A Trace–Based Method for Delay Fault Diagnosis in Synchronous Sequential Circuits

P. Girard, C. Landrault, S. Pravossoudovitch and B. Rodriguez

Laboratoire d'Informatique de Robotique et de Microélectronique de Montpellier,
UMR 9928 UNIVERSITE MONTPELLIER II / CNRS
161 rue Ada, 34392 Montpellier Cedex 05, FRANCE.

***Abstract** – In this paper, we present a method for diagnosing gate delay faults in synchronous sequential circuits. This method is an outgrowth of our previous work on delay fault diagnosis in combinational circuits, and is therefore based on a path tracing algorithm appropriate for sequential circuits. Input data for diagnosis are (1) the gate level description of the circuit, (2) the set of test sequences, and (3) the set of failing patterns and failing outputs provided by the tester. Output data are a set of potential fault locations. In order to correctly interpret the tester results, and avoid multiple fault effects and self–masking problems during diagnostic processing, each test sequence is considered under different combinations of slow and fast clock cycles (slow clock test methodology). Experimental results are given to show the feasibility, reliability and efficiency of the diagnosis method.*

1. Introduction

Since the introduction of the LSSD Design For Testability technique, most of the work reported on delay testing is applicable only to scan–based circuits. This restricted problem is, of course, more tractable than the delay testing of general sequential circuits. Unfortunately, many practical circuits do not conform to this kind of architecture because the hardware overhead in such a design may be high and affects performances. Delay testing must therefore be studied for circuits designed without scan. At the present time, several specific methods exist in the areas of fault modeling, fault simulation and test pattern generation [2,6,8,9,16]. Conversely, no oriented method exists for delay fault diagnosis in non–scan circuits.

From a general point of view, the aim in diagnosis is not to find out whether there is a fault (*detection*), but rather where the fault is located (*diagnosis*). This kind of information is used to perfect the manufacturing or the designing process of a circuit. Typically, a preliminary diagnosis provides a set of lines containing a number of potential fault locations (*software approach*), whereas a visual inspection or an electron–beam probing of the faulty circuit is carried out later to identify the cause of failure (*hardware approach*).

A software method to diagnose delay faults in combinational circuits was proposed recently and can be found in the literature [12]. This method is based on a Critical Path Tracing (*CPT*) algorithm, thus representing an alternative to delay fault simulation. Results of experiments obtained with benchmark circuits have shown the effectiveness of the method.

Delay fault diagnosis in synchronous sequential circuits that do not use scan–design is a more difficult problem. Basically, it results from the fact that direct access to the flip–flops in the circuit is not provided. This implies that a delay fault activated after the application of a test vector (during the test mode) may not always be observed when the next clock pulse is applied, but may sometimes need to be propagated during a number of clock cycles to appear on primary outputs. The study of fault effect propagation, which is necessary during diagnosis, must therefore be extended to the entire sequential circuit (combinational and memory parts) rather than be limited to the combinational part of the circuit. However, to find fault effect propagation paths without any restriction on the delay test methodology can be a serious problem on account of multiple fault effects which may, for example, generate self–masking.

If one considers a particular testing scheme, the method to diagnose delay faults in combinational circuits [12] can be extended to sequential circuits. In this paper, we aim to present the trace–based method we have developed, that combines the critical path tracing algorithms proposed to handle stuck–at [1] and delay faults [13]. The restrictive delay test methodology we use allows fault effect propagation paths to be found without uncertainty. Preliminary results are given to show the effectiveness of the method.

This paper is organized as follows: a description of the circuit model commonly used for synchronous sequential circuits is given in section 2. Section 3 presents a discussion about the delay test methodologies which exist for this kind of circuit. Input data for delay fault diagnosis are described in section 4, and the method is presented in detail in section 5. Results of experiments on ISCAS–89 sequential benchmark circuits are then discussed in section 6, and concluding remarks are given in section 7.

