Fault Simulation Using Irsim

Arturo Salz
Computer Systems Laboratory
Stanford University

April 3, 1993

1 Introduction

Digital VLSI circuits are tested by applying to their input pins a set of patterns that exercise some or all of the circuit's functions. During fabrication, some of these functions may develop faults that are sensitive to the test patterns. In most cases, however, a test pattern cannot, for practical reasons, test all faults since that would require use of exhaustive data patterns. Therefore, although a circuit may pass the test, there is no guarantee that the circuit is fault-free. Thus, it is important to determine how well a test can identify a faulty circuit.

The most common criterion for judging the quality of a test is the fault coverage obtained from a fault simulator. Typically, fault coverage refers to the percentage of *single faults* detected by the test. As a rule, test engineers attempt to provide as close as possible to 100% fault coverage. Due to the difficulty in developing such tests, however, in practice, a fault coverage that is too high is difficult to achieve.

2 Fault Simulation: Basic Concept

In order to fault-simulate a design, the designer must provide the circuit netlist and the test pattern to be applied to circuit's inputs. The test pattern is used to simulate the circuit once, yielding the expected values at the outputs of the circuit (i.e. the "good" machine). This step is the same as a conventional simulation. After the fault-less circuit has been simulated, the simulator proceeds to inject into the circuit one fault at a time. For each such fault, the circuit is resimulated in order to determine whether the fault is observable at the circuit's outputs. To do this, the designers must also tell the fault simulator which nodes correspond to output pins and when they are to be sampled. For example, output pins whose data is valid on $\Phi 1$ should be sampled only on the falling edge of $\Phi 1$.

Faults that cause an output pin (or pins) to deviate from the value provided by the good machine at the time of sampling are said to be observable: their effect can be detected by the test pattern in question. Note, however, that since **X** represents an intermediate or unknown logical value, a fault that causes an output to become **X** (or viceversa) may not be detectable in practice. Such faults are said to be non-deterministically observable.

Although faults on actual chips are caused by physical defects, such as breaks or shorts between the fabrication layers, the most common type of fault model used by fault simulators is the "stuck-at" fault. This type of fault models each faulty circuit by sticking (or forcing) a fixed

value onto one of the internal nodes in the circuit. Thus, to test whether a fault at a particular node is observable, the stuck-at model simulates the circuit twice: once with the node stuck-at-0, and once with the node stuck-at-1. Since testing all nodes in the circuit in this fashion would be extremely time-consuming, most fault simulators restrict their analysis to some statistically representative number of nodes. The process of selecting which nodes are to be tested is termed *fault seeding*.

3 Fault Simulation Using Irsim

In order to run a fault simulation, you must use a special version of *Irsim*, called "ifsim". If you previously used *Irsim* to simulate your design, you only need to create one additional file before starting a fault simulation. *Ifsim* accepts all the same commands as *Irsim*, plus one additional command to initiate a fault simulation: **faultsim**.

The easiest way to run a fault simulation on your design is to first run *Ifsim* as you would using *Irsim*, and at the end of the simulation run issue the faultsim command. This command has the following form:

faultsim *setup-file* [*output-file*]

where *setup-file* is the name of a file containing the information regarding output pins and their sampling timing, and *output-file* is the name of the file where the fault-simulation output will be recorded. If no *output-file* is specified, *Ifsim* will use the file **fsim.out**.

After reading the setup file, *Ifsim* will select the maximum number of nodes into which stuck-at faults will be injected. By default, *Ifsim* uses 20% of all nodes in the circuit. Fault seeding then proceeds by randomly selecting the required number of nodes from all candidate nodes. A node is a fault candidate if at least one transistor gate is connected to the node, and the node is not a primary input (it is never driven). At the end of the simulation, the output file will contain detailed information regarding each fault tested; this is followed by a brief summary that includes the fault coverage.

