Acta Cybernetica **16** (2004) 545–566.

An addition to the methods of test determination for fault detection in combinational circuits

Ljubomir Cvetković*

Abstract

We propose a procedure for determining fault detection tests for single and multiple fault in combinational circuits. The stuck-at-fault model is used. By the proposed procedure all test vectors for single and multiple stuck-at-fault in combinational circuit are determined. The path sensitization method is used in the test signal propagation while test signals are defined on a four element set. The procedure can also be applied to the fault detection in programmable logic devices. We consider two-level combinational circuits which are realized by the PAL architecture and we propose a procedure for determining a test set which detects all single stuck-at-faults. As a mathematical tool, the cube theory is used.

Keywords: combinational circuit, programmable logic devices, single fault, multiple fault, fault detection, test cube

1 Introduction

In literature many approaches for fault detection in combinational circuits can be found (see, for example, [1, 13, 14]).

Test generation problem can be considered viewed as a finite space search problem of finding appropriate logic value assignments to the primary inputs such that the given fault is detected. Since the size of the search space is exponential in the number of inputs, the test generation problem is proven to be NP-complete [9].

In the stuck-at-fault model it is assumed, that a faulty line of the combinational circuit is permanently set to either the value of 0 or 1. The corresponding faults are called stuck-at-0 and stuck-at-1 respectively. Combinational circuit lines are understood to be all connections along which logic signals move. The fault 0 (or 1) at the line a is denoted by $a/0$ (or $a/1$).

The procedure for the single stuck-at-fault detection in a combinational circuit is based on the following two conditions. First, one should generate on the faulty line a test signal whose value is opposite to the fault signal. Secondly, one should enable the test signal propagation to the output line of the combinational circuit,

*Ljubomir Cvetković, Trg Žitna Pijaca 3, 22000 Sremska Mitrovica, Serbia and Montenegro, JP PTT "SERBIA"-HOLDING, Belgrade, Serbia and Montenegro. E-mail: ljubomir@ptt.yu

which means that the path from the faulty line to the output line must be active. Such an approach to fault detection is known as the path sensitization method. The path sensitization method is presented for the first time by Armstrong [5], where it is applied to the single fault detection. Afterwards some new procedures for the multiple fault detection are developed.

The D-algorithm is one of the first algorithms for automatic generation of test vectors for single stuck-at-faults in combinational circuits. This algorithm has been developed by Roth [17]. Test signals take five algebraic values: $0, 1, X, \bar{D}$ and D. Logical zero and one are denoted by 0 and 1, respectively, while X belongs to the set $\{0, 1, \bar{D}, D\}$, where D is equal to 1 and 0 in fault free and faulty circuit, respectively.

One has quickly realized that the D-algorithm should be improved. The reduction of the number of paths is achieved by means of the so called 9-V algorithm, whose structure is closest to the D-algorithm [8].

In construction of the algorithms for automatic test generation for combinational circuits the notions of D-boundary and P-boundary are defined [1]. The D-boundary is the set of logic gates whose output lines are in state X, while at least one test signal with values D and $\bar{D}$ appears on the input lines of these gates. It is necessary to choose one of the logic gates from the D-boundary, together with determination of the signals on its input lines, in such a way that output line gets the value D or $\bar{D}$, in order to direct the test signal propagation towards the primary output. The P-boundary is the set of logic gates, in which the state of the output line is fully determined, while the state of values on the input lines is undetermined.

A more efficient decision making process was proposed almost 20 years ago in the PODEM algorithm [10]. PODEM only allows assignment of values at primary inputs (PIs) and these values are then propagated towards internal lines using implications. After the initial assignment of values on some primary inputs, and implication of these values, the process continues by assigning additional PI values and checking if the fault effect has been propagated to the primary outputs. For the combinational circuits, in which the number of paths is large, we have to enumerate and label all the paths. The algorithms are based on the search on graphs and are the subject of research [16].

Generally speaking, a lot of procedures for testing combinational circuits have been developed. Most of them are designed for restricted type circuits. For fan-out free combinational circuits the test set which detects all single stuck-at-faults, also detects all multiple stuck-at-faults [12].

It is well known that in a two-level irredundant combinational circuit every set of test vectors which detects all single stuck-faults detects at the same time all multiple stuck-at-faults [7], p. 65.

The AND/OR structure of a two-level circuit corresponds naturally to sum of products structure of disjunctive-normal-form. All single and multiple stuck-at-faults are detectable in such circuits if all single faults at input and output lines of the AND gates are detectable [11].

In the literature more subtle and specific types of faults have been derived for PLAs.

Since PLAs can be used to implement any two-level logic expressions, we can treat the PLA as a two-level logic gate implementation and model the typical stuck-at-faults at input, input inverters, product lines and output lines.

A fault model, according to which incorrect logical connections in AND and OR arrays are tested, is proposed in [18]. Since AND gate inputs and OR gate inputs can be either improperly connected or disconnected, modeled faults can be conveniently divided into four classes. First, if an input literal is disconnected from AND gate, this causes the implicant to "grow" since the implicant becomes independent of some input variable. This growth of an implicant can be seen quite easily on a Karnaugh map. These faults are called growth faults. The set of all single growth faults are denoted as G. If an AND gate becomes disconnected from an OR gate, this causes an implicant to disappear from the map of the function. Hence, this set of single faults is the set of disappearance faults D. If an input literal becomes incorrectly connected to an AND gate, then the implicant "shrinks". These faults form the set of shrinkage faults S. Finally, if an AND gate becomes incorrectly connected to an OR gate, than an implicant appears in the map of the affected functions. Hence, these are called appearance faults A.

There could be also bridging faults in the PLAs. Especially the bridging faults could easily happen between the adjacent lines due to the regular layout style of PLAs. The effects of the bridging could be AND or OR depend on the technology on which PLA realizes.

In PLA and PAL, the test sets, which detects cross points faults, detect at the same time a great number of stuck-at-faults and short faults. Cross point faults are important in this devices and therefore several procedures for detecting these faults have been developed (see, for example [3, 4]).

The paper [19] presents a method for obtaining a minimal set of test configurations and their associated set of test patterns that completely tests re-programmable Programmable Logic Arrays (PLAs) including EEPROM, UV-EPROM, and SRAM based re-programmable PLAs typically found in Complex Programmable Logic Devices (CPLDs). The resultant set of test configurations and vectors detect all single and multiple stuck-at-faults (including line and transistor faults) as well as all bridging faults in the PLA.

Bridge defects also cause less obvious effects that can introduce further modelling complications. If a bridge defect creates a feedback loop, a formerly stable combinational circuit may take on oscillatory or sequential properties. Also, when intermediate voltage levels occur, downstream logic gates with varying input voltage thresholds can interpret the same voltage level differently. This is known as the *Byzantine General's Problem* and can significantly complicate bridge defect modelling issues [2].

The problem of determining a minimal number of control inputs for converting a programmable logic array (PLA) with undetectable faults to crosspoint-irredundant PLA for testing has been formulated as a nonstandard set covering problem. By representing subsets of sets as cubes, this problem has been reformulated as familiar problems. It is noted that this result has significance because a crosspoint-irredundant PLA can be converted to a completely testable PLA in a straightfor-

ward fashion, thus achieving very good fault coverage and easy testability [15].

A system of Boolean functions can be realized by blocks arranged in several levels, where each block represents a PAL with a small number of inputs and outputs. Moreover, multi-level circuits are harder tested than two-level circuits [6].