2. Circuit modeling for sequential circuits

A synchronous sequential circuit can be modeled as composed of a combinational part and a memory part. The boundaries of the combinational logic consist of primary inputs (*Data Input*), primary outputs (*Data Output*) and flip–flops (*FF's*). Only primary inputs are controllable and only primary outputs are observable. All flip–flops in the memory part are synchronized by a common clock signal of a given frequency. In this paper, we refer to the *single gate delay fault model* and to the gate level description of the circuits. We assume that all flip–flops in a circuit are D–type flip–flops. We refer to inputs and outputs of flip–flops as *pseudo–outputs* and *pseudo–inputs* of the circuit (respectively). We associate faults with inputs and outputs of gates and with primary inputs and primary outputs.

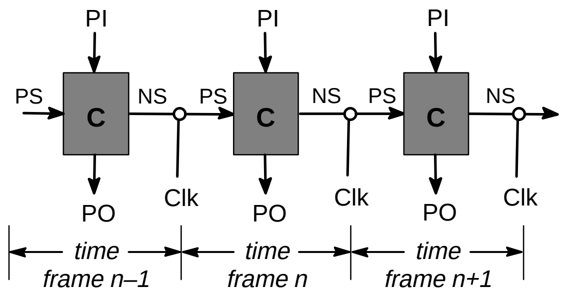


Figure 1: Iterated array model for a sequential circuit

A mapping of the time domain behavior into space domain is depicted in Figure 1. Each copy of the circuit represents the behavior of the circuit during one **time frame**. A time frame is one clock period. Thus, a sequential circuit is handled as the concatenation of time frames in an **iterative logic array** composed of several identical copies of the combinational logic. In figure 1, PI denotes the primary inputs, PO the primary outputs, PS the present state lines (pseudo–inputs) and NS the next state lines (pseudo–outputs). The clock Clk transfers the next state signals to the present state input of the next block.

3. The delay test methodology

In synchronous sequential circuits, a test for delay faults consists of three phases that perform (1) initialization, (2) fault activation and (3) fault effect propagation. **Initialization** brings the circuit to the appropriate initial state. It may require several input patterns, all applied at a clock speed (called a *slow clock*) that allows all signals in the circuit to stabilize before the next state variables are latched into the flip–flops. After initialization, a single input pattern is applied to **activate** the fault and exhibit the fault effect at a primary output or next state line. The effect of this input pattern is sampled at the end of the normal clock period, called the *fast clock*. Depending on whether or not a delay fault exists, the fault effect can be observed on a primary output or latched into a flip–flop. In the latter case, it has to be propagated. **Fault effect propagation** is achieved by applying additional input patterns using a slow clock again. Fault effects latched into flip–flops are thus propagated through a number of time frames until they reach a primary output. Except for fault activation, the circuit is assumed to operate in a fault–free manner (from a timing point of view) as the slow clock period is chosen to ensure that all signal values stabilize in the circuit, even if it suffers from delay faults. This testing methodology, called a *slow clock methodology,* is the simplified testing scheme we use in this paper. The combination of slow and fast clock cycles under which a test sequence is applied is called a *clocking scheme*.

Of course, this testing methodology has a main drawback compared with a method that uses consecutive fast clocks during a test sequence, called an *at–speed* testing methodology. Typically, as activation of faults is allowed only in the fast clock cycle, a large number of test sequences may result. On the other hand, correct activation and correct propagation cannot be predicted when at–speed testing is used. For example, as signals are not guaranteed to stabilize before the end of the fast clock cycle (it depends on whether or not a delay fault is activated during the clock period), unpredicted values may be latched into the memory elements during activation or propagation. Therefore, the use of consecutive fast clocks would require restrictive assumptions on signal values to provide reliable results [3,15].

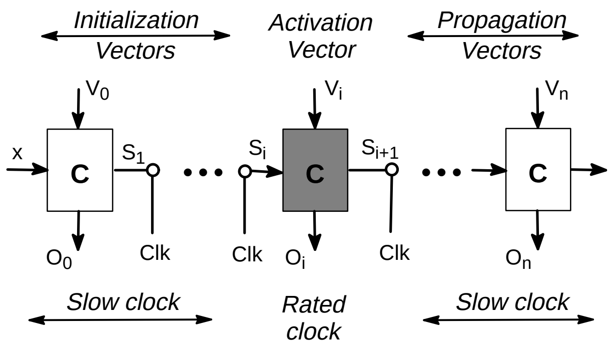


Figure 2: Clocking scheme for delay fault testing

The major elements of the slow clock methodology are illustrated in figure 2. Consider a test sequence (V_0, V_1, ..., V_n) for a gate delay fault in a sequential circuit, applied under a clocking scheme with a single fast clock at time i.

