synchronous into asynchronous
Basic Fault Simulation

- Fault Specification
- Fault Insertion
- Fault Propagation
- Fault Detection
- Post Processing
Fault Specification

• This is the activity which determines which faults and fault types will be modeled during simulation
• Normally the total set of faults to be considered is determined before any simulation takes place and is the first fault-related activity to occur
• Later certain techniques or heuristics can be used to reduce this fault set but the resulting fault set is a subset of the one that is initially
Fault Insertion

- Fault insertion in a fault simulation system is very analogous to the process of physical fault insertion for a fabricated machine.
- It can also be termed a fault transformation, where the fault free network is transformed into a network which contains a physical defect or fault.
- This process can occur once for a given fault, before fault simulation begins, or it can occur numerous times during a fault simulation process, for each fault that is being modeled.
A fault simulation system obviously must have the ability to propagate the effects of a fault.

- **Stuck-at-0** fault propagation:
  - $b=1$ → fault propagation to $c$
  - $b=0$ → fault is not propagated
Fault Propagation

- (a) Fault is propagated
- (b) Fault is propagated through multiple paths
- (c) Fault is blocked
Fault Detection

- A fault simulation system must have the ability to determine when a fault has become observable at some given point or primary output of the object network.
- Whether this activity takes place concurrently with the actual simulation or at the end of a fault simulation pass depends on other attributes of the simulator.
Postprocessing

- The fault simulation of a given digital network can result in a large amount of data which the user must interpret and efficiently utilize.
- In an attempt to aid the user in this process one of the last activities which a fault simulation system performs is to organize the resulting data in a way which makes it most amenable to the user.
- This activity can actually take place as a stand alone process.
Serial Fault Simulation

- Simplest
- Consider fault one at a time
- No special fault simulator is required
- Can handle any types of faults
- Impractical
  - Excessive amount of CPU time
Parallel Fault Simulation

- A good circuit and w faulty circuits are simulated simultaneously passes are required
  - w : number of bits of the host machine

Word 1

<table>
<thead>
<tr>
<th>W-1</th>
<th>· · ·</th>
<th>0</th>
</tr>
</thead>
</table>

Bit 0 = Fault-Free Value
Bit 1 ~ (W-1) = Faulty Value

<table>
<thead>
<tr>
<th></th>
<th>g</th>
<th>f₁</th>
<th>f₂</th>
<th>f₃</th>
<th>f₄</th>
<th>f₅</th>
<th>f₆</th>
<th>f₇</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>a·b</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Parallel Fault Simulation

- One good machine

Test Pattern

Good Circuit

Circuit with Fault 1

Circuit with Fault 2

Circuit with Fault n
Fault Insertion

- 2 input AND

**Fault List**

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>s-a-1</th>
<th>b</th>
<th>s-a-1</th>
<th>c</th>
<th>s-a-0</th>
<th>c</th>
<th>s-a-1</th>
</tr>
</thead>
</table>

**Fault Insertion**

```
<table>
<thead>
<tr>
<th></th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td></td>
<td></td>
<td>c</td>
<td>s-a-1</td>
<td>c</td>
<td>s-a-0</td>
<td>b</td>
<td>s-a-1</td>
</tr>
<tr>
<td>b</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>a</td>
<td>s-a-1</td>
</tr>
<tr>
<td>c</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Good</td>
</tr>
</tbody>
</table>
```

Unused

**Fault Detection**

```
a = 0, b = 1 simulation

<table>
<thead>
<tr>
<th></th>
<th>7</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>c</td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
```
Parallel Fault Simulation

- Apply all vectors against one group of faults

  generate a fault list
  for all faults which are not detected
    for all patterns
      initialize the circuit
      choose a pattern which is not simulated
      select 31 faults which are not detected
      fault simulation with fault insertion
      fault detection
    end
  end
end
Parallel Fault Simulation

- Apply one vector against all faults

  generate a fault list
  for all patterns
    choose a pattern which is not simulated
    initialize the circuit
    for all faults which are not detected
      select 31 faults which are not detected
      fault simulation with fault insertion
      fault detection
    end
  end
end
Parallel Fault Simulation

- Possible to reduce the number of passes by simulating several independent faults simultaneously
  - Independent faults can be simulated in the same bit
  - Least upper bound is $p$ number of POs
- $F/(Wp)$ passes are required
- Limitations
  - Requires Boolean equation models
    - Logical operations only
  - Complex for multi-valued logic
  - Cannot use selective trace
    - Fault dropping is not effective
Deductive Fault Simulation

- Simulates a good circuit and deduce all faulty circuits
- Only a few faults may produce values different from fault-free values
- Keep fault-free values and differences

- Parallel
  - 0 1 2 3 4 5 6 7
  - 1 1 1 1 0 1 0 1

- Deductive
  - {4,6}
Concurrent Fault Simulation

- Simulates good circuit and simulates only faulty circuits that are different from the one in good circuit
- Concurrent Fault List
- Directly applicable to functional level modeling
- Requires lots of memory
Concurrent Fault Simulation

- Explicit simulation of fault-free and faulty circuits as in serial
- Keep differences between faulty and fault-free as in deductive
Algorithm Features

- Separate fault-free and faulty events
- Independent evaluations
- Keep only faulty states that are different from fault free states: delete others (convergence)
- Create faulty states when they become different from fault-free (divergence)
Concurrent Fault Simulation

<table>
<thead>
<tr>
<th>A</th>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fault-free</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>a, s-a-1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>b, s-a-1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>c, s-a-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a, b, c : FO</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B</th>
<th>c1</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fault-free</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>d, s-a-1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>c1, s-a-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>e, s-a-0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c, s-a-0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c1, d, e : FO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c : FE</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C</th>
<th>e1</th>
<th>f</th>
<th>g</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fault-free</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>e1, s-a-0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>f, s-a-1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>g, s-a-1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c, s-a-0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>c1, s-a-0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>e, s-a-0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>e1, f, g : FO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c, c1, e : FE</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Concurrent Fault Simulation

