

Identification of The Nearly Untestable Path Delay Faults

Source: SynTest Technologies, Inc.

December 26 2001

I. Introduction

A VLSI chip may pass the stuck-at fault testing at low speed testing but may still exhibit erroneous behavior due to timing problems. Path delay fault model [1] is one of the most useful timing fault models. A path fault can be defined by a sequence of nodes connected by nets from primary input (PI) to primary output (PO). To detect a path fault, an initialization pattern and a propagation pattern may be applied to the circuit under test. The basic hardware model for delay testing is shown below as in Fig. 1.

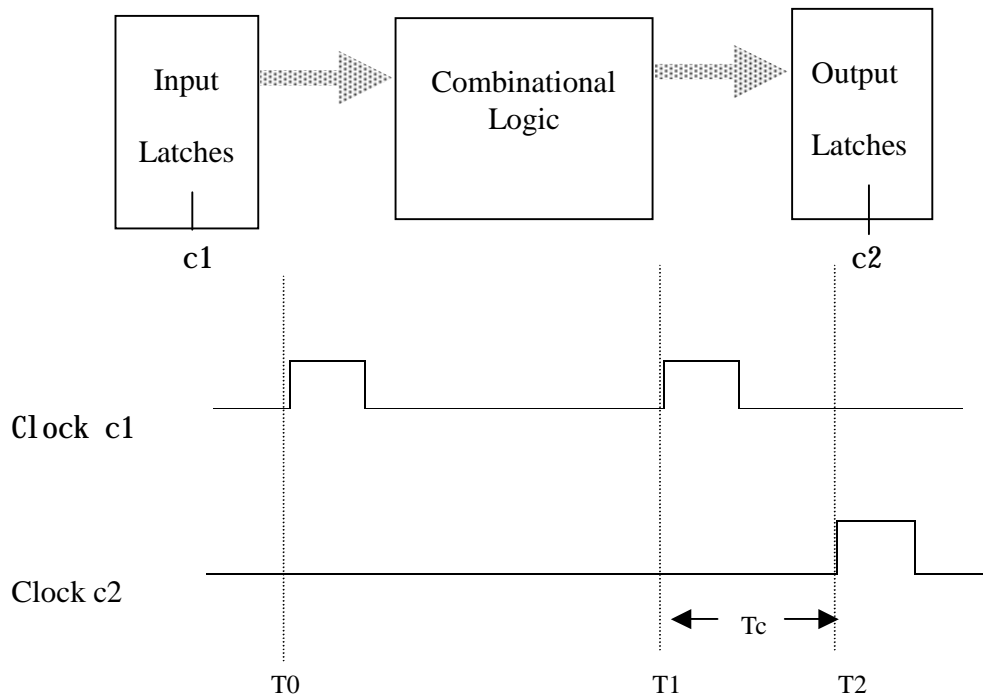


Fig. 1: Model for path delay fault testing.

In this model, the initialization pattern is applied to input latches at time T_0 . After all signals of the circuit are stable, the propagation pattern is applied to the circuit at time T_1 . Consequently, the output signals are sampled after a time interval T_c , where T_c is the system clock interval.

There are two models for path delay testing, the robust and non-robust tests [2]. The robust test gives a lower bound while the non-robust gives the upper bound of the real fault testing. A robust path delay test detects the path fault on a path independent of the other path delays in a circuit. A

non-robust path delay test, on the other hand, detects the path delay fault on a path based on the assumption that the off-path inputs of all gates on the path stabilize when the transitions propagating along the path arrive at the gates. To simplify the problem, a path delay fault is referred to as *nearly untestable* if there does not exist a *robust* or a *non-robust test* for this fault.

II. Problem Description

Given a gate-level circuit, develop a tool that can identify the *nearly untestable faults*. The basic requirements of the tool include:

1. Rank each path based on its length (using unit delay model).
2. Identify as many nearly untestable faults as possible.

III. Input

The input is a gate-level circuit (ISCAS benchmarks format). Circuit s208 and s298 from ISCAS'89 will be used as test circuits. Assume that circuit s208 and s298 (or any sequential circuit that will be used as test circuit) is full-scanned. In all test circuits, only gates of type INPUT, OUTPUT, NOT, AND, OR, NAND, NOR and DFF are used. All gates have single output only. Line comments are preceded by the # sign. The ISCAS circuits can be downloaded from:

<ftp://ftp.syntest.com.tw/pub/benchmark/>.

Below is the input format of the circuit in figure 2.

```
INPUT(X1)
INPUT(X2)
INPUT(X3)
```

```
# The output is driven by the output of G3
OUTPUT(G3)
```

```
# gate G1 is an AND gate. The inputs to G1 are the outputs from X1 and X2
G1 = AND(X1, X2)
```

```
# gate G2 is an OR gate. The inputs to G2 are the outputs from X1 and X3
G2 = OR(X1, X3)
```

```
# gate G3 is an OR gate. The inputs to G1 are the outputs from G1 and G2
G3= OR(G1, G2)
```

IV. Output

The output file consists of a list of path faults. Each fault is associated with two parts: (1) the polarity and (2) the path. The polarity, indicating whether the fault is a rising fault or falling fault, is denoted by a keyword *Rising* or *Falling*. The path is denoted by a sequence of gates.

