Pseudo-exhaustive Testing of Sequential Circuits for Multiple Stuck-at Faults[1]

A. Matrosova, E. Mitrofanov
Tomsk State University, Tomsk, Russia
mau11@yandex.ru, qvaz@yandex.ru

Abstract

A Sequential circuit design is based on using mixed description of a behavior of a combinational part. The behavior is represented with a composition of ROBDDs and monotonous products. The design method provides fully delay testability of a combinational part of a sequential circuit. In this paper it is shown that the method also provides multiple stuck-at faults testability of a combinational part. The pseudo-exhaustive test consisting of two parts is developed. One part is used to test sub-circuits obtained by covering nodes of the proper ROBDDs by elementary circuits (Invert-AND-XOR circuits). It allows detecting multiple stuck-at faults at gate poles of elementary circuits evenly remote from the combinational part inputs. The second test part detects all multiple stuck-at faults at gate poles of the rest component of the combinational part. It is supposed that only one of these two components of a combinational part may be faulty. An estimation of the length of such test is given.

1. Introduction

For high performance circuits it is not enough to detect single stuck-at faults (SAF) at gate poles it is desirable to detect delay faults and also multiple stuck-at faults. It is important to provide testability for these faults during circuit design. One of such approaches is suggested in [1]. The complexity of the obtained circuits is of the same order that the complexity of the circuits not fully delay testable and obtained directly from State Transition Graph (STG) description with using the shortest code words for encoding states [1]. In this paper we investigate testability properties of such circuits for multiple stuck-at faults.

A test generation strategy called as pseudo-exhaustive testing was proposed in [2, 3]. The advantages of this strategy are as follows: 1) Test inputs are calculated at test application time. 2) Test computation time depends only on the set of inputs and outputs. 3) The strategy allows partitioning of the circuit so that each partition can be tested exhaustively. In this paper we apply such strategy.

In Section 2 the method of deriving combinational part of sequential circuit is given. Testing single and multiple stuck-at faults for Invert-AND-XOR circuit is discussed in Section 3. Algorithm of obtaining pseudo-exhaustive test for a combinational part of a sequential circuit is suggested in Section 4.

2. Combinational part design

A State transition graph (STG) description of a sequential circuit behavior is used [1]. Internal states are encoded with (m,p) code words. Here m is the number of 1-value components and p - the length of a code word. After encoding states 0-values of code words are changed for don't cares. As a result we get monotonous products among state variables. A monotonous product is a factor of the corresponding non monotonous Sum of Products (SoP) depending on input variables.

Each SoP depending on input variables is represented with the Reduced Ordered Binary Decision Diagram (ROBDD). ROBDDs are derived with using the same order of variables for each SoP. As a result, we obtain the graph with s roots and two terminal nodes called Shared BDD (SBDD). Here s is the number of different SoPs. A SBDD internal node is covered with EXOR-AND-OR circuit in Fig.1. Call the circuit in Fig.1 as the elementary circuit. Note that the elementary circuit implements formula $f_v = \overline{x_i} f_v^{x_i=0} \oplus x_i f_v^{x_i=1}$.

Monotonous products are implemented with using AND gates. It is proved [1], that the derived combinational part of a sequential circuit is fully delay testable.

The circuit design is illustrated by the example. The STG description is represented with Table 1.

[1]This work is partly supported by the TSU Competitiveness Improvement Program.

978-1-5090-0693-9/16/$31.00 ©2016 IEEE

Encode FSM states with (2, 4) code words: 1 (1100), 2 (0110), 3 (0011), 4 (1001). Change 0-values of code words for don't cares and obtain Table 2.

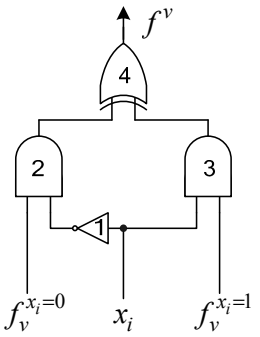

Fig. 1. Invert -AND-XOR circuit

Table 1. The STG-description of a sequential circuit

x_1	x_2	x_3	q	q	y_1	y_2	y_3	y_4	y_5
0	-	1	1	1	1	0	0	1	0
1	0	-	1	1	0	0	0	1	0
1	1	1	1	2	1	0	0	1	0
-	1	0	2	2	1	0	1	1	0
0	-	1	2	3	1	0	1	1	0
1	0	1	3	3	1	1	0	0	0
0	-	0	3	4	0	1	0	0	0
-	1	1	3	4	1	1	0	0	0
1	-	0	4	4	0	1	0	0	1
-	1	1	4	1	1	1	0	0	1

