ECE 538

VLSI System Testing

Krish Chakrabarty

Fault Simulation

Fault Simulation

- Problem and motivation
- Fault simulation algorithms
  - Serial
  - Parallel
  - Deductive
  - Concurrent
- Random Fault Sampling
- Summary

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

Motivation

- Simulate a circuit in the presence of faults
- Given test set T, determine the fault coverage of T (*test grading*)
  - Fraction of faults (percentage) detected by T
- How is fault coverage related to *defect coverage* and *yield*?

$$DL = 1 - Y^{1-d}$$

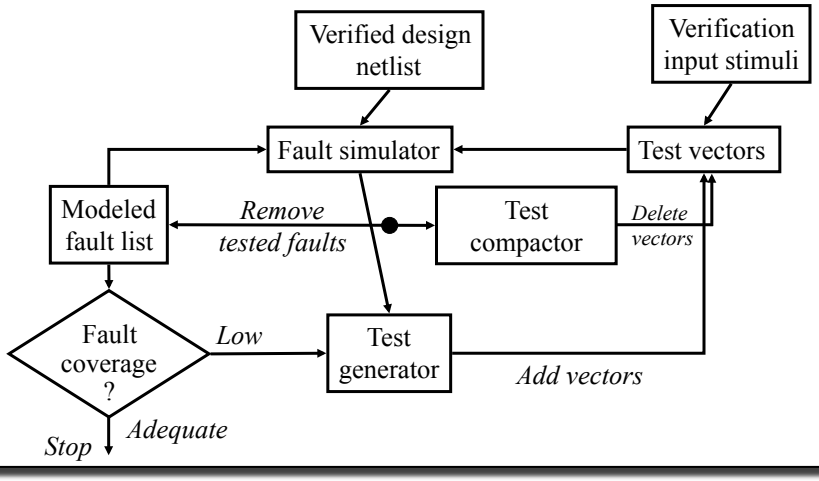
DL (defect level): Probability of shipping a defective chip
Y: yield, d: fault (defect) coverage

e.g. if Y = 0.5, 99% fault coverage needed to achieve 0.01 defect level
95% fault coverage implies 0.035 defect level

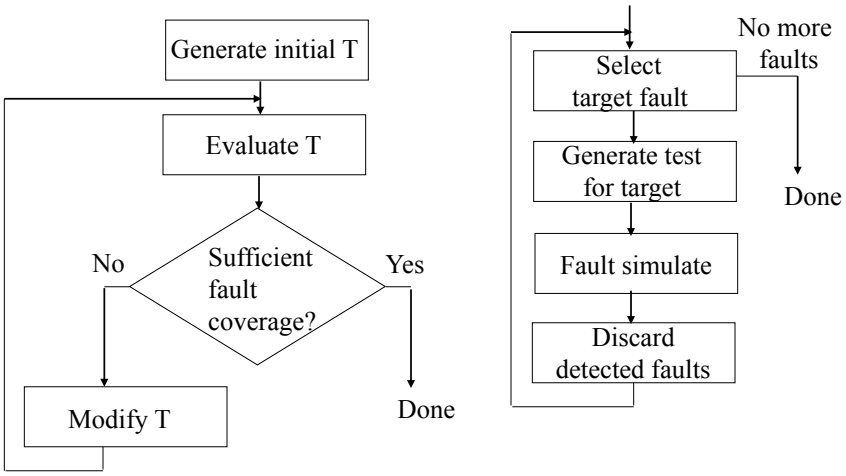
Applications

- Evaluate effects (criticality) of faults
- Evaluate fault coverage of given test set
- Generate fault dictionaries (for diagnosis)
- Aid in test pattern generation
  - Fault dropping
  - Test set compaction
  - Simulation-based and random test generation

Fault Simulator in a VLSI Design Process



Use in Test Generation



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 (continued)

- 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

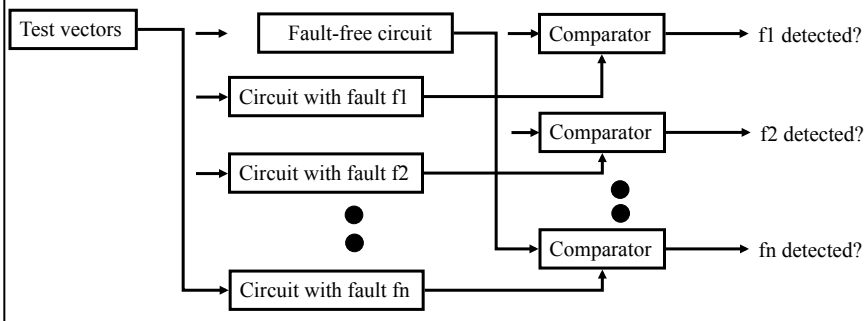
- Serial
- Parallel
- Deductive
- Concurrent

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 (Cont.)

