FAULT COLLAPSING AND TEST GENERATION FOR A CIRCUIT

ABSTRACT

A novel method to generate a complete list of faults and their corresponding test vectors for a gate-level circuit is presented. This method creates the distinguishable faults of a circuit based on the paths they propagate, along with the test vector(s) for each fault. While the other available methods for fault list and test vector generation are expensive, this method tries to reduce the cost by avoiding all the unnecessary steps and merging the two tasks together.

Key words:
Fault list generation, test vector generation, fault paths

1 INTRODUCTION

Every manufactured chip or device should be tested for physical defects. The main concern in testing a digital device is to test it as thoroughly and quickly as possible. With complexity of today's digital devices, it is a very challenging task to test a circuit completely and in as little time as possible. To resolve this problem, there are a number of "test methods" developed that try to reduce the complexity of electronic testing. These methods help reduce the number of test vectors by selecting them more wisely than just trying every possible input combination.

2 TEST METHODS

2. 1 FAULT MODEL

In order to analyze faulty behavior of a digital system, a simple fault model is needed to represent faults on circuit lines. This fault model should have a close correspondence with the actual defects, it should be easy to implement as a computer program, and it should not have the complexities present in the actual circuit.

Defects in a circuit can be viewed at various levels of abstraction [1−4], including physical, transistor, gate, register transfer, and system levels. A defect at the physical level would cause different higher level fault effects, depending on various conditions that the defective part is used in. One approach to fault modeling is to model the effect of the faults on functionality of the system at various levels [5, 6]. While this approach is an accurate representation of a fault at one level, the complexity involved in analysis of effects of various defects in physical level on higher levels is very high. Moreover, this approach does not consider the faults associated with interconnections. A better approach to fault modeling is to only consider interconnection faults, called structural faults.

Structural fault modeling takes advantage of the fact that the components of a system lump their faulty behavior into interconnections [5, 6]. Therefore, this fault modeling approach assumes that the components are fault-free, and only considers the faults associated with interconnections. Since interconnections can be viewed at various levels of abstraction, for fault simulation purposes, the interconnection faults at gate-level are easier to handle than the ones at the other levels. Throughout this article, the focus is on gate-level structural faults.

2. 2 FAULT REDUCTION

Two issues that make analysis of a circuit for its faults too complicated are various fault types and presence of multiple faults in a circuit. To reduce this complexity, we should reduce the number of fault types that we need to deal with, and only consider single faults. Among various fault types, stuck-at-1 and stuck-at-0 faults [7] are general enough that almost any other fault type (e.g., stuck-open, and bridging faults [8]) can be categorized as one of these two fault types. Also, analysis based on presence of a single fault in a circuit can be generalized to multiple faults [9, 10, 11]. Single stuck-at fault model is the most used fault model for testing electronic devices today. Our focus in this article is on single stuck-at faults.

While considering only single stuck-at faults on the lines of a circuit simplifies fault model of the circuit significantly, more fault reduction can still be done. When several faults produce the same output for every input combination, they are said to be indistinguishable. Indistinguishable faults can be detected but cannot be located, and they have the same set of tests. Further fault reduction can be done by eliminating the indistinguishable faults in a circuit. This process is called “fault collapsing”. Some methods for fault collapsing are gate-oriented fault collapsing, line-oriented fault collapsing, and dominance fault collapsing. These fault collapsing methods have deficiencies when there are reconvergent fanouts in the circuit. Fault collapsing is part of our proposed method explained in section 3. Our method handles reconvergent fanouts as well, as will be explained later in this article.

2. 3 TEST GENERATION

Test generation is the process of finding input vectors (called test vectors) which create different outputs for faulty and fault-free circuits. There are several ways to generate test vectors for a circuit. In “random test generation,” random test vectors are generated and checked for the faults they can detect [12]. This is most efficient when there are many undetected faults in the circuit. Random tests can cover a high percentage of the faults in a short time. In “fault-oriented deterministic test generation,” a fault is considered in the circuit, and a test is generated for that fault. This type of test generation has a high cost and is most appropriate when there are few faults in the circuit to be detected.

In a complete test generation scheme, the common practice is to generate tests in two phases: at the beginning of the process, random test generation is employed to cover a high percentage of the faults in the circuit. In the second phase, the remaining uncovered faults are detected using deterministic test generation algorithms like the popular D-algorithm [13]. Complete test generation is a part of our novel method described in the following section.