Concurrent

A

B

c1

e1

c2

e

d

C

g

\begin{tabular}{|c|c|c|c|}
\hline
A & a & b & c \\
\hline
Fault-free & 1 & 0 & 1 \\
\hline
a & s-a-0 & 0 & 0 & 1 \\
b & s-a-1 & 1 & 1 & 0 \\
c & s-a-0 & 1 & 0 & 0 \\
\hline
a, b, c : FO \\
\end{tabular}

\begin{tabular}{|c|c|c|c|}
\hline
B & c1 & d & e \\
\hline
Fault-free & 1 & 0 & 1 \\
\hline
d & s-a-1 & 1 & 1 & 1 \\
c1 & s-a-1 & 0 & 0 & 0 \\
e & s-a-0 & 1 & 0 & 0 \\
b & s-a-1 & 0 & 0 & 0 \\
c & s-a-0 & 0 & 0 & 0 \\
\hline
d, c1, e : FO \\
b, c : FE \\
\end{tabular}

\begin{tabular}{|c|c|c|}
\hline
C & e1 & f & g \\
\hline
Fault-free & 1 & 1 & 0 \\
\hline
e1 & s-a-0 & 0 & 1 & 1 \\
f & s-a-0 & 1 & 0 & 1 \\
g & s-a-1 & 1 & 1 & 1 \\
b & s-a-1 & 0 & 1 & 1 \\
e & s-a-0 & 0 & 1 & 1 \\
c & s-a-0 & 0 & 1 & 1 \\
c1 & s-a-1 & 0 & 1 & 1 \\
\hline
e1, f, g : FO \\
b, e, c, c1 : FE \\
\end{tabular}
## Comparison of Three Methods

<table>
<thead>
<tr>
<th></th>
<th>Par</th>
<th>Ded</th>
<th>Conc</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Values</strong></td>
<td>binary</td>
<td>binary</td>
<td>any</td>
</tr>
<tr>
<td><strong>Elements</strong></td>
<td>logic</td>
<td>logic</td>
<td>any</td>
</tr>
<tr>
<td><strong>Storage</strong></td>
<td>min / fixed</td>
<td>med / variable</td>
<td>max / variable</td>
</tr>
<tr>
<td><strong>Delays</strong></td>
<td>restricted</td>
<td>restricted</td>
<td>any</td>
</tr>
<tr>
<td><strong>Fault-dropping</strong></td>
<td>not useful</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>
PPSFP

• Parallel Pattern Single Fault Propagation
  ▪ Single fault propagation
  ▪ Parallel pattern evaluation

• Advantage
  ▪ Simple and efficient.
  ▪ Selective trace: Single fault propagation
  ▪ Signals whose values are identical in the fault free circuit and the faulty circuit are not recalculated.

• Limitation
  ▪ Fails to consider the timing environment.
  ▪ Since it does not work like event driven simulation, it is limited to devices which are combinational.
  ▪ Faults which can produce races or hazards, which can make a circuit oscillate, and which can change a combinational circuit into a sequential one, require additional considerations.
Example
- Fault Free Circuit Evaluation

Consider C s-a-0 fault

Pattern 001 or 011 can detect the fault
Example

- Consider B s-a-0 fault

- After D evaluation, simulation stops since the result is the same as the fault free evaluation
- Try other patterns
PPSFP Algorithm

- for all patterns
  - choose 32 patterns which are not simulated
  - simulate a good circuit
  - record the values
- for all faults which are not detected
  - select a fault
  - simulate a circuit under this fault
    - if the value is the same as the value in the previous fault-free simulation after fault insertion
      - stop simulation
    - else continue
  - detect a fault at primary outputs
    - if the fault is detected
      - delete the fault from the fault list
  - end
- end
Critical Path Tracing

- A line L has a critical value v in the test T iff T detects the fault L s-a-v
- A line with a critical value in T is said to be critical in T
- A gate input is sensitive if complementing its value changes the value of the gate output
- If only one input j has the controlling value (c) of the gate then j is sensitive
- If all inputs have c’ then all inputs are sensitive
- Otherwise no input is sensitive
- If a gate output is critical then its sensitive inputs, if any, are also critical
Critical Path Tracing

- Let $T$ be a test that activates fault $f$ in a single output combinational circuit.
- Let $y$ be a line with level $l_y$, sensitized to $f$ by $T$.
- If every path sensitized to $f$ either goes through $y$ or does not reach any line with level greater than $l_y$, then $y$ is a capture line (surrogate line) of $f$ in the test $T$. 


Critical Path Tracing

- **FFR**
  - stops at FFR inputs and collects all the stems reached in the set `stems_to_check`

```
extend(j)
  mark j as critical
  if j is fanout branch
    add stem(j) to stems_to_check
  else
    critical(j)
  for every input k of j
    if sensitive(k)
      extend(k)
```
Critical Path Tracing

Stem Analysis
- determine whether stem j is critical by a breadth-first propagation

```plaintext
critical(j)
    frontier = { fanout of j }
    repeat
        k = lowest level gate in frontier
        remove k from frontier
        if frontier != NULL
            if propagates(k)
                add fanouts of k to frontier
            else critical(j)
        else critical(j)
            if propagates(k) and k is critical
                return TRUE
            else
                return FALSE
```
Critical Path Tracing

- Directly identifies the fault detected by a test
- Deals with faults only implicitly
- Based on a path tracing algorithm
- Approximate method
- Faster
- Less memory