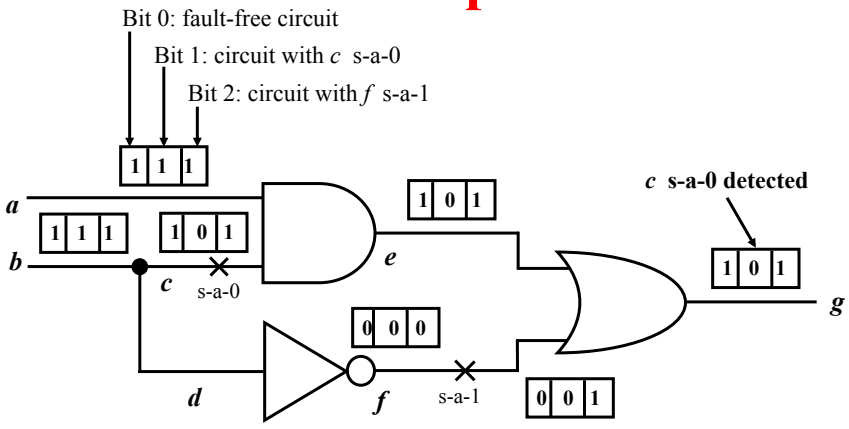
- Disadvantage: Much repeated computation; CPU time prohibitive for VLSI circuits
- Alternative: Simulate many faults together



Parallel Fault Simulation

- Compiled-code method; best with two-states (0,1)
- Exploits inherent bit-parallelism of logic operations on computer words
- Storage: one word per line for two-state simulation
- Multi-pass simulation: Each pass simulates w-1 new faults, where w is the machine word length
- Speed up over serial method ~ w-1
- Not suitable for circuits with timing-critical and non-Boolean logic

Parallel Fault Simulation Example



Limitations of Parallel Fault Simulation

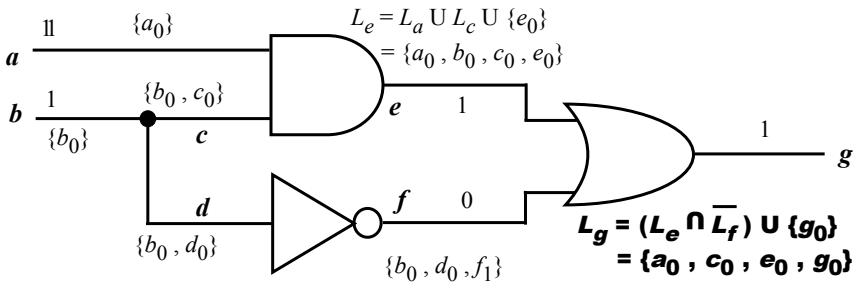
- Useful for two (1,0) or three (0,1,X) logic values, not suitable for multiple logic values, e.g. (0,1,U,R,F,…)
  - Multiple logic values *can be* handled but operations are complex
- Wasted computations
  - Fault dropping not carried out effectively
  - Not possible to discard faults that are in the same word

Deductive Fault Simulation

- One-pass simulation
- Each line k contains a list L_k of faults detectable on k
- Following true-value simulation of each vector, fault lists of all gate output lines are updated using set-theoretic rules, signal values, and gate input fault lists
- PO fault lists provide detection data
- Limitations:
  - Set-theoretic rules difficult to derive for non-Boolean gates
  - Gate delays are difficult to use

Deductive Fault Simulation Example

Notation: L_k is fault list for line k
k_n is s-a-n fault on line k



Faults detected by the input vector

Fault List Propagation

Consider an AND gate Z = A.B, and let A = B = 1
Then $L_Z = L_A \cup L_B \cup \{Z \text{ s-a-0}\}$

Let A = 0 and B = 1
Then $L_Z = \{L_A \cap \overline{L_B}\} \cup \{Z \text{ s-a-1}\} = (L_A - L_B) \cup \{Z \text{ s-a-1}\}$

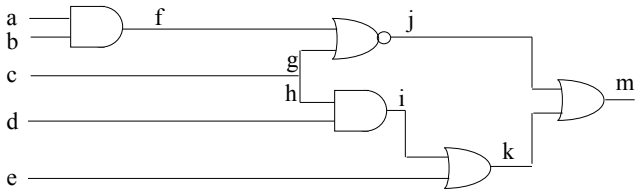
In general, let I be the set of inputs of gate Z with controlling value c and inversion i. Let C be the set of inputs with value c.