Table 2. The system of Boolean functions

x_1	x_2	x_3	z_1	z_2	z_3	z_4	z_1	z_2	z_3	z_4	y_1	y_2	y_3	y_4	y_5
0	-	1	1	1	-	-	1	1	0	0	1	0	0	1	0
1	0	-	1	1	-	-	1	1	0	0	0	0	0	1	0
1	1	1	1	1	-	-	0	1	1	0	1	0	0	1	0
-	1	0	-	1	1	-	0	1	1	0	1	0	1	1	0
0	-	1	-	1	1	-	0	0	1	1	1	0	1	1	0
1	0	1	-	-	1	1	0	0	1	1	1	1	0	0	0
0	-	0	-	-	1	1	1	0	0	1	0	1	0	0	0
-	1	1	-	-	1	1	1	0	0	1	1	1	0	0	0
1	-	0	1	-	-	1	1	0	0	1	0	1	0	0	1
-	1	1	1	-	-	1	1	1	0	0	1	1	0	0	1

Let we extract the system F of completely specified Boolean functions directly from Table 2. For each function f of the system we select the cubes that marked with the 1-value in Table 2 column that corresponds to function f. Each sub-cube depending on state variables we implement with sub-circuit that consists of two input AND gates. In the example, the sub-circuit contains the only AND gate.

Consider SoP of function f. Divide the products corresponding to function f into subsets. Include products with the same literals among state variables in the same subset. Note that sub-products (depending on state variables) of all subsets are monotonous. In the example $f=y_1$, monotonous products are represented as factors:

$$y_1 = z_1z_2(\bar{x}_1x_3 \vee x_1x_2x_3) \vee z_2z_3(x_2\bar{x}_3 \vee \bar{x}_1x_3) \vee \\ \vee z_3z_4(x_1\bar{x}_2x_3 \vee x_2x_3) \vee z_1z_4(x_2x_3) \quad (1)$$

Represent each SoP as ROBDD. Integrate all these ROBDDs into Shared BDD (Fig. 2).

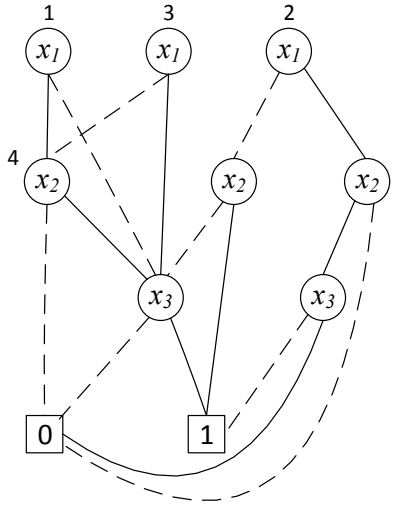

Fig. 2. Shared BDD (SBDD)

Cover each node of Shared BDD with circuit in Fig. 1 not paying attention on the edges coming to 0 terminal node. As a result, we get circuit in Fig. 3.

Connect the outputs of the circuit implementing Shared BDD with the outputs of the corresponding circuits implementing sub-products depending on state variables (monotonous sub-products of function f). For that we use two input AND gates. Then we connect all obtained sub-circuits applying OR gates and get circuit implementing function f of system F (Fig. 4).

