ISSN 0005-1179, Automation and Remote Control, 2021, Vol. 82, No. 11, pp. 1949–1965. © Pleiades Publishing, Ltd., 2021.
Russian Text © The Author(s), 2021, published in Avtomatika i Telemekhanika, 2021, No. 11, pp. 148–168.

Constructing a Sequence Detecting Robustly Testable Path Delay Faults in Sequential Circuits

A. Yu. Matrosova[1*], **S. V. Chernyshov**[1**], **O. Kh. Kim**[1***], **and E. A. Nikolaeva**[1****]

[1] *Tomsk State University, Tomsk, 634050 Russia*

e-mail: *[*] *mau11@yandex.ru,* [**] *svchernyshov@mail.ru,* [***] *oh.kim@yandex.ru,* [****] *nikolaeve-ea@yandex.ru*

Received May 22, 2020; revised February 1, 2021; accepted March 16, 2021

Abstract—We propose a method for constructing a sequence of Boolean vectors of input variables that delivers test pairs (v_1, v_2) of neighboring vectors in the space of input and internal variables for robustly testable path delay faults (robust Path Delay Faults (PDFs)) in sequential logic circuits. The purpose of this work is to clarify the possibility of constructing a test sequence for a given subset of paths without using scanning technologies, i.e., without additional hardware costs within the constraint of the sequence length for a single path. The experiments carried out show that test sequences can be constructed not for all paths (sometimes for none) for which test pairs exist in the combination component of the sequential circuit.

Keywords: sequential logic circuit, homing sequence, Reduced Ordered Binary Decision Diagram (ROBDD), robustly testable path delay fault (PDF), rising (falling) transition

DOI: 10.1134/S0005117921110102

1. INTRODUCTION

One of the main parameters of high-performance logic circuits is high clock speed, i.e., high speed of operation of the circuit. It is determined by the speed of signals passing through the logic gates along the paths connecting the inputs and outputs of the circuit. Manufacturers are striving to increase the clock speed.

To determine the circuit clock frequency, one allocates the path with the maximum delay—the critical path. The delay in this path determines the speed at which the circuit operates. However, at a high speed of operation and a high level of integration, there appear capacities, inductances, and resistances in the circuits that are not foreseen by the developers and cannot be calculated in advance. This leads to additional delays in the circuit and a decrease in its design speed, which is undesirable. The Path Delay Fault (PDF) model is one of the most common logic delay models used in practical logic testing. The path delay fault model assumes that the delays in individual path elements and in communication lines between them are small, but path delay as a whole can exceed the time between adjacent clock signals and distort circuit operation. Robustly and nonrobustly testable path delay faults are distinguished. A robustly testable fault is understood as a path delay fault that is detected regardless of the existence of other path delays in the circuit that exceed admissible values. Otherwise, the fault is called nonrobustly testable. When a robustly testable fault occurs, one can accurately determine the path along which the delay occurs, that is, exceeds the time between adjacent clock signals. Appearance of a nonrobustly testable fault does not provide such an opportunity. Information about the resulting delays allows one to modify the circuit so as to maintain its design performance or to mask the effect of the detected delay [1] with the same purpose. The pairs of vectors (v_1, v_2) are used to test path delays. The vectors in each pair can differ from each other in the variable that marks the beginning of the path under consideration and possibly also in the values of other variables. In the future, we will agree to use (Boolean) "vectors" and "test-pair sets" as synonyms. The delays of the opposite (rising and falling) transitions of signal values may differ, so in the general case, it is necessary to have two test pairs for each path. By a rising transition, as is customary in foreign publications, we mean a sequence of changes in signal values along the path that ends with a change from zero to unit signal at its output. Accordingly, a falling transition is a sequence of changes in signal values along the path that ends with a change from unit to zero signal at its output. Let the test-pair vector v_1 arrive at the circuit at time t with the arrival of a clock signal. When the next clock signal arrives, the vector v_2 arrives at the circuit. If a signal is observed at the output of the circuit with a value that differs from the expected one with the arrival of the third sync signal, then we assume that there is a delay in the paths of the circuit.