3 OUR METHOD OF FAULT COLLAPSING AND TEST GENERATION

In this article, we propose a new approach to generating all the possible distinguishable faults and their corresponding test vectors in a circuit. Applying this method, the two expensive tasks of fault collapsing and complete test generation for a circuit are accomplished in one step.

Classically, the lines (i.e., wires connecting components at gate-level abstraction) of a circuit are labeled with the faults. However, a path in the circuit, starting from an input and ending at the output of the circuit, can represent a fault. In other words, a path through which a fault effect propagates to reach the circuit output can be labeled as a fault. Finding all these paths will provide us with all the distinguishable faults on a circuit. We can detect these fault paths by starting from the output and propagating a fault effect from the output to the inputs of the circuit and hence finding all the possible fault paths in the circuit. If we use D representing 1/0 (good/faulty) value and D̄ representing 0/1 value for faulty lines, the propagation from the output to the inputs of the individual logic gates encountered in a circuit will be as shown in Tab. 1.

Table 1 Propagation of faulty and good values from the output to the inputs of basic logical functions

	AND		**OR**		**NAND**		**NOR**		**NOT**
Out	**In1**	**In2**	**In1**	**In2**	**In1**	**In2**	**In1**	**In2**	**In**
D	D	1	D	0	D̄	1	D̄	0	D̄
D	1	D	0	D	1	D̄	0	D̄	–
D	D	D	D	D	D̄	D̄	D̄	D̄	–
D̄	D̄	1	D̄	0	D	1	D	0	D
D̄	1	D̄	0	D̄	1	D	0	D	–
D̄	D̄	D̄	D̄	D̄	D	D	D	D	–
1	1	1	1	X	0	X	0	0	0
1	–	–	X	1	X	0	–	–	–
0	0	X	0	0	1	1	1	X	1
0	X	0	–	–	–	–	X	1	–

To keep track of the fault effect propagation through various paths in the circuit, we create a tree structure as help. For a given circuit, first we give a D value to the primary output of the circuit and put it as the root of the tree. As we move backward in the circuit, when we face a gate, we create one branch for every possible input combination for that gate output, using Tab. 1 (the only exceptions are D-D or $\bar{D}$ -$\bar{D}$ input combinations which are only considered for inputs of convergence points, as will be explained later). This process is continued for every input value in every new node created in the tree. The tree is complete when all the leaves of the tree are either the primary inputs of the tree, or the "X" value ("Don't Care"). As an example, Fig. 1 shows a two-input multiplexer circuit at gate level and Fig. 2 represents the tree structure created to help us generate the fault list and the corresponding test vectors.