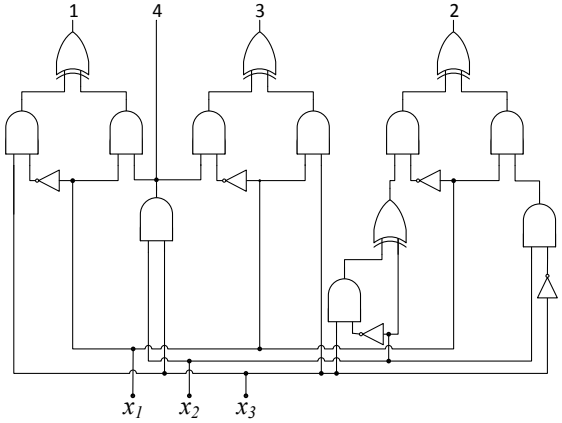

Fig. 3. SBDD implementation

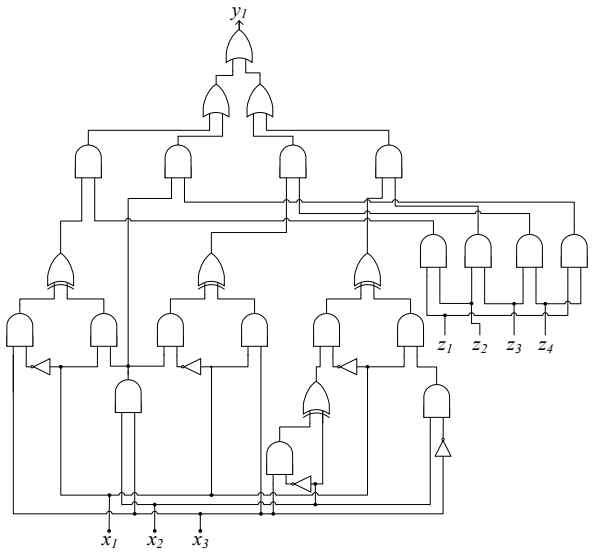

Fig. 4. Implementation of function y_1

3. Testing of single and multiple stuck-at faults for Invert-AND-XOR circuit

Consider single stuck-at 1, 0 faults at elementary circuit inputs $f_v^{x_i=0}$, $f_v^{x_i=1}$ and output f_v. (Fig.1). Each of them correlates with single stuck-at fault at the ROBDD low *v*, high *v* and *v*, correspondingly. Stuck–at 1(0) fault of variable x_i correlates with 01(10) fault of edges running from node *v*. This means that any test pattern for stuck-at fault at the external pole of the elementary circuit covering node *v* is the test pattern for the corresponding fault of ROBDD node *v*, low *v* and high *v*.

Table 3

$f_v^{x_i=0}$	$f_v^{x_i=1}$	x_i	$f_v^{x_i=0}$	$f_v^{x_i=1}$	x_i	f_v
1	0	1		1	0	1
0	1	0	1		1	1
1	1	0	0			0
1	1	1		0		0

Let Table 3represents test for single stuck-at faults at the external poles of the elementary circuit (Fig.1). Its first three columns correspond to input variables of the circuit. The next three columns correspond to 0, 1 stuck-at faults at the circuit inputs detected by the Boolean vector represented by the first three columns. The last column represents stuck-at faults at the circuit output.

Take into consideration that sometimes only one of two vectors

$f_v^{x_i=0}$	$f_v^{x_i=1}$
0	1
1	0

may be delivered to inputs of the elementary circuit. Table 4 (5) represents test patterns for single stuck-at faults at external poles of the circuit when only vector 10 (01) may be delivered.

Table 4

$f_v^{x_i=0}$	$f_v^{x_i=1}$	x_i
0	0	0
1	0	0
1	0	1
1	1	0
1	1	1

Table 5

$f_v^{x_i=0}$	$f_v^{x_i=1}$	x_i
0	0	1
0	1	0
1	1	0
0	1	1
1	1	1

It is proved that test represented by one of Tables 3-5 for single stuck at faults at external poles of the elementary circuit detects all multiple stuck-at faults at gate poles of the circuit.

4. Deriving pseudo-exhaustive test

Testing of sub-circuits implementing SoPs depending on input variables

We consider that either sub-circuits implementing SoPs or the rest component of the combinational part of sequential circuit may be faulty, but not both together.

Let only one elementary circuit be faulty. This circuit covers node *v* of Shared BDD. We need provide the fault manifestation of the elementary circuit to any output of the SBDD by any path connecting *v* with one of roots of ROBDD comprising SBDD. Denote this path as ε. Let k_ε be the product originated by ε. Let k_δ be the product that provides delivering the test pattern on inputs of the faulty elementary circuit. Having got $k_\varepsilon k_\delta$ for each input Boolean vector from the proper Table (among Tables 3-5) of faulty elementary circuit we find test patterns depending on input variables $x_1,\ldots, x_n$. Call these test patterns as test fragment of the elementary circuit. Note that algorithm of deriving $k_\varepsilon k_\delta$ has a polynomial complexity.

Divide elementary circuits of sub-circuits, originated by input variables SoPs into levels in conventional way. The first level consists of elementary circuits which inputs are combinational part inputs. An elementary circuit belongs to the *i*-th level if its inputs are connected with outputs of elementary circuits of the (*i*-1)-th and less levels.

Sub-circuits (each implements input variable SoP) connected with the same factor (Table 2) comprise the separate group.

We test each group separately using the proper test fragments beginning from elementary circuits of the

first level. Then we test elementary circuits of the second level and so on. During testing the group, the values of state variables turn the factor corresponding to the group into 1.