The vectors V_0 through V_{i-1} are called the *initialization vectors*. **The state inputs (pseudo–inputs) prior to the application of V_0 are assumed to be in *don't care* state (x)**. At the end of the initialization sequence, all flip–flops are set into states required by the fault activation vector. Next, the fault is activated by applying V_i and using a fast clock. Concepts of testing for delay faults in combinational or scan–based circuits are applicable in this phase. If the fault is present, faulty values will be latched into the memory elements or observed on the primary outputs. Finally, *propagation vectors* (V_{i+1}, ..., V_n) are applied, again using a slow clock, to propagate the fault effect to the primary outputs.

During initialization and propagation phases, the tests work irrespective of any delay or timing situations that may occur in the circuit. **The propagation of a delay fault through the time frames that use a slow clock can hence be viewed as the propagation of a stuck–at fault.**

4. Input data for delay fault diagnosis

If we consider delay fault diagnosis in a combinational circuit, input data are typically (1) the gate level description of the circuit, (2) the set of test patterns, and (3) the set of failing patterns and failing outputs provided by the tester [11,12]. Output data are a set of potential fault locations.

With respect to the testing scheme described above, using a slow clock methodology, input data to perform delay fault diagnosis in a synchronous sequential circuit are (1) the gate level description of the circuit, (2) the set of test sequences **with the clock cycle in which the fast clock is applied**, and (3) the set of failing patterns and failing outputs provided by the tester. Of course, we assume that a test sequence has been generated by a test pattern generator, and that the clock cycle in which the fast clock is applied is specified [16]. As explained in the previous section, all delay test sequences begin from unknown values on pseudo–inputs. As for combinational circuits, output data are a set of potential fault locations.

When diagnosing a faulty circuit, the aim is to locate the failure that has produced a delayed signal on a primary output during the test. As a delay fault can only be activated during the fast clock cycle (in the assumed testing methodology), the time frames to be considered for fault effect analysis are those comprised between the application of the activation vector and the observation of the delay fault on a primary output. In other words, assuming that V_i is the activation vector (*time frame i*) and that a delayed signal is observed on a primary output after application of the vector V_{i+k} (*time frame i+k*), the number of time frames to be considered during diagnosis is:

$$\textbf{\textit{Time Frames}} = (\boldsymbol{i} + \boldsymbol{k}) - (\boldsymbol{i}) + \mathbf{1} = \boldsymbol{k} + \mathbf{1}$$

The time frames from 0 to i–1 are processed only to define the state input values required by the fault activation vector.

5. The trace–based method

Given the above concepts and a number of test sequences in which a failing vector was found (*failing test sequences*), our method consists of the following steps (these steps are detailed in sub–sections 5.1 and 5.2):

(1) *Good–machine* ***three–valued simulation*** *of each initialization vector (V_0 through V_{i-1})*. The aim is to set pseudo–inputs in states required by the fault activation vector. The values 0, 1 and X are used to simulate the fault–free machine so as to taken into account the unknown values on pseudo–inputs at the beginning of this step. At the end of this phase, we assume that all lines in the circuit have a known value (0 or 1).

(2) *Good–machine* ***four–valued simulation*** *of the activation vector V_i*.

(3) *Good–machine* ***two–valued simulation*** *of each propagation vector (V_{i+1} through V_{i+k})*. Only the values 0 and 1 are required for simulation in this step because the propagation of a delay fault through the slow time frames is the same as that of a logical fault. For a current time frame, the values on primary inputs are given by the propagation vector. The values on pseudo–inputs are given by the final values on pseudo–outputs in the previous time frame.

The signal values on each line in the activation and propagation phases are recorded in order to be used later by the critical path tracing process.

(4) *Pseudo–critical Path Tracing for logic (or stuck–at) faults*. This is a backtracing procedure starting from the failing primary outputs and ending on pseudo–inputs of the circuit in the time frame i+1. The lines which can propagate the fault effect through the slow clock cycles, referred to as *pseudo–critical lines*, are thus identified.