In this paper we propose the stuck-at-fault model for PLA and assume that the programmable elements, where the faults appear, get the signal values stuck-at-0 or stuck-at-1. The acceptance of the proposed model is motivated by the fact that the faults in real PLA appear in programmable elements. Therefore, we accept the following approximation of defects: PLA programmable points, where faults appear, get the signal values permanent logic 1 or permanent logic 0. This approximation is the main strength of the proposed model. It is well known that the quality of defect abstraction has an influence on the relevance of the fault model. On the other hand, the defect abstraction by faults has an influence on the testing methodology. In our case it is appropriate to use the path sensitization method as a testing method and to introduce a four element signal value set. We shall show in the sequel that the fault types, appearing in the above mentioned standard fault models for PLA, can be detected by the proposed procedure as well, thus proving the relevance and sufficiency of our model. Details on the fault models comparison are given in section 5.

2 Test signal propagation

The values of test signals are denoted by D and C. Test signal D has value 1 in a fault-free combinational circuit, and it has value 0 in a faulty combinational circuit. Test signal C has value 0 in a fault-free combinational circuit, and it has value 1 in a faulty combinational circuit. Test signals have values from the set $\{0, 1, C, D\}$ where $C, D \in \{0, 1\}$. It should be noted that values of C and D are not determined at the beginning of the fault detection procedure; they are concretized during the execution of the procedure. The set $\{0, 1, C, D\}$ and operations $+, \bullet$ and $^-$ represent a Boolean algebra, if the operations are defined by the following tables

Table 1: Operation $+$

$+$	0	1	C	D
0	0	1	C	D
1	1	1	1	1
C	C	1	C	1
D	D	1	1	D

Table 2: Operation $\bullet$

$\bullet$	0	1	C	D
0	0	0	0	0
1	0	1	C	D
C	0	C	C	0
D	0	D	0	D

The intersection operation ϕ of the set $\{0, 1, X\}$ is defined by Table 4. Symbol $\emptyset$ in Table 4 means that operation ϕ is not defined.

Of course, using a four element set in constructing test vectors is well- known [7], p.44. The inputs can take the four values $\{0, 1, \bar{D}, D\}$ where the symbols D

Table 3: Operation $^-$

$^-$	0	1	C	D
	1	0	D	C

Table 4: Operation ϕ

ϕ	0	1	X
0	0	$\emptyset$	0
1	$\emptyset$	1	1
X	0	1	X

and $\bar{D}$ are used to represent respectively a signal which has the value 1 and 0 in a well functioning circuit and 0 and 1 in a faulty circuit.

Consider now a logic gate G with n input lines and output line v (Fig.1). Gate G can be OR, NOR, AND or NAND. Suppose that test signals of value D or C appear on input lines $u_1, u_2, \ldots, u_p$ of gate G while propagation signal values 0 or 1 appear on the remaining lines $z_1, z_2, \ldots, z_q$ $(p + q = n)$.

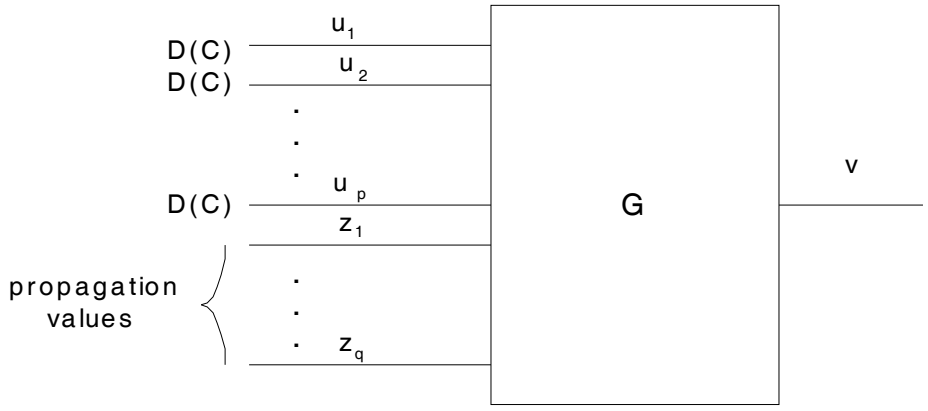

Figure 1: Test signals have the same value

The test signal propagation to the output line v can occur in the case when test signals of the same value (D or C) appear on input lines $u_1, u_2, \ldots, u_p$ while signal values on the remaining lines $z_1, z_2, \ldots, z_q$ must have value 0 for gates OR and NOR, and value 1 for gates AND and NAND.

Let us analyze conditions under which test signals are propagated from input lines to the output line of gates OR, NOR, AND and NAND. Logical operation "$\wedge$" is used in logical relations which describe the test signal propagation.

The following logical relations hold:
for OR gate

$$(u_1 = D) \wedge \cdots \wedge (u_p = D) \wedge (z_1 = 0) \wedge \cdots \wedge (z_q = 0) \rightarrow (v = D), \qquad (1)$$

$$(u_1 = C) \wedge \cdots \wedge (u_p = C) \wedge (z_1 = 0) \wedge \cdots \wedge (z_q = 0) \rightarrow (v = C), \qquad (2)$$

for NOR gate

$$(u_1 = D) \wedge \cdots \wedge (u_p = D) \wedge (z_1 = 0) \wedge \cdots \wedge (z_q = 0) \rightarrow (v = C), \tag{3}$$

$$(u_1 = C) \wedge \cdots \wedge (u_p = C) \wedge (z_1 = 0) \wedge \cdots \wedge (z_q = 0) \rightarrow (v = D), \tag{4}$$

for AND gate

$$(u_1 = D) \wedge \cdots \wedge (u_p = D) \wedge (z_1 = 1) \wedge \cdots \wedge (z_q = 1) \rightarrow (v = D), \tag{5}$$

$$(u_1 = C) \wedge \cdots \wedge (u_p = C) \wedge (z_1 = 1) \wedge \cdots \wedge (z_q = 1) \rightarrow (v = C), \tag{6}$$

for NAND gate

$$(u_1 = D) \wedge \cdots \wedge (u_p = D) \wedge (z_1 = 1) \wedge \cdots \wedge (z_q = 1) \rightarrow (v = C), \tag{7}$$

$$(u_1 = C) \wedge \cdots \wedge (u_p = C) \wedge (z_1 = 1) \wedge \cdots \wedge (z_q = 1) \rightarrow (v = D). \tag{8}$$

In relations (1)-(8) test signal values D and C appear p times each. Test signal on the output line v can be obtained also by setting some test signals on input lines $u_1, u_2, \ldots, u_p$ to propagation signal values. This can be done with each k $(1 \leq k \leq p-1)$ element subset of the set of input lines $\{u_1, u_2, \ldots, u_p\}$. Hence the total number of logical relations is

$$\binom{p}{1} + \binom{p}{2} + \cdots + \binom{p}{p-1} = 2^p - 2 \tag{9}$$

Consider the situation presented in Fig. 2. Test signal value D appears on r input lines of the gate G while value C appears on s input lines. Lines $z_1, z_2, \ldots, z_q$ have propagation signal values for gate G. In this case no one of the relations (1)-(8) can be applied since $D \cdot C = 0$.

Suppose that gate G on Fig.2 is an OR gate.