In practice, different scanning methods are used when testing path delays. When these methods are implemented within the Launch-on-Shift (LOS) technology, one of the vectors of the pair is a test set for a constant fault of the internal pole of the circuit, and the second vector is obtained by shifting the first. When using Launch-on-Capture (LOC) technology, the second vector is obtained based on the response of the combination component of the circuit to the first vector. It is clear that with such methods of generating pairs of vectors, it is far from being always possible to obtain test pairs for robustly testable path delay faults. Usually, when using these technologies, it is possible to detect approximately 20% of such faults, while the use of the precise methods [2], which guarantee the construction of a test pair for a robustly testable fault if it exists, allows detecting from 40% to 90% and more such faults for a given set of paths of the combination component of the sequential circuit. The precise methods focus on reducing power consumption. The problem of reducing power consumption in delay testing is explored in many publications [3–7]. Another problem in delay testing is the high hardware cost associated with delivering tests within scanning technologies. As is well known [8, 9], scanning technologies use special triggers in feedback lines that are more complex than D-flip-flops. In the circuit operational mode, these triggers perform, by analogy with D-flip-flops, a one-cycle delay of the signal associated with the state (internal) variable, and in the test mode, the circuits form a shift register into which the component of the test vector corresponding to the internal variables is introduced clock cycle by clock cycle. Additional hardware is also required to separate the operating and testing modes of a sequential circuit. The exact methods can be applied within the framework of Random Access Scan (RAS) technologies, the implementation of which, unfortunately, involves more hardware costs than the use of LOS or LOC technologies. Thus, the listed methods of testing path delays are associated with considerable hardware costs, which we would like to avoid. The studies carried out in the present paper permit one to find out whether this is always possible.

There are many test pairs of adjacent Boolean vectors (all or some) that detect a robustly testable path delay fault in the combination component of a sequential circuit. It is required to construct an input sequence that delivers one (arbitrary) test pair in a given set from the initial state q_0 of a sequential circuit within the constraint on the sequence length. Algorithms for constructing such sequences are proposed under the following conditions:

1. The beginning of the path is marked by an input variable.
2. The beginning of the path is marked by a state (internal) variable of the circuit.

If the desired sequences can be found for each of the paths in a given set, then we detect path delays without additional hardware costs by combining them.

Section 2 discusses the properties of tuple pairs for robustly testable path delay faults. Section 3 describes a procedure for obtaining a Reduced Ordered Binary Decision Diagram (ROBDD graph) representing the set of all test pairs of neighboring tuples for robustly testable path delay faults. In Sec. 4, we present algorithms for constructing a sequence that delivers a test pair for a robustly tested path delay fault in a sequential circuit from the initial state. Section 5 discusses experimental results.

2. SOME PROPERTIES OF TEST PAIRS OF TUPLES FOR ROBUSTLY TESTABLE PATH DELAY FAULTS

We have a single-output circuit C and the corresponding equivalent normal form (ENF). In [10], the problem of constructing pairs of test cases for robustly and nonrobustly testable path delay faults is considered based on ENF analysis. The sets v_2 of test pairs are test sets for constant faults of ENF letters.

In the case of a rising transition, a 0-constant malfunction of the ENF letter corresponding to a path α is tested (a_p-fault). In the presence of a malfunction, all occurrences of the letter are replaced in the ENF by the constant 0. The test set that detects this malfunction is the set v_2 of the test pair, which generates a change in the signal at the circuit output from zero to one.

In the case of a falling transition, a 1-constant failure of the ENF letter corresponding to a path α is tested (b_p-fault). In the presence of a malfunction, all occurrences of the letter are replaced in the ENF by the constant 1. The test set that detects this malfunction is the set v_2 of the test pair, and it generates a change in the signal at the circuit output from one to zero.

It was shown in [10] that test pairs that detect robustly testable path delay faults satisfy the following conditions:

1. If v_2 is a a_p-test tuple, then v_1 is a b_p-test tuple of the test pair and vice versa.
2. The function implemented by circuit C takes opposite values on the tuples of a test pair.
3. The minimum-cover interval u of the vectors v_1, v_2 is orthogonal to all ENF conjunctions not containing the letter associated with the path α.
4. The tuples (v_1, v_2) of the test pair are generated by the same nonempty conjunction.