(5) *Critical Path Tracing for delay faults*. This is a backtracing procedure starting from the pseudo–outputs of the circuit in the fast clock cycle (time frame i) and ending on the pseudo–inputs. The potential failing lines for the test simulated in step (2), referred to as *critical lines*, are provided after this phase.

(6) *Set intersection.* Having completed the trace–based part of the algorithm, we carry out a set intersection between the potential failing–line set and the set constructed with critical lines from the current test (of course, we make use of the single fault assumption).

All these steps are repeated for every failing test sequence of the test set. In the next sections, we discuss the value systems required for initialization, fault activation and fault propagation, and we present the trace–based algorithm for sequential circuits.

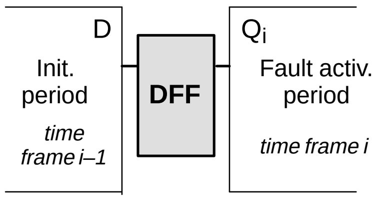
5.1 The multi–valued simulations

Given a failing test sequence composed of several vectors and a gate–level circuit description, we use a three–valued algebra (0,1,X) to simulate the fault–free machine in the **initialization** phase. Except for the first considered time frame, in which the state lines have the value X, the values on pseudo–inputs in a current time frame are computed from the values on pseudo–outputs in the previous time frame, according to the well–known equation of a D flip–flop (Q_{n+1}=D). Note that each line in the circuit has a defined value 0 or 1 at the end of this phase.

The simulation of the fault–free machine in the **activation** phase, performed in a logic–level order, has a four–valued signal representation in two consecutive vectors: S0, S1, U0, U1 [7]. The value S0 (S1) is assigned to signals remaining stable at 0 (1) whatever the delays and timing defects of the circuit may be. The value U0 (U1) is assigned to signals with the final value 0 (1) that may or may not be affected by a transition, depending on gate delays or delay faults. The primitive–gate evaluation tables that give the output values corresponding to input symbols can be found in the literature [12], and therefore are not reproduced here. These tables are used for simulation in the combinational parts of the sequential circuits.

One advantage of such a simulation is that it exhibits transition possibilities even if no transition occurs in the fault–free circuit. Another advantage is that because the simulation is independent of gate propagation delays and delay fault size, it does not need to consider timing specifications.

For the purpose of propagating signal values through the flip–flops during the application of the activation vector, we use a new evaluation table providing the symbolic values on pseudo–inputs in the activation time frame from both the state values and the pseudo–output values in the previous time frame (figure 3).



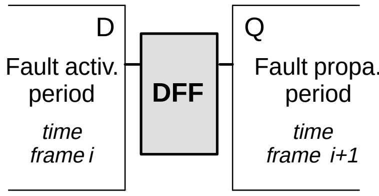
D \ Q_{i-1}	0	1
0	S0	U0
1	U1	S1

Q_i

Figure 3: Evaluation table of D–type flip–flops in the activation period

Remark: As a slow clock is used for simulation with the initialization vectors, pseudo–inputs in the activation time frame are considered to be glitchless.

Although it can be done with the four–valued algebra, only the two boolean values are required to simulate the fault–free machine during the **fault propagation** phase. Pseudo–input values when the first propagation vector V_{i+1} is applied to the circuit are therefore converted into 0 or 1 according to the latch input values (see figure 3). For example, a latch input (pseudo–output) encoded by S0 or U0 during the fault activation period leads to the value 0 on the flip–flop output (pseudo–input) when V_{i+1} is applied. A state assignment table for fault propagation from time frame i (activation) to time frame i+1 (propagation) is given in figure 4. Next, the boolean simulation for the propagation vectors is continued until the failing test vector V_{i+k} is reached.