Here is the output format:

```
% begin
Polarity <gate>+
% end
```

For example:

```
% begin
Rising G1 G2 G10
Falling G1 G3 G9
% end
```

V. Example

Consider a circuit shown in Fig. 2. There are three primary inputs {X1, X2, X3}, one primary output {Z}, and three internal gates {G1, G2, and G3}. There are four paths in this circuit:

Path 1: X1 → G1 → G3 → Z
Path 2: X2 → G1 → G3 → Z
Path 3: X2 → G2 → G3 → Z
Path 4: X3 → G2 → G3 → Z

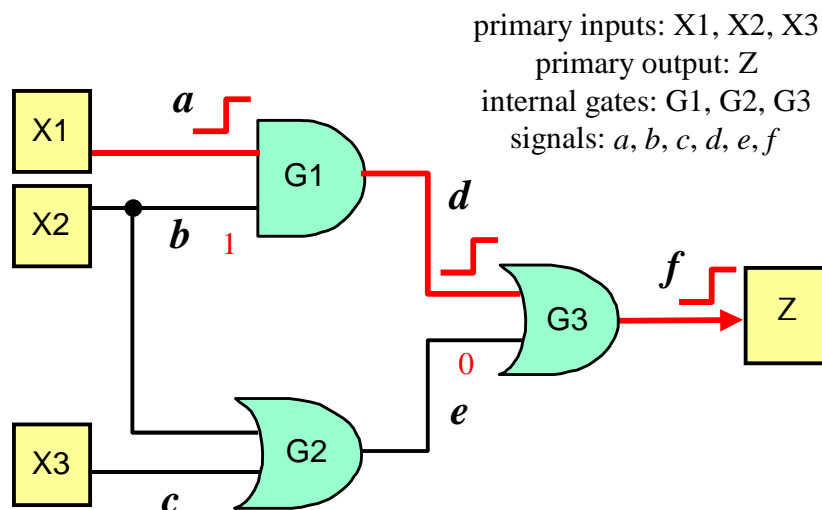


Fig. 2: An example for almost untestable fault identification.

When it comes to propagation delay of a path, each path is associated with a rising or falling transition at its source node (i.e., the primary input node), hence each path corresponds to two potential path faults that need to be considered (one for the rising path delay fault, the other for the falling path delay fault). Now let us consider the test generation for a path delay fault:

Path Fault F1: Rising $X1 \rightarrow G1 \rightarrow G3 \rightarrow Z$

In order to test this path fault robustly or non-robustly, we need to find a *two-pattern test* through the following steps. If it turns out such a test is not possible, then we can claim that the fault being considered is not robustly or non-robustly testable, or *nearly untestable*.

Here is a typical procedure for path delay fault test generation:

- (1) Launch a rising transition at the source node (i.e., $X1$). Hence the requirement of the two-pattern test at $X1$ is $(0 \rightarrow 1)$.
- (2) Propagate the launched transition in step (1) through every *on-path signals* to the destination node (i.e., Z). Here the on-path signals refers to those signals that lie on the path being considered. In order to complete this transition propagation, every *side-input signals* (e.g., b , c , and e) have to be set to the non-controlling values for the second test pattern. Therefore, we arrive at the requirements for the following signals (using u to denote the don't care logic value):
Signal b : $(u \rightarrow 1)$ Signal e : $(u \rightarrow 0)$
- (3) Justify the requirements derived in step (1) and (2) through the assignments at primary inputs. If any inconsistency occurs among these requirements, it can be claimed that no 2-pattern robust or non-robust test exists, and this fault is *nearly untestable*.

It is worth mentioning that the identification of nearly untestable path faults does not always require a full-blown test generation as shown above. Through the implication techniques, one might be able to identify some of the nearly untestable faults easily and efficiently.

Let us go back to the fault that we were considering. Because setting signal b to '1' will always imply a '1' at signal e in the circuit, we can conclude that the requirements of setting b to '1' and e to '0' simultaneously for the transition propagation derived above is not possible. The fault being considered is thus declared as a *nearly untestable path fault*. As a result, we should include the following path fault in the final output file:

```
% begin
Rising X1 G1 G3
(other nearly untestable faults)
% end
```

VI. Language/Platform

- Language: C
- Platform: Sun/Solaris

VII. Evaluation

- (1) Robustness of the program
- (2) The number of identified nearly untestable faults.
- (3) CPU time
- (4) Memory requirement

References:

- [1] G.L. Smith, "Model for delay faults based upon paths," *Proc. IEEE International Test Conference, Sept. 1985, pp 342-348.*
- [2] E.S. Park and M.R. Mercer, "Robust and non-robust tests for path delay faults in a combinational circuit," *Proceedings of 1987 International Test Conference (ITC'87)*, page 1027-1034.