When the tuple v_1 is replaced by the tuple v_2, the circuit can get into an interim state, which is a test set v_1 for another path of the circuit. As a result, one can determine the delay of a path other than the considered one. This situation is excluded if condition 3 is satisfied. To check condition 3, it is proposed to use the ENF scheme in [10]. However, as a rule, the ENF turns out to be cumbersome even for small circuits. For neighboring sets of the test pair (v_1, v_2), no intermediate states arise when passing from v_1 to v_2. The use of this fact eliminates the need to analyze the ENF.

Theorem 1. *The neighboring tuples of the test pair (v_1, v_2) in which v_2 is the a_p-test tuple and v_1 is the b_p-test tuple, or vice versa, detect a robustly testable path delay fault for both excursions of the signal values of this path (for rising and falling transitions).*

Theorem 2. *If there exists a test pair of tuples (b_p, a_p) that are not neighboring and detect a robustly testable path delay fault α, then there exists a test pair of neighboring tuples that detects this delay.*

The very fact of the existence of a test pair of neighboring tuples for robustly testable faults in the presence of a test pair for nonneighboring tuples, perhaps for the first time, was noted as a consequence of theorems on the properties of the ENF when substituting constants identical for neighboring tuples into it [11]. From the ENF, there remains a disjunction of the letters associated with the variable that marks the beginning of the path. The configuration of these letters allows one to determine what type of path delay is tested by the presented pair of adjacent tuples. Theorem 2 confirms this result on the basis of studying the properties of ENF conjunctions (in which no variables are replaced by constants) that ensure the existence of a test pair for robustly testable path delay faults [10].

It follows from what has been said that if the path under consideration is robustly testable, then there exists a test pair for it consisting of neighboring tuples.

This also means that to establish the existence of a test pair for a robustly testable path delay fault, it suffices to restrict oneself to the search for test pairs consisting of neighboring test tuples. The algorithm proposed in [12] for constructing all test pairs of adjacent tuples for detecting robustly testable path delay faults guarantees that all robustly testable paths are found in the circuit.

To construct the set of all test pairs of neighboring vectors for robustly testable faults of the path under consideration, in [12] it is proposed to precompute the Boolean difference (Boolean derivative) of this path. The method for calculating the Boolean difference and representing it in the form of a ROBDD graph $R(D_{\text{path}})$ is based on performing operations on ROBDD graphs extracted from fragments of a combination circuit characterized by polynomial complexity.

3. OBTAINING TEST PAIRS OF NEIGHBORING TUPLES OF ROBUSTLY TESTABLE FAULTS FROM ROBDD $R(D_{\text{path}})$

All kinds of test tuples v_2 for both signal excursions are presented in the form of the ROBDD $R(D_{\text{path}})$ [12]. Recall that in the case of a rising transition, v_2 is a test tuple for the a_p-fault of the letter marking the beginning of the path α. For a falling transition, v_2 is a test tuple for the b_p letter malfunction marking the beginning of the path α. Here is the procedure for extracting all test pairs for the path under consideration [12]. Let the beginning of the path α be marked by the variable x_i without inversion.

The graph $R_{\text{rise}} = R(D_{\text{path}})\,\&\,x_i(R(D_{\text{path}})\,\&\,\overline{x}_i)$ is a ROBDD graph representing all tuples v_2 for a rising transition.

The graph $R_{\text{fall}} = R(D_{\text{path}})\,\&\,\overline{x}_i(R(D_{\text{path}})\,\&\,x_i)$ is a ROBDD graph representing all tuples v_2 for a falling transition.

Two formulas are given for each type of excursions in the signal values of the path α, since the ENF letter associated with the path under consideration can have different inversion signs.

By R'_{rise} we denote a ROBDD graph obtained from R_{rise} by removing the variable x_i, and by R'_{fall}, a ROBDD graph obtained from R_{fall} by removing the variable x_i. In both cases, the signs of the variable x_i are irrelevant.