D	Q_{i+1}
S0 U0	0
S1 U1	1

Figure 4: State assignment for fault propagation

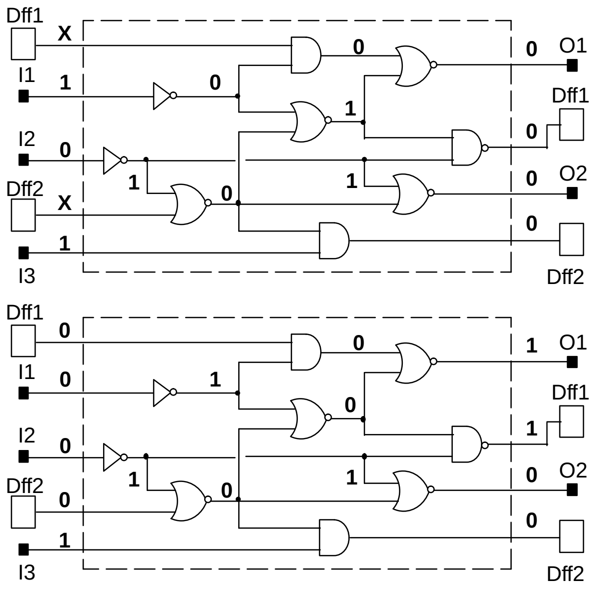


Figure 5.a: Simulation of the initialization vectors

Example: Let us consider the circuit shown in figure 5 with a failing test sequence (V_0=101, V_1=001, V_2=010, V_3=110) applied to its inputs. As regards input data provided by the tester, we assume that a delayed signal was found on the two primary outputs O1 and O2 after the application of vector V_3. In the two first time frame, a slow clock is used, and the three–valued simulation leads to the results shown in figure 5.a. Next, the state values are propagated through the flip–flops according to the evaluation table given in figure 4. The circuit is therefore set in states DFF1=U1 and DFF2=S0 required by the fault activation vector V_2 (figure 5.b). During the fast clock cycle, the multi–valued simulation is performed conventionally. After propagation of the state values through the flip–flops, vector V_3 is applied to primary inputs. As depicted in figure 5.c, only the final values of pseudo–inputs are used to simulate the circuit with the fault propagation vector. Results provided by the boolean simulation are shown in the circuit diagram.

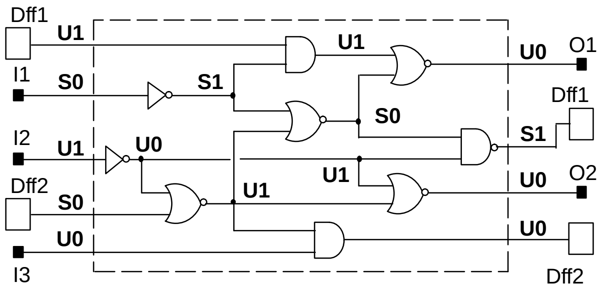


Figure 5.b: Simulation of the fault activation vector

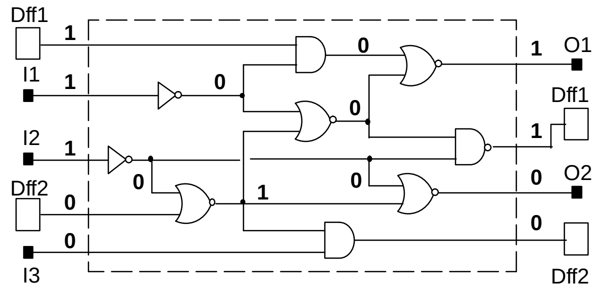


Figure 5.c: Simulation of the propagation vectors

5.2 The Critical Path Tracing

Authors of [1] presented an alternative to stuck–at fault simulation for combinational circuits, referred to as **critical path tracing** (CPT). The main concept in critical path tracing, namely a *critical value*, was first defined by Wang [17]:

- a line *l* has a *critical value x* in a test T, if T detects the fault *l* stuck–at $\overline{x}$,
- a gate input *i* is *sensitive* if complementing the value of *i* complements the gate output value.

The basic algorithm simply traces paths from primary outputs (which are always critical) by recursively marking the sensitive inputs of a gate as critical if its output is critical. This procedure generates critical paths, consisting of lines with critical values. Because critical path tracing did not recognize faults that are detected only by multiple–path sensitization, it was first proposed as an approximate method.

