Fault Simulation

Virendra Singh
Associate Professor
Computer Architecture and Dependable Systems Lab
Dept. of Electrical Engineering
Indian Institute of Technology Bombay
viren@ee.iitb.ac.in

EE 709: Testing & Verification of VLSI Circuits
Lecture – 8 (Jan 19, 2012)
Fault Model - 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.
Fault Simulation
Simulation Defined

• Definition: Simulation refers to modeling of a design, its function and performance.

• A software simulator is a computer program; an emulator is a hardware simulator.

• Simulation is used for design verification:
  • Validate assumptions
  • Verify logic
  • Verify performance (timing)

• Types of simulation:
  • Logic or switch level
  • Timing
  • Circuit
  • Fault
Simulation for Verification

- Specification
- Design (netlist)
- True-value simulation
- Input stimuli
- Computed responses
- Response analysis
- Design changes

Synthesis
## Modeling Levels

<table>
<thead>
<tr>
<th>Modeling level</th>
<th>Circuit description</th>
<th>Signal values</th>
<th>Timing</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function, behavior, RTL</td>
<td>Programming language-like HDL</td>
<td>0, 1</td>
<td>Clock boundary</td>
<td>Architectural and functional verification</td>
</tr>
<tr>
<td>Logic</td>
<td>Connectivity of Boolean gates, flip-flops and transistors</td>
<td>0, 1, X, and Z</td>
<td>Zero-delay, unit-delay, multiple-delay</td>
<td>Logic verification and test</td>
</tr>
<tr>
<td>Switch</td>
<td>Transistor size and connectivity, node capacitances</td>
<td>0, 1, X</td>
<td>Zero-delay</td>
<td>Logic verification</td>
</tr>
<tr>
<td>Timing</td>
<td>Transistor technology data, connectivity, node capacitances</td>
<td>Analog voltage</td>
<td>Fine-grain timing</td>
<td>Timing verification</td>
</tr>
<tr>
<td>Circuit</td>
<td>Tech. Data, active/passive component connectivity</td>
<td>Analog voltage, current</td>
<td>Continuous time</td>
<td>Digital timing and analog circuit verification</td>
</tr>
</tbody>
</table>
True-Value Simulation Algorithms

- Compiled-code simulation
  - Applicable to zero-delay combinational logic
  - Also used for cycle-accurate synchronous sequential circuits for logic verification
  - Efficient for highly active circuits, but inefficient for low-activity circuits
  - High-level (e.g., C language) models can be used

- Event-driven simulation
  - Only gates or modules with input events are evaluated *(event means a signal change)*
  - Delays can be accurately simulated for timing verification
  - Efficient for low-activity circuits
  - Can be extended for fault simulation
Compiled-Code Algorithm

- Step 1: Levelize combinational logic and encode in a compilable programming language
- Step 2: Initialize internal state variables (flip-flops)
- Step 3: For each input vector
  - Set primary input variables
  - Repeat (until steady-state or max. iterations)
    - Execute compiled code
  - Report or save computed variables
Event-Driven Algorithm

\[ a = 1 \quad b = 1 \quad c = 1 \quad d = 0 \quad e = 1 \quad f = 0 \quad g = 1 \]

<table>
<thead>
<tr>
<th>Time stack</th>
<th>Scheduled events</th>
<th>Activity list</th>
</tr>
</thead>
<tbody>
<tr>
<td>t = 0</td>
<td>c = 0</td>
<td>d, e</td>
</tr>
<tr>
<td>1</td>
<td>d = 1, e = 0</td>
<td>f, g</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>g = 0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>f = 1</td>
<td>g</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>g = 1</td>
<td></td>
</tr>
</tbody>
</table>
Time Wheel (Circular Stack)

Current time pointer

max

Event link-list

Current time pointer

t=0
1
2
3
4
5
6
7
Efficiency of Event-driven Simulator

- Simulates events (value changes) only
- Speed up over compiled-code can be ten times or more; in large logic circuits about 0.1 to 10% gates become active for an input change
Problem and Motivation

Fault simulation Problem: Given

- A circuit
- A sequence of test vectors
- A fault model

- Determine
  - Fault coverage - fraction (or percentage) of modeled faults detected by test vectors
  - Set of undetected faults

Motivation

- Determine test quality and in turn product quality
- Find undetected fault targets to improve tests
Fault simulator in a VLSI Design Process

- Verified design netlist
- Verification input stimuli
- Fault simulator
- Modeled fault list
- Test vectors
- Test compactor
- Remove tested faults
- Add vectors
- Low
- Delete vectors
- Test coverage?
  - Fault coverage?
    - Adequate
    - Stop
  - Low
Fault Simulation Scenario

- Circuit model: mixed-level
  - Mostly logic with some switch-level for high-impedance (Z) and bidirectional signals
  - High-level models (memory, etc.) with pin faults

- Signal states: logic
  - Two (0, 1) or three (0, 1, X) states for purely Boolean logic circuits
  - Four states (0, 1, X, Z) for sequential MOS circuits

- Timing
  - Zero-delay for combinational and synchronous circuits
  - Mostly unit-delay for circuits with feedback
Fault Simulation Scenario

Faults

- Mostly single stuck-at faults
- Sometimes stuck-open, transition, and path-delay faults; analog circuit fault simulators are not yet in common use
- Equivalence fault collapsing of single stuck-at faults
- Fault-dropping -- a fault once detected is dropped from consideration as more vectors are simulated; fault-dropping may be suppressed for diagnosis
- Fault sampling -- a random sample of faults is simulated when the circuit is large
Fault Simulation Algorithms

- Serial
- Parallel
- Deductive
- Concurrent
Serial Algorithm

Algorithm: Simulate fault-free circuit and save responses. Repeat following steps for each fault in the fault list:

- Modify netlist by injecting one fault
- Simulate modified netlist, vector by vector, comparing responses with saved responses
- If response differs, report fault detection and suspend simulation of remaining vectors

Advantages:

- Easy to implement; needs only a true-value simulator, less memory
- Most faults, including analog faults, can be simulated
Serial Algorithm

- Disadvantage: Much repeated computation; CPU time prohibitive for VLSI circuits
- Alternative: Simulate many faults together

Test vectors → Fault-free circuit → Comparator → f1 detected?
  → Circuit with fault f1 → Comparator → f2 detected?
  → Circuit with fault f2 → Comparator → fn detected?
  → Circuit with fault fn

Thank You