By symbol R_{rob} we denote a ROBDD graph representing test pairs of tuples neighboring in the variable x_i for robustly testable path delay faults α, $R_{\text{rob}} = R'_{\text{rise}}\,\&\,R'_{\text{fall}}$.

It follows from the procedure of constructing R_{rob} that the graph does not contain the variable x_i marking the beginning of the path. The path from the root of the graph R_{rob} to its 1-terminal vertex is a conjunction such that a Boolean vector in the space of variables $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$ specifies a test pair in the space of n variables for the path in question. One of the vectors of the pair is obtained by assigning the value of 0 to the variable x_i and the other, the value of 1.

If the value of 1 is reached at the vector v_2 and the output of the single-output circuit (for a multioutput circuit, at the output associated with path α), then v_2 is the vector for a rising transition; if the value is 0, then v_2 is the vector for a falling transition.

In the sequential scheme, the pairs of neighboring Boolean vectors defined by the ROBDD graph R_{rob} are associated with complete states; i.e., the input components and components by state variables are distinguished in the vectors of the pair.

From the test pair, we form triples of vectors that detect the delays of inverse excursions of the signals of the path under consideration, either v_2-v_1-v_2 or v_1-v_2-v_1. This means that when using test pairs from neighboring Boolean vectors, it is possible to detect opposite excursions in the signal values of the path under consideration in the combination component with three vectors instead of four.

Table 1. STG-description of the behavior of a circuit with memory

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

However, if the state variables in the combination component are not available, it is proposed to separately deliver pairs of the sequential circuit from the initial state q_0 for each of the sequences of excursions of the path under consideration. The point here is that it is necessary not only to get into a state that corresponds to the beginning of a sequence of three vectors, but then to be in a special chain of transitions generated by the triple of test vectors; the longer the chain, the lower the probability of getting into it.

4. ALGORITHMS FOR CONSTRUCTING A SEQUENCE THAT DETECTS A ROBUSTLY TESTED PATH DELAY FAULT IN A SEQUENTIAL CIRCUIT

Let us present algorithms for constructing a sequence of vectors of input variables of a sequential circuit that delivers a test pair to the inputs of the combination component of the circuit from the initial state q_0. Recall that we are talking about the neighboring Boolean vectors of the test pair with respect to the variable marking the beginning of the path under study. These algorithms, together with the corresponding test pairs, form sequences that detect robustly testable path delay faults—one input sequence for each sequence of excursions in the signal values of the path under consideration from a given set of paths.

Let the sequential diagram be derived from the State Transition Graph (STG) description in which the states are encoded. The STG description of the behavior of a discrete sequential device has been used in foreign publications since the last century. It is used in conditions when the symbols of the input and output alphabet of a finite state machine (an abstract model of the behavior of a device with memory) have already been coded by the developer of the device. In the STG description, the condition for the transition from one state to another (with the simultaneous formation of the output symbol) is represented as a ternary vector in the space of input variables and state variables of a sequential circuit. This allows one, in the general case, to compactly specify a set of conditions for this transition in comparison with the transition-exit tables used in the theory of finite automata. Table 1 shows an example of such a description. Here the set of states is represented by the symbols $\{1, 2, 3, 4\}$.

Let us associate the states with Boolean vectors in the space of variables z_1, z_2 as follows: 1—$\overset{z_1}{0}\,\overset{z_2}{0}$, 2—$\overset{z_1}{0}\,\overset{z_2}{1}$, 3—$\overset{z_1}{1}\,\overset{z_2}{0}$, 4—$\overset{z_1}{1}\,\overset{z_2}{1}$, to obtain a table that represents the system of partial Boolean functions on the set of input variables and state variables (Table 2). In the table, the transition functions are marked by variables z_1', z_2', and the output functions, by the variables y_1, y_2, and so on.