Test signal of value D is propagated to the output line v according to the relation

$$\begin{aligned}(u_1 = D) \wedge \cdots \wedge (u_r = D) \ \wedge \ & (u_{r+1} = 0) \wedge \cdots \wedge (u_{r+s} = 0) \\ \wedge \ & (z_1 = 0) \wedge \cdots \wedge (z_q = 0) \rightarrow (v = D).\end{aligned} \tag{10}$$

Test signal value C appearing s times in relation (10) is replaced by 0. Hence, test signal values D appear r times in relation (10), while the remaining $s+q$ $(r+s+q=n)$ input lines of the gate OR must have values of propagation signals. For the propagation of the test signal of value D to the output line one can write $2^r - 2$ relations.

Test signal of value C is propagated to the output line v according to the relation

$$\begin{aligned}(u_1 = 0) \wedge \cdots \wedge (u_r = 0) \ \wedge \ & (u_{r+1} = C) \wedge \cdots \wedge (u_{r+s} = C) \\ \wedge \ & (z_1 = 0) \wedge \cdots \wedge (z_q = 0) \rightarrow (v = C).\end{aligned} \tag{11}$$

Analogously to the above analysis, for the propagation of the test signal of value C to the output line v one can write $2^s - 2$ relations.

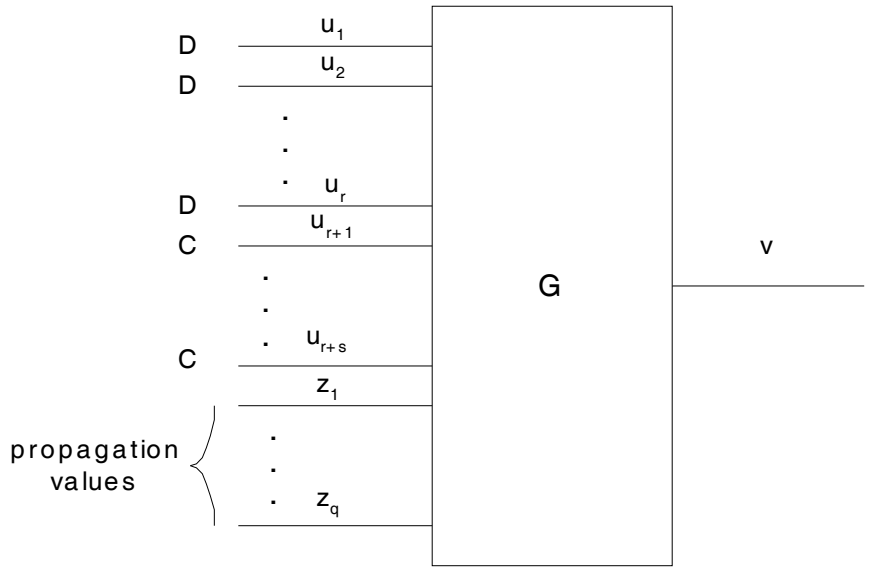

Figure 2: Test signals with distinct values

The following logical relations hold:
for NOR gate

$$(u_1 = D) \wedge \cdots \wedge (u_r = D) \ \wedge \ (u_{r+1} = 0) \wedge \cdots \wedge (u_{r+s} = 0) \\ \wedge \ (z_1 = 0) \wedge \cdots \wedge (z_q = 0) \rightarrow (v = C), \qquad (12)$$

$$(u_1 = 0) \wedge \cdots \wedge (u_r = 0) \ \wedge \ (u_{r+1} = C) \wedge \cdots \wedge (u_{r+s} = C) \\ \wedge \ (z_1 = 0) \wedge \cdots \wedge (z_q = 0) \rightarrow (v = D), \qquad (13)$$

for AND gate

$$(u_1 = D) \wedge \cdots \wedge (u_r = D) \ \wedge \ (u_{r+1} = 1) \wedge \cdots \wedge (u_{r+s} = 1) \\ \wedge \ (z_1 = 1) \wedge \cdots \wedge (z_q = 1) \rightarrow (v = D), \qquad (14)$$

$$(u_1 = 1) \wedge \cdots \wedge (u_r = 1) \ \wedge \ (u_{r+1} = C) \wedge \cdots \wedge (u_{r+s} = C) \\ \wedge \ (z_1 = 1) \wedge \cdots \wedge (z_q = 1) \rightarrow (v = C), \qquad (15)$$

for NAND gate

$$(u_1 = D) \wedge \cdots \wedge (u_r = D) \ \wedge \ (u_{r+1} = 1) \wedge \cdots \wedge (u_{r+s} = 1) \\ \wedge \ (z_1 = 1) \wedge \cdots \wedge (z_q = 1) \rightarrow (v = C), \qquad (16)$$

$$(u_1 = 1) \wedge \cdots \wedge (u_r = 1) \ \wedge \ (u_{r+1} = C) \wedge \cdots \wedge (u_{r+s} = C) \\ \wedge \ (z_1 = 1) \wedge \cdots \wedge (z_q = 1) \rightarrow (v = D). \qquad (17)$$

3 Fault detection

In the sequel we propose a procedure for the single fault detection for fan-out free combinational circuits.

Let a unique path $(h, e_1, e_2, \ldots, e_n, y)$ in combinational circuit go from the faulty line h to the output line y. The first logical relation defines the conditions of the test signal propagation from the line h of the distance d (from the output line) to the line e_1 that has a distance $d-1$. Line e_1 is the logic gate output line whose input line is h. The second logical relation defines the conditions for test signal propagation from the line e_1 of the distance $d-1$, to the line e_2 that has a distance $d-2$. The process of logical relations writing continues as far as the last logical relation is found. It defines conditions for the test signal propagation from the line e_n of the distance 1, to the output line y in combinational circuit. Uniting the d logical relations into a single logical relation is obvious. The test signal value from the right hand side of the first logical relation is replaced into the second relation. Then the test signal value from the right hand side of the second logical relation is replaced into the third relation and the process continues.

Obviously, when writing logical relations we go only forward, unsuccessful steps being excluded in this case. For a single stuck-at-fault, we surely determine all test vectors. At the and of the procedure one obtains a logical relation in which signals on input lines appear. Based on the test signal values on the input lines one can determine sets of test cubes and on the basis of them we determines the detecting test set.

Example 1. Determine all test vectors detecting a single fault $h/1$ in the combinational circuit in Fig. 3.

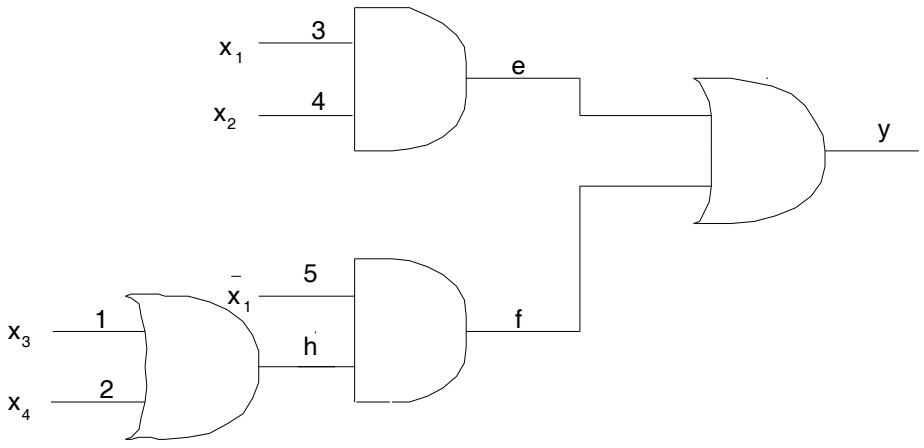

Figure 3: A single fault in a combinational circuit

Table 5: Auxiliary data

$h = 0$	$XX00$
5=1	$0XXX$
$e = 0$	$0XXX$
	$X0XX$

The unique path from the faulty line h to the output of the circuit is (h, f, y). The signal having the value C should be generated on the faulty line h. Logical relations defining conditions for test signal propagation from faulty line to the output of the combinational circuit are

