Unified Framework for Logic Diagnosis

Alexandre Rousset, Patrick Girard, Christian Landrault, Serge Pravossoudovitch, Arnaud Virazel

Unified Framework for Logic Diagnosis

A. Rousset, P. Girard, S. Pravossoudovitch, C. Landrault, A. Virazel
Laboratoire d'Informatique de Robotique et de Microélectronique de Montpellier
UMR 5506 UNIVERSITE MONTPELLIER II / CNRS
161 rue Ada, 34392, Montpellier Cedex 05, FRANCE

Abstract

This paper presents a unified diagnosis method targeting most of the fault models used in practice today. This framework is intended to be used to diagnose faulty behaviors in nanometric circuits for which the classical stuck-at fault model is far to cover all realistic failures. The method is based on an Effect-Cause approach which relies on the two following main operations. The first one is based on critical path tracing (CPT) [4] and consists in identifying critical lines in the Circuit Under Test (CUT) which can be the source of observed errors. The second one consists in allocating a set of possible fault models to each critical line, so that root causes of failures can be finally determined. The main advantage of this method is that it does not need to explicitly consider each fault model during the diagnosis process.

1. Introduction

Failure analysis is an important operation, which may impact the circuit design and the fabrication process and has a growing role in fast yield ramping. Failure analysis first relies on a logical diagnosis aiming at reducing the potential faulty subparts of the circuit before using more sophisticated physical tools in order to precisely locate and identify the defect. This logical diagnosis process is based on the knowledge of the structure of the circuit, the applied test vectors and the responses to these vectors provided by the tester (test data log).

The objective of logic diagnosis is to locate and identify failures leading to a CUT erroneous behaviour. Information provided by the diagnosis process is therefore used to guide the circuit physical observation during failure analysis. Thus, the efficiency of the failure analysis depends on the resolution of the diagnosis process.

There are two types of diagnosis approaches, namely *Cause-Effect* and *Effect-Cause*. The *Cause-Effect* analysis is based on fault simulation. Using fault simulation, one can built a fault dictionary on which all the succeeding steps of the logic diagnosis process will depend. The main advantage of such an approach is to be able to handle both combinational and sequential circuits in the same way, even for circuits not equipped with specific DFT structures. On the other hand, the *Cause-Effect* approach exhibits some drawbacks. The first one is the need to have an accurate description of the fault models used with the associated failure effects. If such an accurate description is manageable for static faults, this is no longer the case for dynamic faults like delay faults and/or parametric faults such as resistive shorts and opens. The second drawback is related to the huge data volume that should be generated by the fault simulator particularly for large industrial circuits.

The *Effect-Cause* approach looks more interesting as the diagnosis process is performed starting from the faulty responses. Generally, an *Effect-Cause* approach uses a backtracking process originating from the CUT outputs, such as the Critical Path Tracing (CPT) process [1]. The *Effect-Cause* approach is particularly interesting for the diagnosis of failures with dynamic and/or parametric effects. Its application and use for sequential circuits is not so easy even if researches have been done in this direction [13]. Nevertheless, its most efficient use concerns full-scan circuits. In this case, the diagnosis process can be divided into two main steps. The first one consists in identifying faulty scan cells from the external outputs of the CUT on which one or several errors have been observed during test. Potential problems introduced by compaction logic (masking, unknown value) have to be considered during this step [2][3]. The second step consists in identifying suspected defects (fault locations and fault types) in the combinational part of the CUT. This paper deals with diagnosis in the combinational part of the CUT.

The *Effect Cause* approach is based on a CPT process initially developed to diagnose stuck-at faults [1][4]. Due to advances in manufacturing technologies, process variations and more aggressive clocking strategies more and more lead to failures which can no longer be modelled by classical stuck-at faults.

Numerous actual failures exhibit timing or parametric behaviours which are not represented by stuck-at faults. Such failures have to be taken into account in the testing process in order to reach acceptable DPM (Defect Per Million) figures. This is the role of dedicated delay fault testing procedures and IDD related approaches. Consequently, the CPT process has been extended and adapted to handle other fault models such as delay faults [5][6], or short faults [7]. However, a common feature of all the methods proposed so far is that they handle one single fault model at a time or scarcely two fault models when the induced effects are similar [8][9][10]. The problem is that when an error is observed during test, it does not exist any deterministic information about the defect which has caused this error and hence there is any knowledge of the fault model to be used a priori for the diagnosis process. As considering each fault model explicitly is not a viable solution, there is a need to develop a unified framework for fault diagnosis.