In an extended version of the critical path tracing, a modification of the method was proposed to remove the approximation and make the algorithm exact [14]. Therefore, we decided to use this method to identify fault effect propagation paths in the propagation time frames (applied at a slow clock). In our work, the trace–based algorithm starts **only** from failing primary outputs in time frame i+k, and not from all primary outputs as in [14]. It terminates on pseudo–outputs of the activation time frame i, thus providing a set of critical lines referred to as *pseudo–critical lines*. They are labelled as pseudo–critical because they will not belong to the final set of potential failing lines although they belong to a fault effect propagation path. A distinguishing feature of our approach is that the *sensitivity* of lines [1] is evaluated during the backward instead of forward pass of the process, thus reducing the overall computation time. Critical path tracing through the flip–flops does not need to be explained in detail, since they are considered as transparent cells.

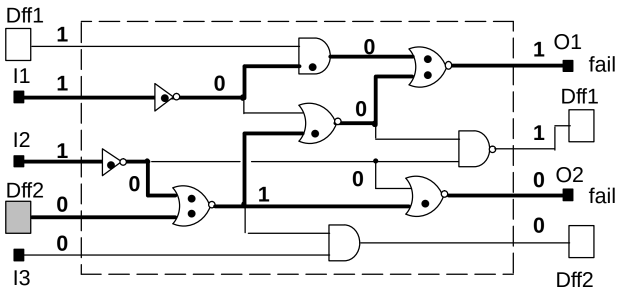


Figure 6: Pseudo–critical path tracing in the propagation time frame

Example: Let us consider again the example given in section 5.1, with two failing outputs O1 and O2 as starting elements for the path tracing (figure 6). The pseudo–critical paths are represented by bold lines, and the sensitive inputs of gates are marked by dots. In our example, pseudo–critical path tracing is stopped on primary inputs of the circuit, and is pursued further only through flip–flop DFF2. As the next time frame to be considered is the activation time frame, the path tracing beyond DFF2 will serve to identify the critical lines for the simulated test sequence.

In a previous study [12], we extended the initial CPT algorithm to deal with delay faults in combinational circuits. As only the activation time frame needs to be considered with a fast clock, it is possible to use the new algorithm to trace critical paths for delay faults in this clock period. The path tracing starts from critical pseudo–outputs reached in the previous step of the process, and tries to extend the critical state of the pseudo–outputs as far as possible towards the primary and pseudo inputs. We consider sensitive lines [13] with the value U0 or U1 to be on a fault effect propagation path (sensitive inputs of gates are marked by dots). Lines with the values S0 or S1 stop the tracing process. The backward pass of the process terminates on the pseudo and primary inputs of the activation time frame. The set of critical lines thus contains the potential failing lines for the simulated test sequence.

As for delay fault diagnosis in combinational circuits, the final set of potential failing lines results from the

intersection of the sets obtained with each failing test sequence.

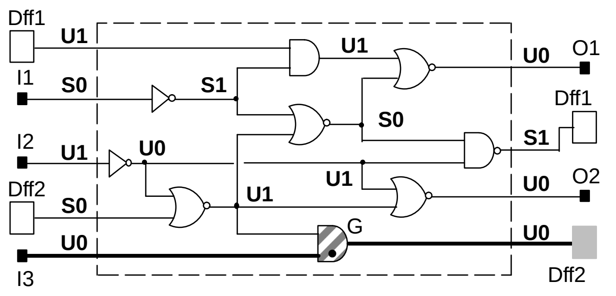


Figure 7: Critical path tracing for delay faults in the activation time frame

Example: Let us consider again the example given in section 5.1, in which only one pseudo–output of the activation time frame was found to be critical during the first part of the tracing process (see figure 7). According to the above description of the path tracing for delay faults, the set of critical lines for the simulated test sequence contains two elements (input and output of gate G6), which are the potential failing lines.

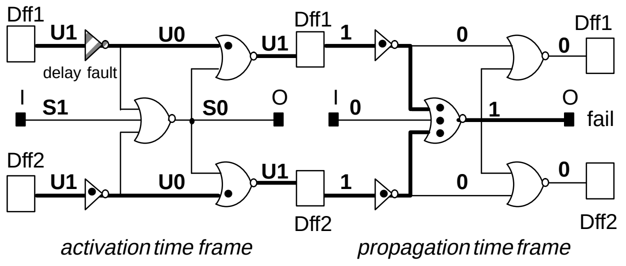


Figure 8: Necessity to do the union of paths coming from critical pseudo–outputs

Remark: In the case of combinational circuits, the set of critical lines which are critical in a test T results from the **intersection** of paths traced from every failing output (with the single fault assumption). A new feature of the presented method is that the **union** of paths traced from every critical pseudo–output is carried out, instead of the intersection of these paths. An example is given in figure 8, in which a delay fault is sensitized during the activation phase and appears on the primary output O during the propagation phase. According to the path tracing algorithm described above, only disjoint critical paths are found in the activation time frame. We therefore ascertain that only the union of critical paths contains the delay fault location.