$$(h = C) \wedge (5 = 1) \rightarrow (f = C), \tag{18}$$

$$(f = C) \wedge (e = 0) \rightarrow (y = C). \tag{19}$$

On the basis of relations (18) and (19) the next relation is derived

$$(h = C) \wedge (5 = 1) \wedge (e = 0) \rightarrow (y = C). \tag{20}$$

Line covers for h, 5 and e are shown in Table 5.
Test cubes detecting the fault $h/1$ are obtained on the basis of Table 5

$$\{XX00\}\phi\{0XXX\}\phi\{0XXX, X0XX\} = \{0X00, 0000\}.$$

Finally, we get the set of all test vectors for the given single fault: $\{0000, 0100\}$.

The described procedure for detecting single stuck-at-faults can be applied also for detection of multiple stuck-at-fault. Under a multiple fault we understand a fault consisting of single faults. In order to detect a multiple fault, one should generate test signals with values complementary to the fault signals at the faulty lines of the combinational circuit. To describe the test signal propagation we use logical relations described in Section 2.

In the next example a double fault is located on two paths in the combinational circuit, the two paths having some lines in common.

Example 2. Let us detect the double fault $\{1/0, 3/0\}$ in combinational circuit on Fig. 4.

To lines 1 and 3 we associate the test signals, having values opposite to the fault signals. We have relations

$$(1 = D) \wedge (2 = 1) \rightarrow (e = D), \tag{21}$$

$$(3 = D) \wedge (4 = 0) \rightarrow (f = D). \tag{22}$$

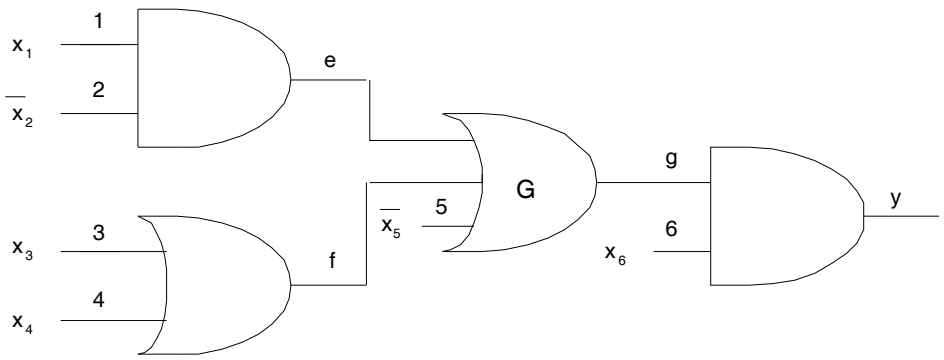

Figure 4: A double fault in a combinational circuit

The propagation of the test signal from input lines e and f of gate G to its output line is described by the relation

$$(e = D) \wedge (f = D) \wedge (5 = 0) \rightarrow (g = D). \tag{23}$$

We write a logical relation which describes the propagation of the test signal from line g to the output line y of the combinational circuit

$$(g = D) \wedge (6 = 1) \rightarrow (y = D). \tag{24}$$

On the basis of relations (23) and (24) we have

$$(e = D) \wedge (f = D) \wedge (5 = 0) \wedge (6 = 1) \rightarrow (y = D). \tag{25}$$

Putting relations (21) and (22) into relation (25) we get

$$(1 = D) \wedge (2 = 1) \wedge (3 = D) \wedge (4 = 0) \wedge (5 = 0) \wedge (6 = 1) \rightarrow (y = D). \tag{26}$$

According to (26) we have

$$\{1XXXXX\}\phi\{X0XXXX\} \ \phi \ \{XX1XXX\}\phi\{XXX0XX\}\phi \\ \phi \ \{XXXX1X\}\phi\{XXXXX1\} = \{101011\}.$$

Test vector 101011 detects the double fault $\{1/0, 3/0\}$.

In redundant combinational circuits we can come across the effect of masking. Masking is a phenomenon which can appear in detecting a multiple fault. If α and β are single faults in a combinational circuit of a general structure, the following situation can occur. The single fault α is detectable (and is detected by a certain test set). However, the fault cannot be detected in the presence of the fault β , i.e. in the case of double fault $\{\alpha, \beta\}$. We say that the fault β masks the fault α.

4 Fault detection in programmable logic devices

The above described procedures for fault detection, where test signals values belong to the set $\{0, 1, C, D\}$, can also be applied to the fault detection in programmable logic devices.

A PLA with n input lines, m internal lines and p output lines is represented in Fig. 5.

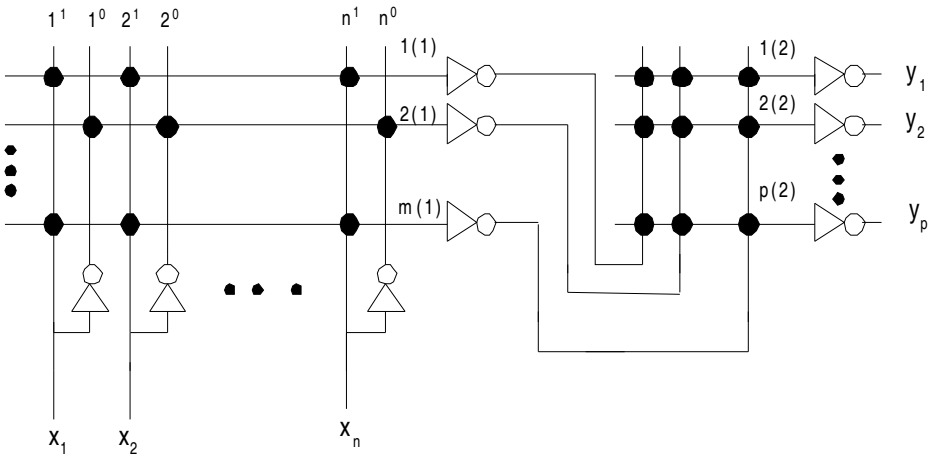

Figure 5: Programmable logic array

Programmable elements are denoted by symbol •. We apply the following way of marking programmable points of PLA:

(i, j^1) - crosspoint of internal line i and a bit line j^1 in AND array $(i = 1, 2, \ldots, m,\ j = 1, 2, \ldots, n)$

(i, j^0) - crosspoint of internal line i and a bit line j^0 in AND array $(i = 1, 2, \ldots, m,\ j = 1, 2, \ldots, n)$

(i, j) - crosspoint between the lines i and j in OR array $(i = 1, 2, \ldots, p,\ j = 1, 2, \ldots, m)$

Within the stuck-at-fault model, PLA programmable points where faults appear, get the signal values stuck-at-0 or stuck-at-1.

The fault of the type of disconnection of a programmable element at point (i, j^1) is denoted by $(i, j^1)/1$ (stuck-at-1 fault), and the fault of the type of a short connection at point (i, j^1) is denoted by $(i, j^1)/0$ (stuck-at-0 fault). Fault marking at the point (i, j^0) is carried out in a similar way.