This paper presents such a unified diagnosis framework. The method is based on an *Effect-Cause* approach which relies on the two following main operations. The first one is based on CPT and consists in identifying critical lines in the CUT which can be the source of observed errors. The second one consists in allocating a set of possible fault models to each critical line, so that root causes of failures can be finally determined. The main advantage of this method is that it does not need to explicitly consider each fault model during the diagnosis process.

The rest of the paper is organized a follows. In section 2, the fault models that can be associated with a given line are described in details. In section 3, we describe the CPT process as well as the fault model allocation procedure. Section 4 concludes this paper.

2. Fault models identification associated with an error

Circuit defects are the result of numerous problems (localised defects, process variations, etc.) and lead to different behaviours (logical error, delay, electrical parameters deviation, etc.). To be handled efficiently by fault simulation tools, test pattern generators or diagnostic tools, these faulty behaviours have been represented by fault models. These models may represent logical deviations (stuck-at, short, open), timing deviations (gate/path delay fault) or parametric errors (resistive short, resistive open). On the other hand, some defects affect the static behaviour of the CUT while some others affect the dynamic behaviour. Faults which affect the dynamic behaviour, such as delay faults or transistor stuck-open faults, require a pattern sequence to be detected. In this study, we consider the following classical fault models: stuck-at, short (AND/OR bridging), resistive short, open, resistive open, gate delay, and by extension path/segment delay, stuck-on and stuck open faults.

When an error is located on a given line L, this error can be caused by a defect affecting the line itself or is due to the propagation of an upstream error. In the first case, we need to know the fault-free circuit behaviour to determine the model to be associated with this defect. Thus, if the expected value on line L is a logic 0, the following models can be suspected for the observed error (Figure 1a):
Stuck-at 1 of line L,
Short (OR bridge between line L and another line at logic 1),
Open (with open line load at 1),
Internal fault inside the upstream gate (transistor N stuck-open or transistor P Stuck-on),
Resistive open on L or StF (Slow to Fall) delay fault (if the applied pattern produces a transition on line L),
Resistive short (with all lines at 1, if the applied vector produces a transition on line L).
Similarly, if the expected value on line L is a logic 1, the following models can be suspected for the observed error (figure 1b):
Stuck-at 0 of line L,
Short (AND bridge between line L and another line at logic 0),
Open (with open line load at 0),
Internal fault inside the upstream gate (transistor P stuck-open or transistor N Stuck-on),
Resistive open on L or StR (Slow to Rise) delay fault (if the applied pattern produces a transition on line L),
Resistive short (with all lines at 0, if the applied vector produces a transition on line L).

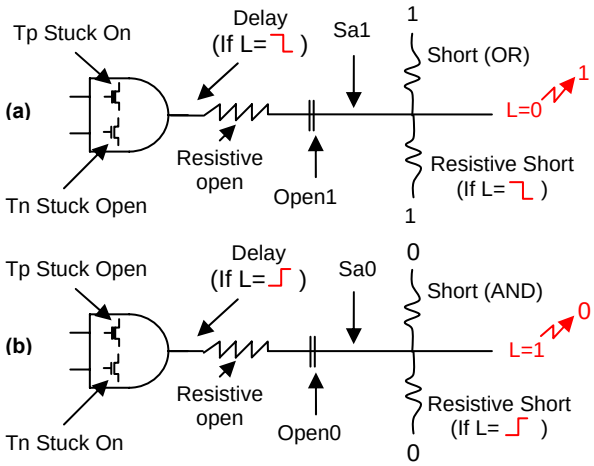

Figure 1. Potential fault models associated with an error on a given line

As it can be seen on Figure 1, faults which affect a faulty line can be determined assuming the knowledge of: (i) the expected value on this line, (ii) the values on others lines (short fault), and, (iii) the transition induced on this line by the applied pattern (dynamic fault).

Thus, all the possible fault models associated with a given logical error can be deduced according to the fault-free signals on this line as shown in Figure 2. In this table, we consider both the pattern V_i producing the error and the previous pattern V_{i-1}.

Among the fault models considered in Figure 2, we can see that several models are equivalent in terms of induced errors. For example, this is the case for open faults and stuck-at faults. Furthermore, it is worth noting that resistive shorts represent a subset of short faults. In fact, resistive short faults require the same conditions than short faults with an additional condition on signal transition. Compared to delay faults which have the same behaviour, resistive short faults imply the knowledge of the *aggressor* line, i.e., the line shorted to the faulty node.