if C = ∅ **then** $L_Z = \{\bigcup_{j \in I} L_j\} \cup \{Z \text{ s-a-}(c \oplus i)\}$

else $L_Z = \{\bigcap_{j \in C} L_j\} - \{\bigcup_{j \in I - C} L_j\} \cup \{Z \text{ s-a-}(\bar{c} \oplus i)\}$

- If no input has value c then fault effect on an input propagates to output
- If some inputs have value c, only faults that affect inputs with c (not $\bar{c}$) propagates to output

Deductive Simulation Example



Input vector: 00110
Fault set (after fault collapsing): {a/0, a/1, b/1, c/0, c/1, d/1, e/0, g/0, h/0, h/1}

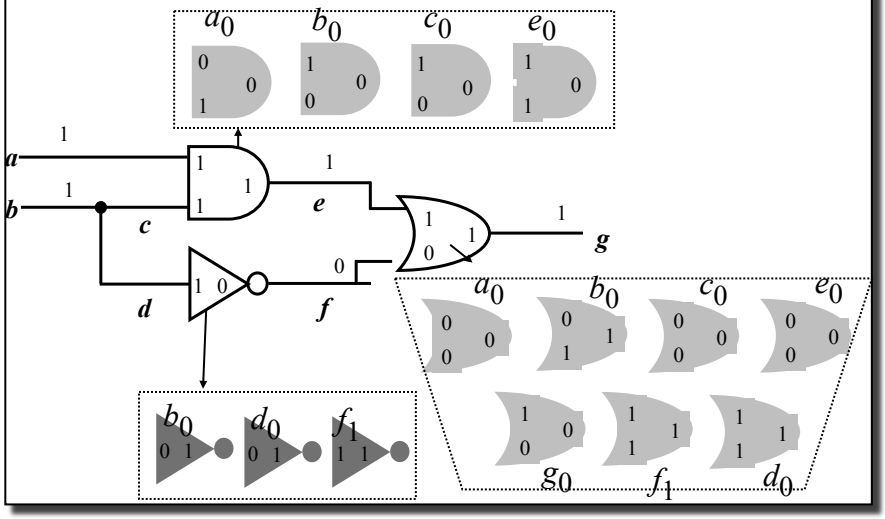
$L_a = \{a/1\}$, $L_b = \{b/1\}$, $L_c = \{c/0\}$, $L_d = \varnothing$, $L_e = \varnothing$
$L_f = L_a \cap L_b = \varnothing$, $L_g = L_c \cup \{g/0\} = \{c/0, g/0\}$, $L_h = L_c \cup \{h/0\} = \{c/0,h/0\}$
$L_j = L_g - L_f = \{c/0, g/0\}$, $L_i = L_d \cup L_h = \{c/0, h/0\}$
$L_k = L_i - L_e = \{c/0, h/0\}$, $L_m = L_k - L_j = \{h/0\}$

h/0 is detected! Delete h/0 from fault lists L_h, L_i, L_k, and L_m

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.
- Faults can be simulated in any modeling style or detail supported in true-value simulation (offers most flexibility.)
- Faster than other methods, but uses most memory.

Conc. Fault Sim. Example



Comments

- Parallel
  - Fast, easy gate evaluation
  - Parallelism limited by host's word size
  - Can handle multiple logic values
- Deductive
  - All faults simulated in one pass
  - Complex gate evaluations
  - Can handle few only few logic values
  - Feedback causes problems
- Concurrent
  - Activated gates are simulated (in one pass)
  - Fairly easy gate evaluation
  - Easy to use multiple values
  - Fault lists can be very long

Fault Sampling

- A randomly selected subset (sample) of faults is simulated.
- Measured coverage in the sample is used to estimate fault coverage in the entire circuit.
- Advantage: Saving in computing resources (CPU time and memory.)
- Disadvantage: Limited data on undetected faults.

Motivation for Sampling

- Complexity of fault simulation depends on:
  - Number of gates
  - Number of faults
  - Number of vectors
- Complexity of fault simulation with fault sampling depends on:
  - Number of gates
  - Number of vectors

Summary

- Fault simulator is an essential tool for test development.
- Concurrent fault simulation algorithm offers the best choice.
- For large circuits, the accuracy of random fault sampling only depends on the sample size (1,000 to 2,000 faults) and not on the circuit size. The method has significant advantages in reducing CPU time and memory needs of the simulator.