The fault in the form of disconnection of the programmable element within OR array at the point (i, j) is marked by $(i, j)/1$, and the fault in the form of a short connection at the point (i, j) is marked by $(i, j)/0$.

Example 3. In the programmable logic array presented in Fig. 6, detect multiple fault $\{(1,1)/0,(1,2)/0,(1,3)/0,(1,4)/0)\}$.

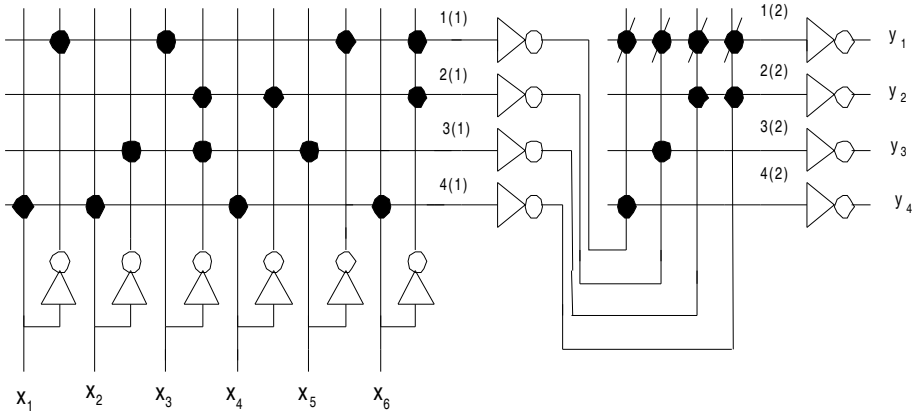

Figure 6: A multiple fault in programmable logic array

It is necessary to generate test signals of value D at points (1,1), (1,2), (1,3) and (1,4). The following logical relation describes the propagation of the test signals to the output line 1(2)

$$(1,1) = D \wedge (1,2) = D \wedge (1,3) = D \wedge (1,4) = D \rightarrow 1(2) = D. \tag{27}$$

Obviously, test signals of value D exist on the input lines of the OR array on the basis of relation (27). Tracing back, these test signals are attributed to the output lines of the AND array, i.e. $1(1) = C$, $2(1) = C$, $3(1) = C$ and $4(1) = C$. Signals of value C or 1 are attributed to the programmable points of the AND array, where at no pair of bit lines identical signals appear. Such an arrangement of test signals is presented in Fig. 7.

On the basis of Fig. 7 we have

$$Q_1 = \{1X1X10\}, Q_2 = \{XX11X0\}, Q_3 = \{X01X1X\}, Q_4 = \{10X1X0\},$$

which implies

$$Q_1 \ \phi \ Q_2 \ \phi \ Q_3 \ \phi \ Q_4 = \{101110\}.$$

Test vector 101110 detects the given multiple fault.

In order to obtain the remaining test vectors it is necessary to construct other test signals arrangements in the programmable points of the AND array.

Of course, for a PLA with a great number of input lines it is necessary to perform a special analysis (using combinatorics) for constructing optimal algorithms assigning test signals to the programmable points of an AND array. Obviously, one

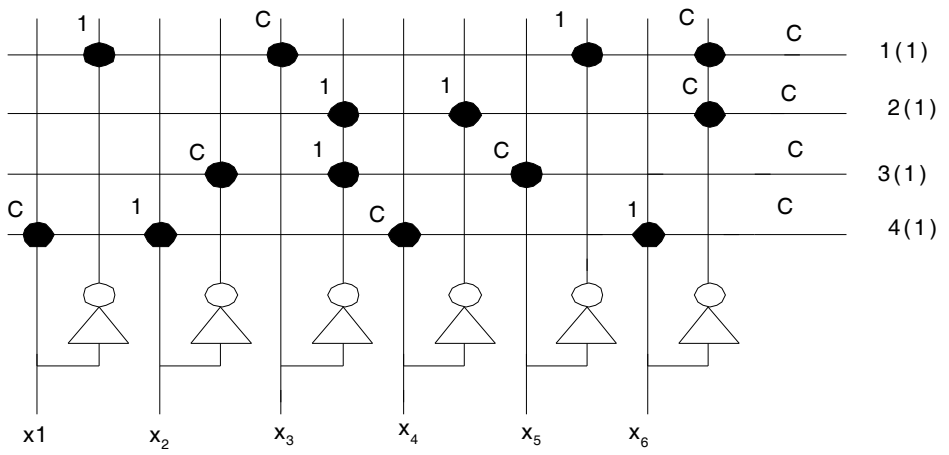

Figure 7: A test signals arrangement in the AND array

or several test vectors correspond to a given test signals arrangement in the AND array. In any case, the problem described is very complex and requires further research.

For Boolean function, a primary and irredundant cover is represented by a two-level combinational circuit for which a test set for all single faults can be found. Obviously, each minimal disjunctive form is irredundant. It is well-known, that a test set, which detects all single stuck-at-faults in irredundant two-level combinational circuits, detects also all multiple stuck-at-faults.

In this paper we assume that the PAL architecture (Fig.8) is used for the realization of two-level irredundant combinational circuits. The output lines of the decoder, in which the minterms of the function y are realized, are connected to the input lines of the OR gate.

Function y is irredundant in the sense that no one of its minterms can be deleted without changing its value. It is well known that irredundant logic can be tested.

In what follows we propose a procedure for constructing a test set which detects all single stuck-at-faults for the PAL architecture in Fig. 8.

In order to construct a procedure for finding the test set for all single stuck-at-faults, we represent the PAL architecture of Fig. 8 by the equivalent two-level combinational circuit with AND/OR gates (Fig. 9).

We apply signals $S_1 = \{100\ldots0, 010\ldots0, \ldots, 000\ldots1\}$ and $S_0 = \{000\ldots0\}$ to input lines of the OR gate in the combinational circuit of Fig.9. Test Signals from $S_1 \cup S_0$ detect $2q+2$ single faults of the OR gate if this gate is considered separately.

Definition 4. *A fault α dominates a fault β if for the set of tests T_α which detect α, and the set of tests T_β which detect β, we have $T_\beta \subset T_\alpha$.*

Let $T = \{X\}$ be the set of input vectors $X = x_1 x_2 \ldots x_n$ which act on input