	$V_{i-1}V_i$			
	00	11	10	01
Stuck at 0	—	x		x
Stuck at 1	x		x	
Tn Stuck open (*)	x		x	
Tn Stuck on (*)		x		x
Tp Stuck open (*)		x		x
Tp Stuck on (*)	x		x	
Open 0		x		x
Open 1	x		x	
Resistive open			x	x
Short Or (with any line at 1)	x		x	
Short And (with any line at 0)		x		x
Resistive Short (with any line at 1)			x	
Resistive Short (with any line at 0)				x
Delay StF			x	
Delay StR				x
Delay StR & StF			x	x

Figure 2. Fault models associated to a given error

(*) Stuck-on and Stuck-open faults produce an error if sequential conditions are fulfilled.

3. Unified diagnosis process

The proposed diagnosis process is based on an *Effect-Cause* approach which relies on the two following main operations. The first one is based on a CPT process and consists in identifying critical lines in the CUT. The CPT process uses i) a fault-free circuit simulation, ii) the critical path tracing operation itself, iii) an intersection procedure between critical paths. The second one consists in allocating a set of possible fault models to each critical line, so that root causes of failures can be finally determined. The data needed for the completion of this process are the following: (i) the gate level circuit description, (ii) the test pattern list, (iii) the subset of *faulty* patterns which exhibit an error during test, and iv) the corresponding erroneous outputs.

3.1. Fault-free simulation and signal encoding

As previously mentioned, we need to know the logic value on a given line as well as its previous value (transition for delay fault, sequential effect for stuck-on and stuck-open fault ...). So, in order to make transition propagation possibilities appear during fault-free circuit simulation, we use a six-valued logic simulation based on the H6 algebra [14]:

C0 : static 0 = 00
C1 : static 1 = 11
R1 : rising transition = 01
F0 : falling transition = 10
P0 : static 0-hazard = 010
P1 : static 1-hazard = 101

C0 (C1): Static 0 (1), represents a signal remaining absolutely stable at 0 (1) (whatever gate propagation delays and timing defects of the circuit may be).

F0 (R1): Fall (Rise), represents a signal with the initial value 1 (0) and the final value 0 (1) (after circuit stabilization)

P0 (P1): Pulse 0 (Pulse 1), represents a signal with the same initial and final value 0 (1), but with possible transitions to 1 (to 0) according to circuit timing parameters or delay faults.

Such an algebra allows an efficient encoding of the logic values produced by the faulty test pattern, as well as an encoding of the transitions (R1, F0) and possibilities of transition (P0,P1) on a given line, without the need of any timing analysis.

At the beginning of the fault-free simulation, the signals to be applied on the circuit's inputs are determined from the *faulty* pattern V_i, and from the previous pattern V_{i-1}, applied just before V_i. Thus, for a given input E, the values to be associated are the following:

$V_{i-1}(E) = 0,\ V_i(E) = 0 \Rightarrow E = C0$
$V_{i-1}(E) = 1,\ V_i(E) = 1 \Rightarrow E = C1$
$V_{i-1}(E) = 0,\ V_i(E) = 1 \Rightarrow E = R1$
$V_{i-1}(E) = 1,\ V_i(E) = 0 \Rightarrow E = F0$

Having determined all the input values, the simulation process consists in propagating these values towards the circuit outputs thanks to propagation tables associated to each logic gates. Propagation tables for classical AND, OR, NOT gates are shown in figure 3. All other propagation tables can be easily obtained from these three basic ones.

AND	C0	C1	F0	R1	P0	P1
C0	C0	C0	C0	C0	C0	C0
C1	C0	C1	F0	R1	P0	P1
F0	C0	F0	F0	P0	P0	F0
R1	C0	R1	P0	R1	P0	R1
P0	C0	P0	P0	P0	P0	P0
P1	C0	P1	F0	R1	P0	P1

OR	C0	C1	F0	R1	P0	P1
C0	C0	C1	F0	R1	P0	P1
C1	C1	C1	C1	C1	C1	C1
F0	F0	C1	F0	P1	F0	P1
R1	R1	C1	P1	R1	R1	P1
P0	P0	C1	F0	R1	P0	P1
P1	P1	C1	P1	P1	P1	P1

IN	OUT
C0	C1
C1	C0
F0	R1
R1	F0
P0	P1
P1	P0

Figure 3. Propagation tables of basic gates

For example, let's consider the circuit in Figure 4. It has 4 inputs (e_1, e_2, e_3, e_4) and 2 outputs (z_1, z_2). The pattern pair applied on the circuit inputs is the following: V_1=(0,0,1,0), V_2=(1,1,1,1). From these two patterns, we first deduce the six-valued input vector V= (R1, R1, C1, R1). Then, the values propagated during the fault-free simulation process are showed in Figure 4.

