Chapter 4
Fault Simulation

Jin-Fu Li
Advanced Reliable Systems (ARES) Lab.
Department of Electrical Engineering
National Central University
Jungli, Taiwan
Outline

- Introduction
- Fault Simulation
  - Parallel Fault Simulation
  - Deductive Fault Simulation
  - Concurrent Fault Simulation
- Summary
**Simulation**

- **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**

- **Simulator is used for design verification**
  - Validate assumptions
  - Verify logic
  - Verify performance (timing)
Levels of Simulation

- System level
- Architecture level
- Functional level/RTL level
- Gate/structural level
- Switch/transistor/circuit level
- Mixed level

System level

RTL level

Gate level
Simulation Process

- Library
- Stimuli
- Model
- Simulator
- Response
Modeling

- Modules, blocks or components described by:
  - Input/output (I/O) function
  - Delays associated with I/O signals
  - Examples: binary adder, Boolean gate, etc.

- Interconnects represent:
  - Ideal signal carriers or ideal electrical conductors

- Netlist:
  - A format (or language) that describes a design as an interconnection of modules. Netlist may use hierarchy.
Example: Full-Adder Netlists

HA;
inputs: a, b;
outputs: c, f;
AND: A1, (a, b), (c);
AND: A2, (d, e), (f);
OR: O1, (a, b), (d);
NOT: N1, (c), (e);

FA;
inputs: A, B, C;
outputs: Carry, Sum;
HA: HA1, (A, B), (D, E);
HA: HA2, (E, C), (F, Sum);
OR: O2, (D, F), (Carry);
Types of Simulation

- **Compiled 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 Simulation

- **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 Simulation

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

Scheduled events:
- Time stack:
  - 0: \( c = 0 \)
  - 1: \( d = 1, e = 0 \)
  - 2: \( f = 1 \)
  - 3: \( g = 0 \)
  - 4: \( g = 0 \)
  - 5: \( g = 0 \)
  - 6: \( f = 1 \)
  - 7: \( g = 1 \)
  - 8: \( g = 1 \)

Activity list:
- \( d, e \)
- \( f, g \)
- \( g \)
Time Wheel (Circular Stack)
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
Fault Simulation

☐ Fault simulation
  ■ In general, simulating a circuit in the presence of faults is known as fault simulation

☐ The main goals of fault simulation
  ■ Measuring the effectiveness of the test patterns
  ■ Guiding the test pattern generator program
  ■ Generating fault dictionaries

☐ Outputs of fault simulation
  ■ Fault coverage - fraction (or percentage) of modeled faults detected by test vectors
  ■ Set of undetected faults
Elements of Fault Simulation

- The fault simulation process is illustrated as below

- The fault simulator affects the speed of overall fault simulation
Fault Simulation Scenario

- Circuit model: mixed-level
  - Mostly logic with some switch-level for high-impedance (Z) and bidirectional signals

- Signal states: logic
  - Two states (0, 1), three states (0, 1, X), four states (0, 1, X, Z), etc.

- Timing:
  - Zero-delay
    - For combinational circuits with no feedback
  - Unit-delay
    - It can maintain the proper sequencing of signal changes
  - Multiple-delay
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
Serial Fault Simulation

- Serial fault simulation 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 is required
Serial Fault Simulation

- Disadvantages
  - Much repeated computation; CPU time prohibitive for VLSI circuits

- Alternative
  - Simulate many faults together
Parallel Fault Simulation

- Assumptions
  - The simulated circuit consists of only logic gates and all gates have the same delays
  - Signals take only binary (0 and 1) values

- Main idea
  - Take advantage of the bit-parallelism of logical operations in a digital computer
    - For a 32-bit machine word, an integer consists of a 32-bit binary vector
    - A logic AND or OR operation involving two words performs simultaneous AND or OR operations on all respective pairs of bits

- Storage requirement
  - One word per line for two-state simulation
Speedup

- If the computer word size is $N$, then $N-1$ copies of faulty circuit are also generated.
  - For a total $M$ faults in the circuit, $\left\lfloor \frac{M}{(N-1)} \right\rfloor$ simulation runs are then necessary.
- Speedup over serial fault simulation about $N-1$.
- Disadvantages:
  - Lacking the capability to simulate accurate rise and fall delays of signals.
  - Not suitable for circuits with non-Boolean logic.
An Example of Parallel Fault Sim.