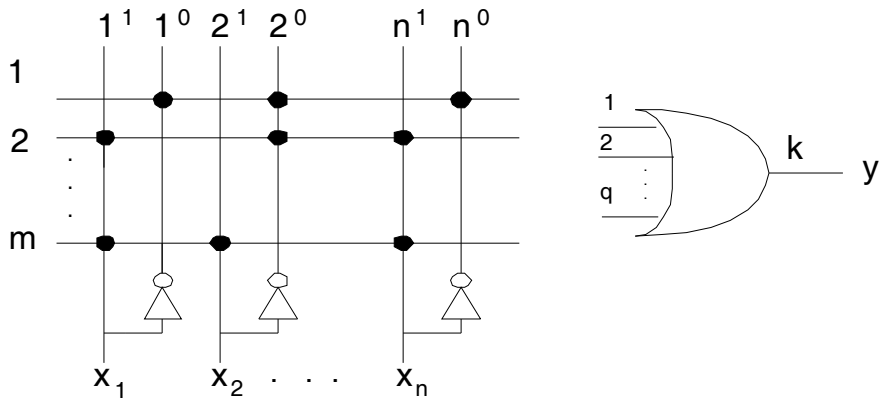

Figure 8: Implementation of Boolean function

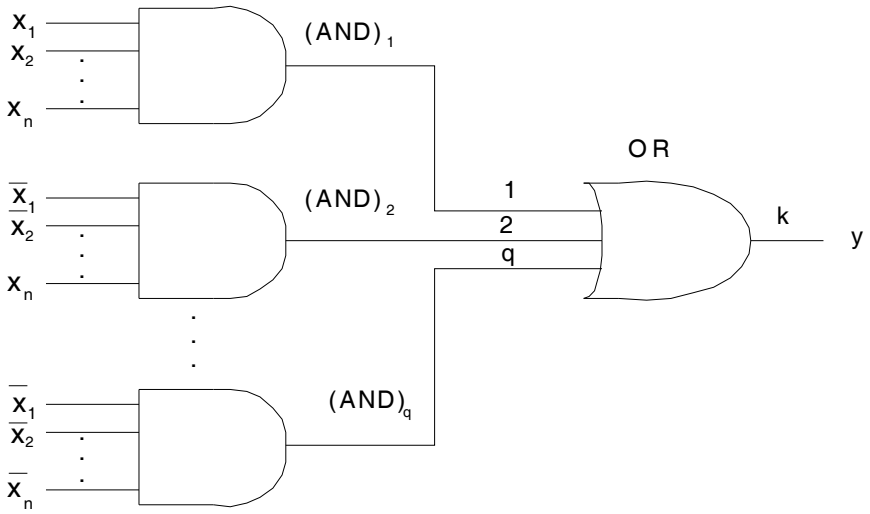

Figure 9: Equivalent combinational circuit

lines of the combinational circuit, i.e. before possible negations, with the property of producing a vector from $S = S_1 \cup S_0$ at the input lines of the OR gate.

Theorem 5. *The set T detects all single stuck-at-faults.*

Proof. Signal values at the output line k in the combinational circuit depend only on the signal values from the set $S_1 \cup S_0$, which act at input lines of the OR gate. The corresponding vectors from T will detect $2q+2$ single stuck-at-faults of the OR

gate. Since signals from $S_1 \cup S_0$ are dominant with respect to signals at the input lines in the combinational circuit, the set T will detect also all single stuck-at-faults of the input lines of the combinational circuit. Hence, the set T detects all single stuck-at-faults in the combinational circuit of Fig. 9. □

A cube which generates signal values 0 or 1 at a given line in the combinational circuit, is called 0-cover or 1-cover respectively.

A cube generates a signal $s \in \{0, 1\}$ at any line in combinational circuit which will be denoted by $i = s$. We shall say that the cube satisfies relation $i = s$.

Consider arbitrary lines **i** and **j** in the combinational circuit. Let $s_i, s_j \in \{0, 1\}$ be the signals at **i** and **j**, respectively. Then the following lemmas hold:

Lemma 6. *If cubes A and B satisfy relations $i = s_i$ and $j = s_j$ respectively, then the cube $C = A\phi B$ satisfies relation $(i = s_i) \wedge (j = s_j)$.*

Lemma 7. *Let S_i, S_j be sets of cubes satisfying $i = s_i$, $j = s_j$, respectively, then all cubes of the set $S_i \cup S_j$ satisfy relation $(i = s_i) \vee (j = s_j)$, while all cubes of the set $S_i \phi S_j$ satisfy relation $(i = s_i) \wedge (j = s_j)$.*

Our procedure for constructing a test set for all single stuck-at-faults consists of the following steps.

1. We construct covers for signals $S_1 \cup S_0$. To do this it is necessary to write $q + 1$ logical relations for the test signal transfer from input lines of the OR gate to the output line k (Fig. 9).

$$\begin{aligned}
&(1 = D) \wedge (2 = 0) \wedge \quad \ldots \quad \wedge (q = 0) \rightarrow (y = D),\\
&(1 = 0) \wedge (2 = D) \wedge \quad \ldots \quad \wedge (q = 0) \rightarrow (y = D),\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (28)\\
&(1 = 0) \wedge (2 = 0) \wedge \quad \ldots \quad \wedge (q = D) \rightarrow (y = D),\\
&(1 = C) \wedge (2 = C) \wedge \quad \ldots \quad \wedge (q = C) \rightarrow (y = C).
\end{aligned}$$

2. For each relation (i), $i = 1, 2, \ldots, q + 1$, construct the covers $P_{i1}, P_{i2}, \ldots P_{iq}$ for the output lines of the gate $(\mathrm{AND})_i$ $(i = 1, 2, \ldots, q)$.

3. Define cubes

$$Q_i = P_{i1} \phi P_{i2} \phi \ldots \phi P_{iq} \quad (i = 1, 2, \ldots, q + 1)$$

   Since the combinational circuit of Fig. 9 is irredundant, it cannot happen that any of the cube sets Q_i $(i = 1, 2, \ldots, q + 1)$ is empty.

4. By developing the test set Q_i we obtain a set T_i of test vectors. A test set for all single-stuck-at-faults is given by

$$T = T_1 \cup T_2 \cup \ldots \cup T_q \cup T_{q+1}.$$

The statement made in 3. is obvious. Namely, if a test set Q_i were empty, a certain combination of signals from $S_1 \cup S_0$ could not appear at the input lines of the OR gate. On the basis of Theorem 1, the single stuck-at-fault (1 or 0) at the corresponding input line of the OR gate cannot be detected, as well as the stuck-at-fault (1 or 0) of the output line of the OR gate. This would mean that some single faults in the combinational circuit of Fig.9 are undetectable.

The sum of products of a Boolean function can be realized by a two-level combinational circuit in NAND-NAND implementation. Obviously, the proposed procedure can be applied to these cases, as the following example shows.

Example 8. Let us determine a test set for all single stuck-at-faults for the combinational circuit of Fig. 10.

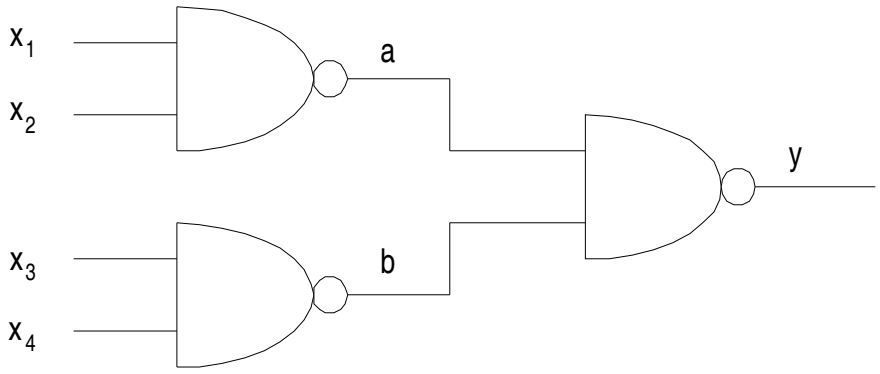

Figure 10: NAND-NAND implementation

Cascade NAND-NAND connection realizes an AND/OR function. Signals from the set $S_1 \cup S_0$ ($S_1 = \{C1, 1C\},\ S_0 = \{DD\}$) are applied to the input lines of the NAND gate of the second level.