Table 2. Description of the behavior of a circuit with memory after state encoding

x_1	x_2	x_3	z_1	z_2	z_1'	z_2'	y_1	y_2	y_3	y_4	y_5
0	–	–	0	0	0	0	0	0	0	1	0
–	0	–	0	0	0	0	0	0	0	1	0
1	1	–	0	0	0	1	1	0	0	1	0
–	–	0	0	1	0	1	0	0	1	1	0
–	–	1	0	1	1	0	1	0	1	1	0
1	0	–	1	0	1	0	0	1	0	0	0
0	–	–	1	0	1	1	1	1	0	0	0
–	1	–	1	0	1	1	1	1	0	0	0
–	–	0	1	1	1	1	0	1	0	0	1
–	–	1	1	1	0	0	1	1	0	0	1

This table is a task for the synthesis of a device. The codes of minimum length have been used to encode the states. Other codes are often used when receiving a synthesis task, for example, constant-weight codes, Berger codes, and their modifications [13, 14], and others.

Consider the following facts:

1. If there are no loops in the transition diagram corresponding to the STG description, then it is impossible to construct a sequence that delivers a test pair for paths whose beginning is marked by the input variable of the circuit.
2. If the resulting set of codes (in the work area specified by the STG description) lacks neighboring vectors as a result of the coding of states, then it is impossible to construct a sequence that delivers a test pair for paths whose origins are marked by a state (internal) variable in the sequential circuit.

Thus, the set of all pairs of neighboring Boolean vectors that define the complete states of the circuit and detect robustly testable delays of the considered path in the combination part of the sequential circuit is represented by the ROBDD graph R_{rob}. In the ROBDD R_{rob}, the Shannon expansion is performed first in terms of internal variables and then in terms of input variables. This follows [12] from the method of its construction.

Bearing in mind that the test pairs (v_1, v_2) specified by R_{rob} consist of an input and an internal component, we represent them as $v_1 = v_1^{\text{in}}$, v_1^s; $v_2 = v_2^{\text{in}}$, v_2^s; i.e., the test pair will be further defined as $(v_1^{\text{in}}, v_1^s; v_2^{\text{in}}, v_2^s)$.

We consider the following situations:

(a) The adjacent Boolean vectors extracted from R_{rob} differ in the input variable.

(b) The adjacent Boolean vectors extracted from R_{rob} differ in the internal variable.

Case (a). Consider a test pair $(v_1^{\text{in}}, v_1^s; v_2^{\text{in}}, v_1^s)$ of vectors that must be fed to the combination component of the sequential circuit to detect delays in a given path α under the conditions of difference between the vectors of the test pair (components $v_1^{\text{in}}, v_2^{\text{in}}$) in the input variable x_i and given the coincidence of the components in the state variables. For simplicity, we assume that the variable marking the given path has no inversion. Let us describe the procedure for delivering a test pair at the appropriate times:

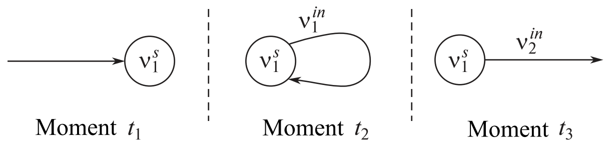

Fig. 1.

1. First, using a suitable input sequence, the construction of which will be described below, we set the circuit (at time t_1) to the state v_1^s. Here neither the input signal nor the state from which the transition to the state v_1^s is performed is of any importance.
2. From the state v_1^s, under the action of the input signal v_1^{in} arriving at time t_2 (it is at this time that the input of the combination component of the sequential circuit of the first vector v_1^{in}, v_1^s of the test pair is supplied), we go to the state v_1^s; i.e., we implement a loop in the automaton transition diagram so as to reach the state v_1^s at time t_3.
3. In the state v_1^s at time t_3, we supply the vector v_2^{in} to the inputs of the circuit; i.e., we ensure the arrival of the second vector v_2^{in}, v_1^s of the test pair at the inputs of the combination component. It does not matter what the next state of the circuit will be, it is only important that there should be a change in the signal at the output associated with the path under consideration.

At time t_4, we observe a delay fault at the output of the combination component corresponding to the path if a delay fault occurs.

Note that the neighboring vectors of the test pair arrive at consecutive times. The transitions at the time instants t_1, t_2, and t_3 are illustrated in Fig. 1.