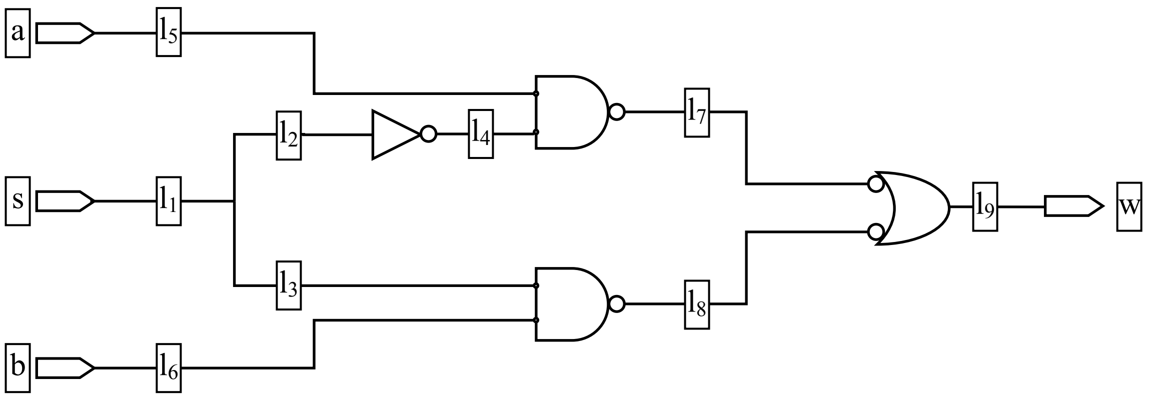

Figure 1 Structural gate-level multiplexer circuit

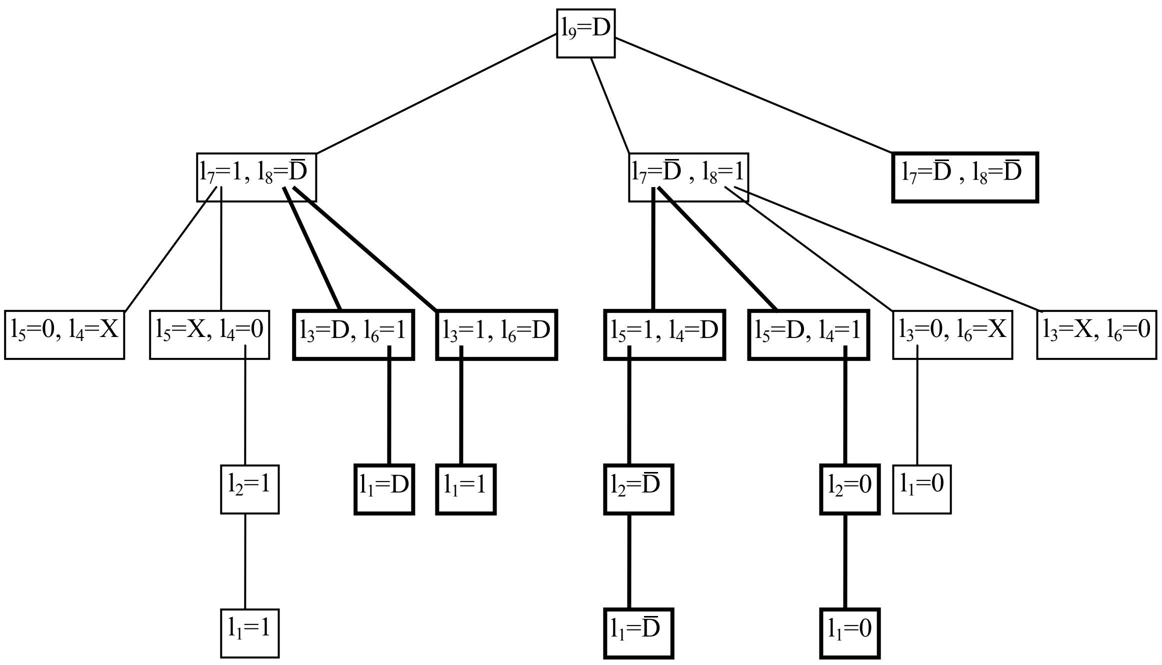

Figure 2 The fault tree illustrating the fault propagation through the circuit of Fig. 1

The fault tree will be used to detect the available fault paths in the circuit. To do so, we detect every path in the tree which has a D or $\bar{D}$ for a primary input on that path. We start with the leaf of such a path and move up to the root of the tree. As we move upward to the root, when we encounter a node (representing the inputs of a gate), if the other input on that node is branched downward, we need to backtrack from that node. If there exists at least one branch for backtracking which does not block the propagation of the fault, our fault path is valid so far, and we continue this process toward the root of the tree. For our example fault tree in Fig. 2, the valid fault paths and their test vectors are (inside the brackets are the backtracks):

1) $l_1 = D \rightarrow l_3 = D, l_6 = 1 \rightarrow l_8 = \bar{D}, l_7 = 1 \{l_5 = 0, l_4 = X\} \rightarrow l_9 = D$
   test vector: asb = 011
2) $l_1 = 1 \rightarrow l_6 = D, l_3 = 1 \rightarrow l_8 = \bar{D}, l_7 = 1 \{l_5 = 0, l_4 = X\} \rightarrow l_9 = D$
   test vector: asb = 011
   or:
   $l_1 = 1 \rightarrow l_6 = D, l_3 = 1 \rightarrow l_8 = \bar{D}, l_7 = 1 \{l_5 = X, l_4 = 0, l_2 = 1, l_1 =1\} \rightarrow l_9 = D$
   test vector: asb = X11
3) $l_1 = \bar{D} \rightarrow l_2 = \bar{D} \rightarrow l_4 = D, l_5 = 1 \rightarrow l_7 = \bar{D}, l_8 = 1 \{l_3 = X, l_6 = 0\} \rightarrow l_9 = D$
   test vector: asb = 100
4) $l_1 = 0 \rightarrow l_2 = 0 \rightarrow l_5 = D, l_4 = 1 \rightarrow l_7 = \bar{D}, l_8 = 1 \{l_3 = X, l_6 = 0\} \rightarrow l_9 = D$
   test vector: asb = 100
   or:
   $l_1 = 0 \rightarrow l_2 = 0 \rightarrow l_5 = D, l_4 = 1 \rightarrow l_7 = \bar{D}, l_8 = 1 \{l_3 = 0, l_6 = X, l_1 = 0\} \rightarrow l_9 = D$
   test vector: asb = 10X

As pointed out earlier, the special case of both inputs being D or $\bar{D}$ in Tab. 1 is applied only on the convergence point of a reconvergent fanout in the circuit, if there is any. The purpose is to find the fault effects diverging from the fanout stem and propagating through two separate paths and converging again at the convergence point. In our example fault tree, there is one reconvergent fanout and hence one such combination; $l_7 = \bar{D}$ and $l_8 = \bar{D}$. The traces for these two cases are already done in the tree as shown in bold in Fig. 2. In this specific example, when we trace the branches shown in bold, no new valid fault path is detected due to the propagation blocking during backtracking.