The following relations hold

$$(a = C) \wedge (b = 1) \rightarrow (y = D), \tag{29}$$

$$(a = 1) \wedge (b = C) \rightarrow (y = D), \tag{30}$$

$$(a = D) \wedge (b = D) \rightarrow (y = C). \tag{31}$$

The corresponding cube sets are

$$\begin{aligned} Q_1 &= \{11XX\}\phi\{XX0X, XXX0\} = \{110X, 11X0\}, \\ Q_2 &= \{0XXX, X0XX\}\phi\{XX11\} = \{OX11, X011\}, \\ Q_3 &= \{0XXX, X0XX\}\phi\{XX0X, XXX0\} = \\ &= \{0X0X, 0XX0, X00X, X0X0\}. \end{aligned}$$

Cubes determined by $Q_1 \cup Q_2 \cup Q_3$ are

$$110X$$

$$11X0$$

$$0X11$$

$$X011$$

$$0X0X$$

$$0XX0$$

$$X00X$$

$$X0X0.$$

We adopt $X = 1$, which gives the following vector sequence

$$1101$$

$$1110$$

$$0111$$

$$1011$$

$$0101$$

$$0110$$

$$1001$$

$$1010.$$

It is interesting that this vector sequence can be reordered in such way that each vector switches from 0-1 and from 1-0 during the test sequence. The obtained test set is minimal.

It should be pointed out here that in some combinational circuits, the detecting test set for single stuck-at-faults, detects at the same time all multiple stuck-at-faults. For example, this happens in fan-out free combinational circuits.

Combinational circuit of Fig. 9, which is considered in this paper, is a two-level irredundant combinational circuit. In the introduction we have already pointed out that any set of tests which detects all single stuck-at-faults also detects all multiple stuck-at-faults [7].

Of course, the length of the test set for all single stuck-at-faults in a combinational circuit, realized by the PAL architecture, depends on the Boolean function which is implemented.

We use the PAL architecture in constructing parity bit generators. The even parity bit P is a function of input variables $x_1, x_2, \ldots, x_n$ defined by

$$f(x_1, x_2, \ldots, x_n) = \begin{cases} 1, & \text{if the number of 1' is odd in } (x_1, x_2, \ldots, x_n) \\ 0, & \text{otherwise.} \end{cases}$$

This function can be represented in the form

$$P = f(x_1, x_2, \ldots, x_n) = \sum_{\alpha \in N} x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n} \tag{32}$$

where $\alpha = (\alpha_1 \alpha_2, \ldots, \alpha_n) \in \{0,1\}^n$, $N = \{\alpha \mid f(\alpha) = 1\}$ and $x^0 = \bar{x}$, $x^1 = x$.

The odd parity bit is defined as $\overline{f(x_1, x_2, \ldots, x_n)}$ and can be treated similarly.

As is known, function P has only one sum-of-product expression and it is just the quoted canonical sum-of-product expression. It has $2^n/2$ minterms. Length of test set for all single stuck-at-faults is 2^n, what means that we have exhaustive testing, (see, for example, [1]).

5 Comparing fault models

We shall discuss now the relevance of the proposed fault model by comparing it with existing models. In fact, we shall show that, by the use of the proposed procedure, one can detect also the faults appearing in most frequently used PLA fault models: stuck-at-faults, crosspoint faults and bridging faults.

First, we shall show that a number of classical stuck-at-faults can be detected using the proposed procedure for the detection a multiple fault appearing in programmable points of PLA.

Within the classical stuck-at-fault model we can treat the PLA as a two-level circuit implementation and the typical stuck-at faults appear at input, input inverters, product lines and output lines. We shall show that in our model these faults are detected as well.

In order to simplify the analysis but without the loss of generality, let us consider in our model the detectable fault α_i $(i = 1, 2, \ldots, m)$ which consists of stuck-at-faults located at r programmable points of the horizontal line $i(1)$, $i = 1, 2, \ldots, m$ (we assume that the multiple fault ai consists of single stuck-at faults at the first r programmable point). In these points test signal of values C and D (depending of the fault type) are generated. Obviously, test cubes detecting fault α_i on the line $i(1)$, have rank r, i.e. they have r coordinates of value 0 or 1, while in the last $n-r$ coordinates we have variable X $(X \in \{0,1\})$.

Obviously, the cube which detects the fault α_i will detect also the stuck-at-fault $i(1)/s$, $s \in \{0,1\}$, $i = 1, 2, \ldots, m$, since we get $i(1) = C$ and $i(1) = D$. The fault $i(1)/s$ is a dominant fault w.r.t. classical stuck-at-faults located at first r inputs of PLA. The cube which detects the fault α_i will detect also the stuck-at-faults at r inputs of PLA.

Since the fault α_i is detectable, the test signals of value C or D are propagated from line $i(1)$, $i = 1, 2, \ldots, m$, to the output line $h(2)$, $h = 1, 2, \ldots, p$ during testing. Suppose that the horizontal line $i(1)$, $i = 1, 2, \ldots, m$ in the AND array crosses the output line of PLA $h(2)$, $h = 1, 2, \ldots, p$ in the OR array, which means that this lines have a crossing point with coordinates (h, i). Obviously, test vectors detecting the fault α_i on the output line $i(1)$ in the AND array, detect this fault at each output line $h(2)$, $h = 1, 2, \ldots, p$, as well.

This analysis can be carried out for other distributions of a multiple fault in PLA, located in programmable points. Of course, the set of stuck-at-faults, appearing in the classical stuck-at-fault model and detected by the use of our model for a multiple fault, depends on the distribution (configuration) of that fault, in particular, on whether the fault is located in the AND or in the OR array or in both. In this analysis we can use properties of dominant and equivalent faults.

By a similar analysis we can conclude that the test vectors, detecting a multiple fault in PLA following the proposed procedure, detect also a number of bridging faults.

In the sequel we shall show that also a number of cross points faults are detected by the proposed procedure for detection of a multiple fault.

Faults of types G, D, S and A for the AND/OR implementation of PLA have been defined in the introduction. The AND gates of the AND array will be denoted by $(\text{AND})_i$, $i = 1, 2, \ldots, m$, while the OR gates of the OR array will be denoted by $(\text{OR})_i$, $i = 1, 2, \ldots, p$.

The fault of type G (growth fault) appears in the case when the line j ($j = 1, 2, \ldots, n$) for the gate $(\text{AND})_i$, $i = 1, 2, \ldots, m$, is broken. The brake of this input line causes that the implicant realizing the gate $(\text{AND})_i$ does not contain the input variable x_j corresponding to the broken line j.

As already described, when applying the procedure for detecting the fault α_i located at r programmable points of the horizontal line $i(1)$, $i = 1, 2, \ldots, m$, at output line $i(1)$ of the gate $(\text{AND})_i$, $i = 1, 2, \ldots, m$, we get test signals of the value C or D. Signals $i(1) = C$ and $i(1) = D$ are dominant w.r.t. the signals at r input lines $1, 2, \ldots, r$ of the gate $(\text{AND})_i$. Test vectors detecting the fault α_i detect also r faults of type G $1/1, 2/1, \ldots, r/1$.

The fault of type D (disappearance fault) appears in the case when an output line of the gate $(\text{AND})_i$ is broken. If this line represents the input of the gate $(\text{OR})_i$, $i = 1, 2, \ldots, p$, then the function f_i, realizing this gate, does not contain the implicant corresponding to the broken line. The fault of the broken line $i(1)$ will be denoted by $i(1)/1$.