6. Results of experiments

Results of delay fault diagnosis in sequential circuits are discussed in this section. As for combinational circuits, the resolution of delay fault diagnosis in sequential circuits greatly depends on the test sequences applied to the machine and on the delay defect size. For example, different test sequences applied to a faulty machine will lead to different numbers of failing test sequences, and therefore to different diagnostic resolutions. So, results presented in this section are variable, and may vary according to the parameters cited above.

In the experiments performed, each gate is assumed to have a unit delay, and each line is considered as a potential fault location. As we do not possess any sequential ATPG program for delay faults, the test sequences applied to the circuits were obtained using SETA, a sequential ATPG tool for stuck–at faults provided by the Politecnico di Torino [5]. Thus, results presented for each circuit would probably be better with test sequences specifically generated for delay faults. In order to obtain the list of failing test sequences with the failing primary outputs, which is normally provided by the tester for real faulty circuits, we simulated each circuit with random delay defects expressed in terms of unit delay. Simulations were performed using the software Genrad's System–Hilo: HISIM.

s208 30 test sequences	**# failing test seq.**	**# pot. failing lines**	**Diag. resol. (%)**	**CPU time (s)**
gate 58 / 1u	2	11	5.28	0.81
gate 104 / 3u	5	6	2.88	1.21
gate 15 / 7u	14	6	2.88	22.17
gate 75 / 1u	2	11	5.28	0.81

s510 77 test sequences	**# failing test seq.**	**# pot. failing lines**	**Diag. resol. (%)**	**CPU time (s)**
gate 24 / 2u	77	7	1.3	20.66
gate 197 / 2u	77	12	2.35	21.06
gate 113 / 2u	77	7	1.3	19.65
gate 188 / 5u	77	15	2.94	19.99

s1494 167 test sequences	**# failing test seq.**	**# pot. failing lines**	**Diag. resol. (%)**	**CPU time (s)**
gate 68 / 5u	161	8	0.53	248.53
gate 405 / 6u	13	5	0.33	226.42
gate 470 / 5u	8	104	6.9	227.24
gate 27 / 5u	4	40	2.6	226.58

Tables 1, 2 and 3: Results of experiments on three benchmark circuits

The diagnostic algorithm has been implemented in the programming language C++, and all experiments were carried out on a SUN–sparc10 workstation. Results obtained for three ISCAS–89 sequential benchmarks [4] are presented in Tables 1, 2 and 3. For each circuit, we have assumed random delay faults, the location and size of which are specified in the first column of the tables. The second column lists the number of failing test sequences obtained after simulation with HISIM. Results provided by the diagnosis process are given in the next column, in terms of the number of potential failing lines. The percentages presented in the fourth column represent the number of potential failing lines over the total number of lines in the circuit, defined as "diagnostic resolution". The last column gives the CPU time required to obtain diagnostic results.

These results show the performance of the diagnosis process, which always identifies less than 6% of the total number of lines as potential failing elements. In addition, the set of potential failing lines always contains the delay fault to be diagnose, demonstrating the reliability of the method. We emphasize that these results would be better with test sequences generated specifically for delay faults.

7. Summary and conclusion

Until now, all the work reported on delay fault diagnosis has focused on combinational circuits. If one assumes that a slow clock delay test methodology is used, we propose a method for diagnosing gate delay faults in synchronous sequential circuits. This method is based on a path tracing algorithm performed from results of a fault–free circuit simulation. Propagation of delay faults through the combinational and memory parts of a sequential circuit is processed by taking advantage of the existing diagnostic methods developed for stuck–at and delay faults. Experimental results are given on three ISCAS–89 benchmark circuits to provide a measure of the diagnostic resolution. Although we can obtain better results, the method seems to be very efficient with a slow clock methodology as the basis for the test. Our goal now is to diagnose sequential circuits with any type of delay test methodology (At–speed testing).