Bit 0: fault-free circuit
Bit 1: circuit with $c$ s-a-0
Bit 2: circuit with $f$ s-a-1

$c$ s-a-0 detected
Deductive Fault Simulation

- Simulating only the behavior of the fault-free logic circuits
- Need only one pass for each test pattern
- All signal values in each faulty circuit are deduced from the fault-free circuit values and the circuit structure
- For each test pattern, a deductive procedure is applied to all lines in a level-order (for combinational logic) from inputs to outputs
Deductive Fault Simulation

- **Definition**
  - The *fault list* $L_A$ is defined as the set containing the name or index of every fault that produces an error on line $A$ when the circuit is in its current logic state.

- Fault lists are to be propagated from PIs to the POs.

- A fault list is generated for each signal line, and updated as necessary with every change in the logic state of the circuit.

- List events occur when a fault list changes.
Consider a 4-input NOR gate with input \{ABCD\}={0011}, given the initial fault lists as show below:

- \( L_A = \{a,e\} \)
- \( L_B = \{b,c\} \)
- \( L_C = \{a,b,c,d\} \)
- \( L_D = \{a,d,f\} \)

The faults that propagate to \( E \) are those causing \( E \) to be complemented:

\[
L'_E = \overline{(L_A \cup L_B)} \cap L_C \cap L_D = \{\alpha \mid \alpha \in L_C \cap L_D \land \alpha \neq L_A \cup L_B\}
\]

We also have to consider internal faults producing incorrect output for the current good input:

\[
L_E = L'_E \cup \{E/1\} = \{d, E/1\}
\]
Fault List Propagation

Let $I$ be the set of inputs of a gate with output $Z$, controlling value $c$, and inversion $i$. Let $C$ be the set of inputs with value $c$. The fault list $L_Z$ is computed as follows:

\[
\text{if } (C = \phi) \text{ then } L_Z = \left\{ \bigcup_{j \in I} L_j \right\} \bigcup \{Z / (c \oplus i)\};
\]

\[
\text{else } L_Z = (\left\{ \bigcap_{j \in C} L_j \right\} - \left\{ \bigcup_{j \in I - C} L_j \right\}) \bigcup \{Z / (\overline{c} \oplus i)\}
\]

Controlling value: $c = \begin{cases} 
0, \text{ for AND/NAND gates} \\
1, \text{ for OR/NOR gates}
\end{cases}$

Inversion: $i = \begin{cases} 
0, \text{ for AND/OR gates} \\
1, \text{ for NAND/NOR gates}
\end{cases}$
An Example of Deductive Fault Sim

Notation: $L_k$ is fault list for line $k$

$k_n$ is s-a-n fault on line $k$

$$L_e = L_a \cup L_c \cup \{e/0\} = \{a/0, b/0, c/0, e/0\}$$

$$L_g = (L_e \cap \overline{L_f}) \cup \{g/0\} = \{a/0, c/0, e/0, g/0\}$$

Faults detected by the input vector
Concurrent Fault Simulation

- Event-driven simulation of fault-free circuit and only those parts of the faulty circuit that differ in signal states from the fault-free circuit.
- A list per gate containing copies of the gate from all faulty circuits in which this gate differs. List element contains fault ID, gate input and output values and internal states, if any.
- All events of fault-free and all faulty circuits are implicitly simulated.
- Faster than other methods, but uses most memory.
An Example of Concurrent Fault Sim

```
a/0  b/0  c/0  e/0
0  0   1  0
1  0   0  1
1  0   1  0
```

```
a  b  c
d  e  f
g
```

```
1  1
1  1
0  0
```

```
a/0  b/0  c/0  e/0
0  0   0  0
0  0   1  0
0  0   0  0
```

```
0  1
1  1
0  0
0  0
```

```
1  1
1  1
1  1
```

```
0  1
1  1
1  1
```
Summary

- Fault simulator is an essential tool for test development
- The main goals of fault simulation
  - Measuring the effectiveness of the test patterns
  - Guiding the test pattern generator program
  - Generating fault dictionaries
- Concurrent fault simulation algorithm offers the best choice