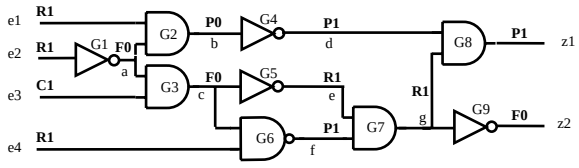

Figure 4. Multi-valued simulation

3.2. Critical path tracing

Critical path tracing has been initially proposed to handle stuck-at faults [4], and was further adapted to handle other fault models such as timing faults [6][11][12], short faults [7], or stuck-open faults [9]. In our approach, the path tracing process is not directly associated with any fault model. In fact, our first target is to determine all the critical lines, i.e. all lines from which a logic error can induced the faulty behaviour observed at the CUT outputs. The combination of fault models with critical lines will be done in a second step. This process can rely on CPT algorithm developed to stuck-at faults [4].

For example, this CPT process is illustrated in Figure 5 which is based on the 6-valued simulation performed on the circuit in Figure 4. The sensitive inputs of each gate (highlighted with a black dot) are defined from final logic values of each signal and with rules provided in [5]. The CPT process begins from each *faulty* output z_1 and z_2. The sets of critical lines obtained from z_1 and z_2 are respectively {e_2, a, b, c, d, e, f, g, z_1} and {e_2, a, c, e, f, g, z_2}.

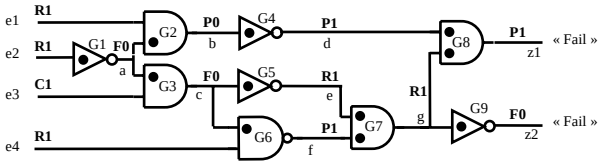

Figure 5. Critical path tracing

For each critical line, the CPT process memorizes the symbol on this line. After the CPT ending and the associated intersection process which will be described later on in the next paragraph, this associated value will allow us to determine the fault model(s) to be associated with a given critical line. Thus, the information provided at the end of the CPT process is a list of pairs (LC, S) where LC is the critical line name, and S is the symbol associated with this critical line. For the example in figure 5, the CPT process applied to z_1 and z_2 produces the two following lists associated with z_1 and z_2 respectively {(e_2,R1), (a,F0), (b,P0), (c,F0), (d,P1), (e,R1), (f,P1), (g,R1), (z_1,P1)} and {(e_2,R1), (a,F0), (c,F0), (e,R1), (f,P1), (g,R1), (z_2,F0)}.

3.3 Identification of critical lines

The CPT process is repeated for all *faulty* outputs observed during test. For each *faulty* output, a critical line list is created with for each line, the symbol associated. All potential sites are presents in these lists. Now, if we assume the single fault assumption, then the fault location is necessarily present in all lists. With this assumption and to determine the possible fault locations, we can do the intersection of all the lists produced by the CPT process. The intersection operation is defined as follows:

Let's take two lists L_1 and L_2 supplied by CPT process and $L_s = L_1 \cap L_2$. Ls is defined as the result of all the intersections between each pair $(LCi,Si)_1$ of list L_1 and each pair $(LCj,Sj)_2$ of list L_2. The intersection of two pairs (LCi,Si)1 $\cap$ (LCj,Sj)2 is defined as follows:

- If LCi $\neq$ LCj or Si $\cap$s Sj not defined, then (LCi,Si)1 $\cap$ (LCj,Sj)2 = $\varnothing$
- else LC=LCi=LCj then $(LC,Si)_1 \cap (LC,Sj)_2$ = (LC, Si $\cap$s Sj)

with the intersection operation ∩s between symbols of two critical lines defined in figure 6.

∩s	C0	C1	F0	R1	P0	P1	D
C0	C0	-	C0	-	C0	-	-
C1	-	C1	-	C1	-	C1	-
F0	C0	-	F0	D	F0	D	D
R1	-	C1	D	R1	D	R1	D
P0	C0	-	F0	D	P0	D	D
P1	-	C1	D	R1	D	P1	D
D	-	-	D	D	D	D	D

Figure 6. Intersection operation between signals

The D symbol, added to the others symbols used for the simulation, enables to represent a falling *or a* rising commutation. This symbol allows characterizing a temporal fault acting on both commutation types.