Acknowledgements

This work is supported by the European Community under the ESPRIT III Basic Research 6575 ATSEC contract. We would like to thank P. Prinetto and his team from the Politecnico di Torino who provided us with their sequential ATPG tool SETA.

References

[1] M. Abramovici, P.R. Menon and D.T. Miller, "***Critical Path Tracing – An Alternative to Fault Simulation***", Proc. 20th Design Autom. Conf., pp. 214–220, June 1983.

[2] P. Agrawal, V.D. Agrawal and S.C. Seth, "***A New Method for Generating Tests for Delay Faults in Non–Scan Circuits***", Proc. of Int. Conf. on VLSI Design, pp. 4–11, January 1992.

[3] S. Bose, P. Agrawal and V.D. Agrawal, "***A Path Delay Fault Simulator for Sequential Circuits***", Proc. of Int. Conf. on VLSI Design, pp. 269–274, January 1993.

[4] F. Brglez, D. Bryan and K. Komminiski, "***Combinational Profiles of Sequential Benchmark Circuits***", Proc. of Int. Symp. on Circuits and Systems, pp. 1929–1934, May 1989.

[5] P. Camurati, F. Corno, P. Prinetto and M. Sonza Reorda, "***A Simulation Based Approach to Test Pattern Generation for Synchronous Sequential Circuits***", Proc. of VLSI Test Symposium, pp. 263–267, November 1992.

[6] T.J. Chakraborty, V.D. Agrawal and M.L. Bushnell**, "*Delay Fault Models and Test Generation for Random Logic Sequential Circuits***", Proc. of 29th Design Auto. Conf., pp. 165–172, June 1992.

[7] S.T. Chakradar, M.A. Iyer and V.D. Agrawal**, "Energy Minimization Based *Delay Testing***", Proc. of Euro. Design Auto. Conf., pp. 280–284, March 1992.

[8] K.T. Cheng, "***Transition Fault Simulation for Sequential Circuits***", Proc. of Int. Test Conf., pp. 723–731, October 1992.

[9] S. Devadas, "***Delay Test Generation for Synchronous Sequential Circuits***", Proc. of Int. Test Conf., pp. 144–152, September 1989.

[10] D. Dumas, P. Girard, C. Landrault and S. Pravossoudovitch, "***An Implicit Delay Fault Simulation Method with Approximate Detection Threshold Calculation***", Proc. of Int. Test Conf., pp. 705–713, October 1993.

[11] D. Dumas, P. Girard, C. Landrault and S. Pravossoudovitch, "***Effectiveness of a Variable Sampling Time Strategy for Delay Fault Diagnosis***", To appear in the Proc. of Euro. Design & Test Conf., February–March 1994.

[12] P. Girard, C. Landrault and S. Pravossoudovitch, "***A Novel Approach to Delay Fault Diagnosis***", Proc. 29th Design Autom. Conf., pp. 357–360, June 1992.

[13] P. Girard, C. Landrault and S. Pravossoudovitch, "***A Reconvergent Fanout Analysis for the CPT Algorithm used in Delay Fault Diagnosis***", Proc. of Euro. Test Conf., pp. 83–88, April 1993.

[14] P.R. Menon, Y. Levendel and M. Abramovici, "***Critical Path Tracing in Sequential Circuits***", Proc. of Int. Conf. on CAD , pp. 162–165, November 1988.

[15] I. Pomeranz and S.M. Reddy, "***At–Speed Delay Testing of Synchronous Sequential Circuits***", Proc. 29th Design Autom. Conf., pp. 177–181, June 1992.

[16] I. Pomeranz, L.N. Reddy and S.M. Reddy**, "*SPADES: A Simulator for Path Delay Faults in Sequential Circuits*",** Proc. of EURO–DAC , pp. 428–434, September 1992.

[17] D.T. Wang**, "*Properties of Faults and Criticalities of Values Under Tests for Combinational Networks*",** IEEE Trans. on Computers, Vol. C–24, no. 7, pp. 746–750, July 1975.