When detecting the fault α_i test signals of values C and D are propagated from the line $i(1)$ $i = 1, 2, \ldots, m$ to the corresponding output line $i(2)$, $i = 1, 2, \ldots, p$ of PLA. Single fault of type D will be detected in the case when the proposed procedure generates the test signal value C at the line $i(1)$.

A similar analysis can be carried out for the faults of types S and A.

6 Conclusion

In the proposed fault detection procedure for combinational circuits, test signals have values from the set $\{0, 1, C, D\}$, while no line in the combinational circuit can be in state X. Obviously, this fact simplifies the testing procedure. In applying the proposed procedure we go only forward, thus avoiding the problem of unsuccessful steps, which appear in structural testing methods. Writing logical relations for the test signal propagation is easier, since one uses only operation "$\wedge$". Unlikely to

the method for combinational circuit testing mentioned in the introductory part, the variable X can have only two values ($X \in \{0,1\}$). If, for a given single or multiple fault, only a part of the test vector set is determined, the computation can be considerably reduced.

In the proposed fault detecting procedure test signals also get values from the set $\{0,1,C,D\}$. Test signals are assigned to faulty programmable elements, which form a multiple fault. Logical relations describing the test signal propagation from the source points to the output lines of PLA are constructed. While executing the testing procedure no programmable point of PLA can have an undefined signal value, i.e. to be in state X, which makes the procedure smoother to a great extent. One should point out that the proposed procedure is related to multiple fault detection, which leaves the space for its development in direction of getting a test set which detects all multiple faults which can appear in PLA.

We shall mention some standard fault models which are used to describe faulty PLA. The typical models used in PLA testing are: stuck-at-faults, crosspoint faults and bridging faults. The fault model, proposed in this paper, differs from the above mentioned models. It enables a PLA testing procedure having some advantages in relation to standard testing methods.

The stuck-at-faults model is a classical model for describing a faulty PLA. We have noted in the introduction that the following faults appear at lines of PLA: input inverters stuck-at-1 and 0, AND gate inputs and outputs stuck-at-1 and 0, OR gate inputs and outputs stuck-at-1 and 0 and output inverters stuck-at-1 and 0. It is well-known that while using the stuck-at-fault model a number of crosspoint faults as well as a number of bridging faults is detected. The starting point of the proposed procedure for detecting a multiple fault is the fact that the fault appears in programmable points of PLA, while the produced PLA corresponds to the designed one with the hypothesis that no mistakes in designing PLA can occur. Such a model reflects to a greater extent a real PLA. One should also point out that by the proposed procedure for detecting a multiple fault also a number of stuck-at-faults and bridging faults are detected as well. The number of detected stuck-at-faults, mentioned above, depends on the arrangement of the multiple fault in PLA.

The proposed procedure differs from the one used in the crosspoint faults model in which the faults of types G, D, S and A appear, where we test for incorrect logical connection in the AND and OR arrays. Obviously, the model involving the faults of types G, D, S and A is rather complicated one, while in our model we have only the faults of the type of disconnection of a programmable element or the faults of the type of a short connection of a programmable element. This means that crosspoints, in which faulty elements are located, obtain logical values 1 and 0.

With proposed procedure for multiple fault detection in the PLA, all test vectors are determined. In determining test vectors for a single or multiple fault, the quantity of work depends on the structure of the fault, and its disposition in PLA.

In this paper the stress is laid on the detection of a multiple stuck-at-fault for PLA circuits. The proposed procedure is also applied to some specific circuits, such as the PAL architecture realizing a irreduntant Boolean function. In this paper we

have shown that the test set which detects all single stuck-at-faults detects at the same time all multiple stuck-at-faults.

The proposed procedure for detecting multiple fault in PLA is very simple since the signal propagation is described by simple logical relations. Let us also point out that no extra hardware is necessary. Proposed procedure for fault detection can be applied to fault detection in integrated circuits implementing Boolean functions which may appear in realization of a computer architecture.

References

[1] M. Abramovici, M.A. Breuer, and A.D. Friedman, *Digital Systems Testing and Testable Design*, IEEE Press, 1995.

[2] J.M. Acken, S.D. Millman, *Fault model evolution for diagnosis: accuracy vs. precision*, Proc. IEEE Custom Integrated Circuits Conference, 1992, pp. 13.4.1–13.4.4.

[3] V. Agarwal, *Easily Testable PLA Design, in Advances in CAD for VLSI*, vol. 5, VLSI Testing, T. Wiliams (ed.), Elsevier Science Publishers, Amsterdam, 1986.

[4] V.Agraval, C.Kime, K.Saluja, *A Tutorial on Built-in-Self-Test*, IEEE Design & Test of Computers, March 1993, pp. 73-80, June 1993, pp. 69–77, IEEE CS Press.

[5] D.B. Armstrong, *On Finding a Nearly Minimal Set on Fault Detection Tests for Combinational Logic Nets*, IEEE Trans. Electronic Computers EC-15, 1966, pp. 66–73.

[6] P. Bibilo, N. Kirienko, *Block Synthesis of Combinational Circuits in the Basis of PLA and Library Gates*, The International Workshop on Discrete-Event System Design, June 27-29, 2001, Poland.

[7] M.A.Breuer, A.D. Friedman, *Diagnosis & Reliable Design of Digital Systems*, Computer Science Press, Inc., p. 68, 1976.

[8] C.W. Cha, W.E. Donath, F. Ozguner, *9-V Algorithm for Test Pattern Generation of Combinational Digital Circuits*, IEEE Trans. on Computers, Vol. C-27, No. 3, March 1978, pp. 193–200.

[9] H.Fujiwara, S.Tolda, *The Complexity of Faults Detection Problems for Combinational Logic Circuits*, IEEE Trans. Comput., pp. 550–560 , June 1982.

[10] P. Goel, *An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits*, IEEE Trans. on Computers, Vol. C-30, No. 3, March 1981, pp. 215–221.

[11] I. Kohavi, Z.Kohavi, *Detection of Multiple Faults in Combinational Logic Networks*, IEEE Transactions on Computers, pp. 556–568, June 1972.

[12] K. Lai, P.K. Lala, *Multiple Fault Detection in Fan-Out Free Circuits Using Minimal Single Fault Test Set*, IEEE Trans. Comput. June 1996, pp. 763–765.

[13] P.K. Lala, *Digital Circuits Testing and Testability*, Academic Press, London 1997.

[14] S. Mourad, Y. Zorian, *Principles of Testing Electronic Systems*, John Wiley & Sons, 2000.

[15] K.S. Ramanatha, N.N. Biswas, *Design of Crosspoint-Irredundant PLAs Using Minimal Number of Control Inputs*, IEEE Trans. of Comp., Vol. 37, No. 9, September 1988, pp. 1130–1134.

[16] R.K. Ranjan, *Dynamic Reordering in a Bredth-First Manipulation Based BDD Package: Chalanges and solution* Proc. Of the 1997 IEEE Int. Conf. on Computer Design: VLSI in Computers & processors, ICCD'97, Austin, Texas, October 1977, pp. 17–26.

[17] J.P. Roth, *Diagnosis of Automata Failures: A Calculus and a Method*, IBM Journal, 1966, pp. 278–291.

[18] J.E. Smith, *Detection of Faults in Programmable Logic Arrays*, IEEE Trans. on Computers., vol.C-28, No 11. November 1979, pp. 845–853.

[19] C.E. Stroud, J.R. Bailey, J.R. Emmert, *A New Method for Testing Reprogrammable PLAs*, Journal of Electronic Testing, December 2000, pp.635–640.

Received April, 2003