Thus, to deliver a test pair, it is necessary to get from the initial state q_0 to the state v_1^s. This can be done by constructing a sequence that sets the circuit from the initial state to some state from the set of states of the type v_1^s. Moreover, not all states with a loop are of interest but only those in which the loop is realized by the input component v_1^{in} contained among the complete states represented by R_{rob}. Having provided the supply of the input component in this state, we then form the vector v_2^{in}, v_1^s by replacing the value of the variable x_i in v_1^{in} with the inverse.

We will use the algorithm proposed in [15] to construct a homing sequence from the initial state q_0 to a state from the given set. In this algorithm, the set of states is represented by the ROBDD graph R^{s_0}. Constructing the homing sequence from the initial state is a common part for different types of signal excursions and different types of variables marking the beginning of a path. For each of these situations, one has to construct its own graph R^{s_0}.

Thus, we construct a set R^{s_0} for the test pair in which the neighboring vectors differ in the input variable x_i. From the STG description, we select a fragment that represents loops in the transition table of the corresponding finite state automaton. The fragment consists of rows of the form

$$\begin{gathered}
k_1 q_i \to q_i \\
k_2 q_i \to q_i \\
\cdots \\
\cdots \\
k_m q_i \to q_i \\
k_{m+1} q_j \to q_j \\
\cdots \\
\cdots \\
k_{m+s} q_j \to q_j \\
\cdots
\end{gathered}$$

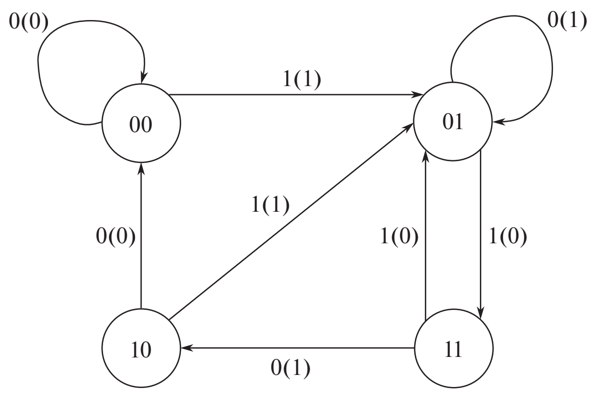

Fig. 2. Transition diagram in a sequential circuit.

Here the k_i are the conjunctions (ternary vectors) on the set of input variables of the circuit, and $q_i, q_j, \ldots$ are the states of the STG description (the states of a finite automaton) represented by Boolean vectors when encoding. The left-hand side of the considered fragment, i.e., the conjunctions of the input variables, together with the states assigned to them, are represented in the form of the ROBDD R_p. This graph will be useful when considering all paths from the presented set the beginnings of which are marked by the input variables of the sequential circuit. Let us move on to considering the given path α.

The sequence of steps for a falling transition of the path α:

1. From R_{rob} we form a_p-test tuples (tuples v_1^{in}, v_1^s) of the test pair that are associated with the input letter x_i, representing them by an appropriate ROBDD graph $R_{\text{rob}/x_i} = R_{\text{rob}} \,\&\, x_i$.
2. Of the a_p-test tuples, we select those which generate loops: $R_{a/p} = R_{\text{rob}/x_i} \,\&\, R_p$; i.e., we obtain the set of all vectors of the form v_1^{in}, v_1^s that are assigned to time t_2 when testing for a path delay.
3. In $R_{a/p}$, we isolate the set of internal states and represent it by a graph R^{s_0}.
4. Having obtained the vector v_1^{in}, v_1^s, we then form a vector v_2^{in}, v_1^s that is generated by the graph R_{rob} and supply it at time t_3. The vector v_2^{in} is different from the vector v_1^{in} in the variable x_i.

After finding a test pair for a rising transition, we perform the following steps of the algorithm for b_p-test tuples.