The - symbol represents an undefined intersection. In this undefined case and with the single fault assumption, the error observed on this line cannot be the result of a defect (through its fault model) affecting this line but only the propagation of an upstream error. As a consequence, the line can be removed from the list of the possible defect location.

In order to illustrate this intersection process, let's take again the results obtained from the figure 5 assuming that the two outputs z_1 and z_2 are faulty. From these two faulty outputs, we have the two following list of *critical* pairs:

L_1 = {(e_2,R1), (a,F0), (b,P0), (c,F0), (d,P1), (e,R1), (f,P1), (g,R1), (z_1,P1)},

L_2 = {(e_2,R1), (a,F0), (c,F0), (e,R1), (f,P1), (g,R1), (z_2,F0)}.

Using the rules defined previously, the intersection between L_1 and L_2 gives the list L_{12}:

L_{12} = {(e_2,R1), (a,F0), (c,F0), (e,R1), (f,P1), (g,R1)}

Now, if we assume that another *faulty* pattern V_j gives a *fault-free* output z_1 and a *faulty* output z_2. Assuming also that associated with its previous pattern V_{j-1}, this gives us the multi-valued pattern {R1,C0,R1,R1}, figure 7 summarizes the corresponding simulation.

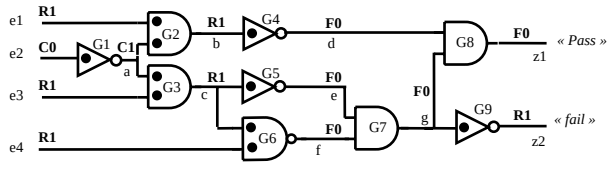

Figure 7. Critical path tracing

The CPT process starting from output z_2 gives the following critical lines list L_3:

L_3 = {(e_2,C0),(e_3,R1),(a,C1),(c,R1),(g,F0),(z_2,R1)}

In this case, we can notice that the CPT process stops at gate G7, defined as *non sensitive*, then resumes at fanout c feeding the two inputs of this non sensitive gate. The details of such a mechanism are described and can be found in [4].

Applying again the previous intersection rules, the result of intersection between L_3 and L_{12} is:

L_{123} = {(c,D),(g,D)}

Although the critical lines e_2 and a belong to the three initial lists, they are not included in the final list L_{123}. This is due to indefinite intersection between symbols associated with e_2 and a. Moreover, lines c and d get the D symbol, because these lines are critical for both falling and rising transitions.

3.4 Fault model allocation

At this stage of the diagnosis process, we have in hand a reduced list of critical lines for each of which one symbol is associated with. According to this symbol, we can associate fault models with each critical lines as described in the table of figure 8. This table is an adaptation of the one given in figure 2 taking into account the different symbols manipulated during the CPT process.

	C0	C1	F0,P0	R1,P1	D
Stuck at 0		x		x	
Stuck at 1	x		x		
Tn Stuck open (*)	x		x		
Tn Stuck on (*)		x		x	
Tp Stuck open (*)		x		x	
Tp Stuck on (*)	x		x		
Open 0		x		x	
Open 1	x		x		
Resistive open			x	x	x
Short Or (with any line at 1)	x		x		
Short And (with any line at 0)		x		x	
Resistive Short (with any line at 1)			x		
Resistive Short (with line any at 0)				x	
Delay StF			x		
Delay StR				x	
Delay StR & StF			x	x	x

Figure 8. Fault models according to symbol associated to critical line

To illustrate this fault model association process, let take the example used in the previous paragraph leading to the final list L_{123} = {(c,D),(g,D)}. In this case, the only possible faults associated with critical lines c and g are:

line c: resistive open, Delay (StR & StF)
line g: resistive open, Delay (StR & StF)

3.5. Using fault-free outputs to static faults screening

In the same way that done in [4], it is possible to use the fault-free outputs in the CPT process in order

to screen critical lines al least for certain kinds of faults. As example, let's consider the example of figure 9.

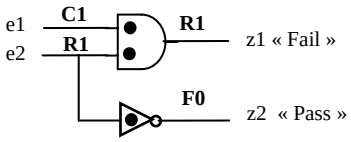

Figure 9. Static fault screening

Assuming that the critical line list provided from faulty output z_1 is:

$$L_1 = \{(e_1,C1),(e_2,R1),(z_1,R1)\}$$

With the previous allocation rules shown in figure 8, the faults associated with the critical line e_2 are (see figure 8, column R1,P1): Sa0, Open0, Resistive open, AND short (with a line at 0 logic value), StR, StR & StF.