3.1 Setup file

This file specifies all output nodes, the timing for their sampling, and, optionally, the percentage of nodes to be seeded. There are basically three commands that can be used in this file; their syntax is as follows:

seed ⟨*percentage*⟩

trigger ⟨*on-node*⟩ ⟨*transition-value*⟩ [*delay*]
⟨*output-list*⟩
***

sample ⟨*period*⟩ [*offset*]
⟨*output-list*⟩
***

The **seed** command, if present, must be the first non-empty line in the file. It's ⟨*percentage*⟩ argument must be an integer number (in the range [1–100]), which specifies the percentage of the circuit's nodes that should be considered for fault seeding. If this command is missing, *Ifsim* will seed up to 20% of the circuit's nodes.

Outputs can be sampled using either a fixed time interval, or the rising/falling edge of some other signal in the circuit. The first type of sampling is specified using the **sample** command, the second is specified through the **trigger** command.

The **trigger** command indicates that all nodes specified in ⟨*output-list*⟩ should be sampled when ⟨*on-node*⟩ makes a transition to ⟨*transition-value*⟩. Note that ⟨*output-list*⟩ can span multiple lines, each of which can contain a list of nodes separated by blanks. The list of outputs is terminated by a line containing the single entry ***. Also, just like in *Irsim*, any node name can include the wildcard character '*', it can denote iteration by using the pair of characters '{' and '}', or it can be the name of a user-defined vector. The optional argument ⟨*delay*⟩ specifies that the signals should be sampled ⟨*delay*⟩ ns after the specified transition. For example, the following file indicates that primary outputs ADR0, ADR1, ADR2, ADR3, RD, and WR should be sampled on the falling edge of signal phi1, while outputs DATA0, DATA1, DATA2, DATA3, and DATA4 should be sampled 10ns after the rising edge of signal phi2:

```
trigger phi1 0
        ADR{0:3}
        RD WR
***

trigger phi2 1 10
        DATA0 DATA1 DATA2
        DATA3
        DATA4
***
```

The format of the **sample** command is similar to the **trigger** command, however, instead of using another signal to trigger the sampling of the outputs, they are sampled every ⟨*period*⟩ ns. The optional ⟨*offset*⟩ argument indicates that the sampling should begin ⟨*offset*⟩ ns from the start of the simulation. For example, the following entry indicates that outputs clock1 and clock2 should be sampled every 50ns starting from time 100ns (150, 200, etc):

```
sample 50.0 100.0
        clock1 clock2
***
```

3.2 Output File

The output file produced by *Ifsim* consists of a series of lines, each of which indicates the result of injecting a single fault. Each line indicates the following information:

Detected Fault:

- node at which the fault was injected.
- type of fault (1 for stuck-at-1, 0 for stuck-at-0).
- simulation time at which the fault was detected.
- the output at which the fault was detected.

Undetected Fault:

- the node at which the fault was injected.
- type of fault (1 for stuck-at-1, 0 for stuck-at-0).

The list of faults is followed by a summary that includes the following:

- The total number of faults seeded.
- The number of detected faults.
- The number of undetected faults.
- The fault coverage as the percentage of detected faults.
- The number of probably detected faults.

The last item corresponds to outputs that deviate from the good machine at the time of sampling, but they do so by deviating to or from an **X** logic state.

For example, the following output file indicates that if NODE1 is stuck-at-0, the fault was detected on output ADR0 at time 250.0ns. Conversely, if NODE1 is stuck-at-1, the fault was not detected. The rest of the entries are similar, except the last one which indicates that if NODE3 is stuck-at-1, the fault was detected as a change from (or to) an **X** value on output ADR3, at time 600.0ns.

```
Detect    0   NODE1 [250.0] ADR0
Fail      1   NODE1
Detect    0   NODE2 [450.0] ADR1
Detect    1   NODE2 [350.0] ADR2
Fail      0   NODE3
Maybe     1   NODE3 [600.0] ADR3
— —
6 faults
4 detected (1 probably)
2 undetected
fault coverage: 66.66% (83.33%)
```