The sequence of steps for a rising transition of the path α:

1. From R_{rob} we form b_p-test tuples (tuples v_1^{in}, v_1^s) of the test pair that are associated with the input letter $\overline{x}_i$, representing them by an appropriate ROBDD graph $R_{\text{rob}/\overline{x}_i} = R_{\text{rob}} \,\&\, \overline{x}_i$.
2. Of the b_p-test tuples we select those which generate loops: $R_{b/p} = R_{\text{rob}/\overline{x}_i} \,\&\, R_p$; i.e., we obtain the set of all vectors of the form v_1^{in}, v_1^s that are assigned to time t_2 when testing a path delay.
3. In $R_{b/p}$, we isolate the set of internal states and represent it by a graph R^{s_0}.
4. Having obtained the vector v_1^{in}, v_1^s, we then form a vector v_2^{in}, v_1^s that is generated by the graph R_{rob} and supply it at time t_3. The vector v_2^{in} is different from the vector v_1^{in} in the variable x_i.

Consider an illustrative example. Let the behavior of an automaton be represented by the transition diagram shown in Fig. 2. In the transition diagram, the states are encoded as follows: 1 is $\overset{z_1 z_2}{0\,0}$, 2 is $\overset{z_1 z_2}{0\,1}$, 3 is $\overset{z_1 z_2}{1\,0}$, and 4 is $\overset{z_1 z_2}{1\,1}$. We have a special case of the STG description in which the conditions of transitions are represented by Boolean vectors. Assume that the initial state q_0 is represented by a vector $\overset{z_1 z_2}{0\,0}$ corresponding to symbol 1.

In the diagram, the labels on the arcs are the input and output (in parentheses) vectors. In this case, these are one-component vectors. When coding states, this description turns into a specification of a system of Boolean functions (see Table 3).

From the table, we obtain a realization of the system of Boolean functions in the form of a system of DNFs of input variables and state variables,

$$\begin{aligned} z_1' &= \overline{x} z_1 z_2 \vee x \overline{z}_1 z_2; \\ z_2' &= x \vee \overline{z}_1 z_2; \\ y &= x \overline{z}_2 \vee \overline{x} z_2. \end{aligned}$$

This system realizes the combination component of the sequential circuit depicted in Fig. 3.

Consider the path $x \to 4 \to 7 \to y$.

Let us calculate the Boolean difference for this path. In the example, we use operations over DNFs rather than ROBDD graphs for simplicity. To preserve compliance with the text of the algorithm, the symbol D (instead of DNF) will be accompanied in brackets by the designation of the corresponding ROBDD graph in the text. Here the symbols $U_1, U_2, \ldots$ denote the outputs of the corresponding circuit elements,

$$\begin{aligned} D_{U_7}/D_{U_4} &= (U_1 \vee U_4)|_{U_4=0} \oplus (U_1 \vee U_4)|_{U_4=1} = \overline{U}_1; \\ D_{U_4}/D_x &= (\overline{x} z_2)|_{x=0} \oplus (\overline{x} z_2)|_{x=1} = z_2; \\ D_{\text{path}} &= (\overline{x \overline{z}_2}) z_2 = (\overline{x} \vee z_2) z_2 = z_2. \end{aligned}$$

Thus, $D_{\text{path}} = z_2$.

Further, we calculate $D(R_{\text{rob}})$,

$$\begin{aligned} D(R_{\text{rise}}) &= \overline{x} z_2; \\ D(R_{\text{fall}}) &= x z_2; \\ D(R'_{\text{rise}}) &= z_2; \\ D(R'_{\text{fall}}) &= z_2; \\ D(R_{\text{rob}}) &= z_2. \end{aligned}$$

From Table 3, we select transitions associated with loops in the transition diagram. We define conditions ensuring these transitions by the appropriate DNF,

$$D(R_p) = \overline{x z_1} \overline{z}_2 \vee \overline{x z_1} z_2.$$

Table 3. Tabular definition of the system of Boolean functions

x	z_1	z_2	z_1'	z_2'	y
0	0	0	0	0	0
0	0	1	0	1	1
1	0	0	0	1	1
0	1	0	0	0	0
1	1	0	0	1	1
0	1	1	1	0	1
1	1	1	0	1	0
1	0	1	1	1	0