Going over the groups, we either find a fault or conclude that there are no multiple stuck-at faults at gate poles of elementary circuits of the same level among all groups. The test length is not more $5\gamma N$, where N is the number of internal nodes of Shared BDD and γ is the middle number of groups, which contain the elementary circuit.

Testing of sub-circuit implementing monotonous system

Correlate the output of each SoP depending on input variables to variable w_i. Form monotonous system F^* (from Table like 2) depending on state variables and variables w_i. For example, expression (1) is rewritten as follows:

$$w_1z_1z_2 \vee w_2z_2z_3 \vee w_3z_3z_4 \vee w_4z_1z_4 .$$

Consider single faults of literals of monotonous SoP f_j^* from F^*. Changing the certain literal in the certain product for constant 1(0) call *b(a)*-fault. If several literals are fault, call it multiple fault of the SoP. It is known [4] that all multiple faults of irredundant SoP consisting of prime implicants are detected with a test for single faults of literals of this SoP. This test is union of test patterns for each *a,b*-faults. Any f_j^* from F^* is irredundant SoP as it consists of monotonous products.

Test pattern α of the monotonous SoP for *a*-fault of product K is obtained by changing all don't care components of a ternary vector corresponding to product K for 0-values. Test pattern β for *b*-fault of product K and its literal x_i is obtained by changing the component corresponding to fault literal x_i for inverse value in vector α (in test pattern for *a*-fault).

Note that any multiple stuck-at fault at gate poles of sub-circuit implementing monotonous SoP f_j^* (sub-circuit is designed by any method keeping SoP [5]) is equivalent to the multiple fault of these SoP literals. Consequently, test for single *b(a)* faults of monotonous SoP f_j^* detects any multiple stuck-at fault at gate poles of the sub-circuit implementing this SoP. In [6] the method of deriving test T for multiple stuck at faults at gate poles of circuit implementing monotonous system F^* is suggested. The system is designed by one of the methods keeping system F^*.

We need form SoP f^* [6] including all products of system F^*. In [6] it is shown that test T for single $a(b)$-faults of SoP f^* is the test for the same faults of system F^* and, consequently, the test for multiple stuck-at faults at gate poles of circuit implementing monotonous system F^*.

The length of test T is not more than a sum of ranks of SoP f^* products and the number of these products. Note that delivering the test pattern for current product K from f^* we need provide 1-value output of a SoP from the group correlating with K.

To get pseudo-exhaustive test for circuit like Fig. 4 we need derive pseudo-exhaustive test for sub-circuits implementing all SoPs depending on input variables and test T for multiple stuck-at faults at gate poles for the rest component of the circuit implementing monotonous system F^*. First we have to test all SoPs depending on input variables.

5. Conclusion

Pseudo-exhaustive test for multiple stuck-at faults at gate poles of the combinational part of a sequential circuit designed with using a description of the behavior as a composition of ROBDDs and monotonous products is developed. It means that circuits designed in this way are full delay testable and pseudo-exhaustive multiple stuck-at fault testable. It is shown that circuits designed in such a way being full delay testable are also pseudo-exhaustive multiple stuck-at fault testable.

References

[1] A.Yu. Matrosova, E.V. Mitrofanov, V. Singh, "Delay Testable Sequential Circuit Designs", *Proceedings of IEEE East-West Design & Test Symposium (EWDTS 2013)*, Ukraine, 2013, pp. 293-296.

[2] E. J. McCluskey, "Verification Testing-A Pseudo-exhaustive Test Technique", *IEEE Transactions on Computers*, Vol. C-33, No.6, 1984, pp. 541-546.

[3] E. Macii, T. Wolf, "Multiple Stuck-at Faults Test Generation Techniques for Combinational Circuits Based on Network Decomposition", *36th Midwest Symp. On CAS,* Vol. 1, 1993, pp. 465-467.

[4] I. Kohavi, Z. Kohavi, "Detection of Multiple Faults in Combinational Logic Networks", *IEEE Transasctions on Computers VC-20*, no. 6, 1975, pp. 556-568.

[5] A. Matrosova, D. Kudin, E. Nikolaeva, "Circuits without False Paths", *Proceedings of IEEE East-West Design & Test Symposium (EWDTS 2014)*, Ukraine, 2014. pp. 160-163.

[6] A. Matrosova, V. Andreeva, V. Tomkov, "Fully delay and multiple stuck-at fault testable FSM design", *Proceedings of IEEE East-West Design & Test Symposium (EWDTS 2015),* Georgia, Batumi, 2015, pp. 212-215.