If we analyze the behaviour of such faults, it is easy to show that all faults with a static effect (i.e. Sa0, Open0, AND short (with a line at 0)) cannot cause the observed error on output z_1.

In fact, if such a static fault is acting on line e_2, then an error would have been propagated to the output z_2. On the contrary, faults with a dynamic effect, i.e. faults with a transition delay (Resistive open, Resistive short (with a line at 0), StR, StR & STF), this is not the case. In fact, one of these faults can affect line e_1, and be propagated and observed on z_1, without any effect on z_2 according to the timing or electrical parameters of the CUT, and to the fault size.

Thus, a fault free output analysis can improve the diagnosis accuracy by static faults screening in the final list. As in [4], this screening process based on the use of fault-free outputs can be based on a path tracing principle adaptation.

4. Conclusion

The proposed diagnostic method relies on an *effect cause* approach based on CPT process. It enables to work without manipulating explicitly the fault models. The potential faults leading to the observed dysfunctions on the outputs are determined solely from an analysis of the effects (errors) produced on circuit lines. In comparison with previous methods developed on the CPT principle, this approach will allow handling a more comprehensive and realistic set of fault models in an unified manner thus improving the diagnosis accuracy and then its overall quality.

5. References

[1] M. Abramovici, P.R. Menon, D.T. Miller, "Critical Path Tracing – An Alternative to Fault Simulation", IEEE Design & Test of Computers, Vol 1, n°1, pp 83-92, February 1984.

[2] S. Mitra, K.S. Kim, "X-Compact, an Efficient Response Compaction Technique for Test Cost Reduction", International Test Conference, pp. 311-320, 2002.

[3] G.Mrugalski, A. Pogiel, J. Rajski, J. Tyszer, C. Wang, "Fault Diagnosis in Designs with Convolutional Compactors", International Test Cconference, pp. 498-507, 2004.

[4] M. Abramovici, M. A. Breuer, "Multiple Fault Diagnosis in Combinational Circuits Based on an Effect-Cause Analysis", IEEE Transactions on Computer, vol. c-29, n°6, pp. 451-460, June 1980.

[5] P. Girard, C. Landrault, S. Pravossoudovitch, "An Advanced Diagnostic Method for Delay Faults in Combinational Faulty Circuits", Journal of Electronic Testing: Theory and Applications, vol. 6, n°3, pp. 277-293, 1995.

[6] Yuan Chieh Hsu, Sandeep Gupta, "A New Path-Oriented Effect-Cause Methodology to Diagnose Delay Failure", International Test Conference, pp. 758-767, 1998.

[7] Srikanth Vendkataraman, W. Kent Fuchs, « A Deductive Technique for Diagnosis of Bridging Fault », Internationel Conference on Computer Aided Design, pp. 562-567, 1997.

[8] Xinyue Fan, Will Moore, Camelia Hora, Guido Gronthoud, "A novel Stuck-At Based Method for Transistor Stuck-Open Fault Diagnosis", International Test Conference, session 16, pp. 1-4, 2005.

[9] David B. Lavo, Brian Chess, Tracy Larrabee, F. Joel Fergusson, "Diagnosing Realistic Bridging Faults with Single Stuck-At Information", IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, vol.17, n°3, pp. 255-267, March 1998.

[10] Piet Engelke, Ilia Polian, Michel Renovell, Bernd Becker, "Simulating Resistive Bridging and Stuck-At Faults", International Test Conference, pp. 1051-1059, 2003.

[11] P. Girard, C. Landrault, S. Pravossoudovitch, "A Novel Approach to Delay Fault Diagnosis", ACM Design Automation Conference, pp. 357- 360, 1992.

[12] P. Girard, C. Landrault, S. Pravossoudovitch, "Delay Fault Diagnosis Based on Critical Path Tracing from Symbolic Simulation", ISCAS92, IEEE International Symposium on Circuits and Systems, vol. 3 of 6, pp.1133-1136, 1992.

[13] P. Girard, C. Landrault, S. Pravossoudovitch, B. Rodriguez, "Diagnostic of Path and Gate Delay Faults in Non-Scan Sequential Circuits", VTS'95: 13th IEEE VLSI Test Symposium, Princeton, USA, May 1-3, 1995, pp 380-386.

[14] J.P. Hayes, "Digital Simulation with Multiple Logic Values", IEEE Trans. Computer-Aided-Design, vol. 5, n° 2, pp. 274-283, April 1986.