So far, we have formed value D on the primary output and propagated it to the primary inputs and detected several faults and their corresponding test vectors. This provides us with half of the distinguishable faults in the circuit. To find the rest of the faults, the same task should be repeated but with $\bar{D}$ on the primary output. A closer look at Tab. 1 reveals that by swapping D and $\bar{D}$ on the output of a gate, and also on the inputs of it, the logical expression remains true. This duality rule can be extended to a circuit. In the above example, by simply swapping D and $\bar{D}$, we can find the rest of the fault paths. Therefore, the four "dual" test vectors will be:

5) asb = 001
6) asb = 010 or asb = X10
7) asb = 110
8) asb = 000 or asb = 00X

Overall, the circuit of Fig. 1 has 8 distinguishable fault paths and their corresponding test vectors.

4 CONCLUSIONS

In this article, a new method for fault collapsing and test generation was introduced. This method starts from the primary output of a gate-level circuit and propagates a D value (representing 1/0, which is a good 1 value and a faulty 0 value) towards the primary inputs of the circuit. A fault tree structure is used to help us find the faulty paths of the circuit. This method handles the reconvergent fanouts in the circuit very well, with minimum level of cost. After detecting these fault paths, the rest of the fault list is generated by simply swapping D and $\bar{D}$. The test vectors corresponding to the faults are easily generated from the fault tree.

ACKNOWLEDGEMENT

The work presented in this paper has been supported under research project SPECTRUM, No. TA01011383 by Technology Agency of the Czech Republic.

REFERENCES

[1] Timoc C., Buehler M., Griswold T., Pina C., Stott F., Hess L. Logical models of physical failures. Proceedings of IEEE international test conference. 1983, pp 546−553

[2] Hayes JP. Fault modeling. IEEE Des Test Comput. 1985, pp 88−95

[3] Shen J.P., Maly W., Ferguson F.J. Inductive fault analysis of MOS integrated circuits. IEEE Des Test Comput. 1985, 2(6): 13−26

[4] Abraham J.A., Fuchs W.K. Fault and error models for VLSI. Proc IEEE. 1986, 74(5): 639−654

[5] Bushnell M.L., Agrawal V.D. Essentials of electronic testing for digital, memory, and mixed-signal VLSI circuits. Kluwer, Dordecht, 2000

[6] Jha N.K., Gupta S. Testing of digital systems. Cambridge University Press, Cambridge, 2003

[7] Galey J.M., Norby R.E., Roth J.P. Techniques for the diagnosis of switching circuit failures. Proceedings of the second annual symposium on switching circuit theory and logical design. Detroit, 1961, pp 152−160

[8] Malaiya Y.K., Rajsuman R. Bridging faults and IDDQ testing. Los Alamitos, California. IEEE Computer Society Press, Silver Spring, MD, 1992

[9] Agrawal V.K., Fung A.F.S. Multiple fault testing of large circuits by single fault test sets. IEEE Trans Comput. 1981, C-30(11): 855−865

[10] Hughes J.L.A., McCluskey E.J. Multiple stuck-at fault coverage of single stuck-at fault test sets. Proceedings of the international test conference. 1986, pp 368−374

[11] Jacob J., Biswas N.N. GTBD faults and lower bounds on multiple fault coverage of single fault test sets. Proceedings of the international test conference. 1987, pp 849−855

[12] Agrawal P., Agrawal V.D. Probabilistic analysis of random test generation method for irredundant combinational logic networks. IEEE Trans Comput. 1975, C-24(7): 691−695

[13] Roth J.P. Diagnosis of automata failures: a calculus and a method. IBM J Res Dev. 1966, 10(4): 278−291

AUTHORS

RNDr. Moslem Amiri:
Masaryk University, Faculty of Informatics, B202; amiri@mail.muni.cz
prof. Ing. Václav Přenosil, CSc.:
Masaryk University, Faculty of Informatics, B406; prenosil@fi.muni.cz