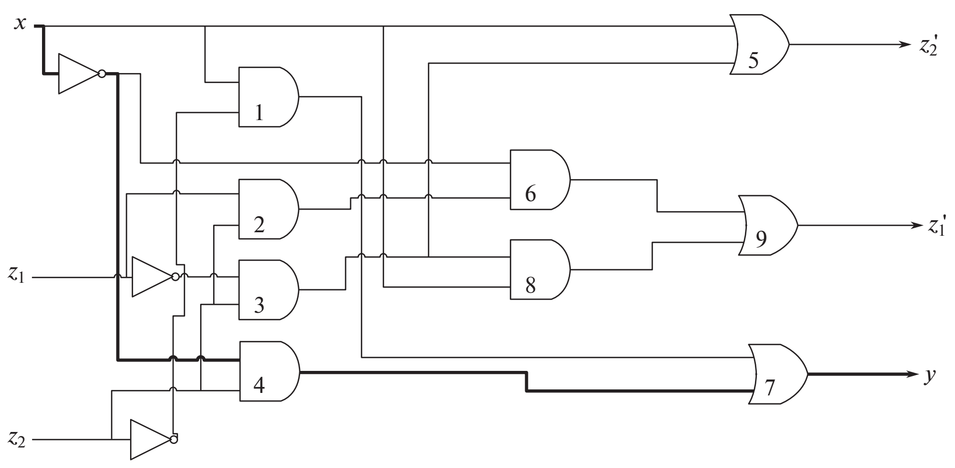

Fig. 3. Beginning of a path in the circuit is marked by variable x.

For a falling transition, we perform steps 1–3 of the algorithm to obtain the state $D(R^{s_0}) = \overset{z_1}{0}\overset{z_2}{1}$:

1. $D(R_{\text{rob}/\overline{x}}) = R_{\text{rob}}\overline{x} = \overline{x}z_2$.
2. $D(R_{a/p}) = \overline{x}\overline{z}_1 z_2$.
3. $D(R^{s_0}) = \overline{z}_1 z_2$.

In what follows, we construct a sequence that sends the circuit from the initial state $q_0(\overset{z_1}{0}\overset{z_2}{0})$ to the state $\overset{z_1}{0}\overset{z_2}{1}$.

We try to find a sequence of length one by multiplying DNFs corresponding to state variables, $D(R^{s_1}) = xz_1 \vee x\overline{z}_2 \vee \overline{x}\overline{z}_1 z_2$.

It follows from the resulting DNF that under the action of the input symbol $\overset{x}{1}$, from the state $\overset{z_1}{0}\overset{z_2}{0}$ we get into the state $\overset{z_1}{0}\overset{z_2}{1}$. From the state $\overset{z_1}{0}\overset{z_2}{1}$ under the action of the input symbol $\overset{x}{0}$ we get into the same state $\overset{z_1}{0}\overset{z_2}{1}$ (Fig. 2). This means that, using a sequence of length one, at time t_1 we get into a state in which we can organize the arrival of the required test pair. Namely, under the action of the input signal $\overset{x}{0}$, we form the first vector of the test pair at time t_2. Being in the same state $\overset{z_1}{0}\overset{z_2}{1}$ at the next time t_3, we send the input signal $\overset{x}{1}$ and form the second vector of the test pair. This can be seen from the transition diagram (Fig. 2).

For a rising transition, we perform steps 1, 2 of the algorithm,

1. $D(R_{\text{rob}/x}) = xz_2$.
2. $D(R_{b/p}) = 0$.

Since $D(R_{b/p}) = 0$, we conclude that it is impossible to deliver a test pair for a rising transition for the path in question.

Case (b). Consider a test pair $(v_1^{\text{in}}, v_1^s; v_1^{\text{in}}, v_2^s)$ of vectors that must be fed to the combination component of the sequential circuit to detect a delay in a given path α under conditions of difference between the vectors of the test pair (components of v_1^s, v_2^s) in the state variable z_i, while the components in the input variables coincide. For simplicity, we assume that the variable marking the given path has no inversion. Let us describe the procedure for delivering a test pair at appropriate times:

1. Preliminarily, using a suitable input action, we set the circuit (at time t_1) to the state v_1^s. In this case, neither the input signal nor the state from which the transition to the state v_1^s is performed is of any significance.

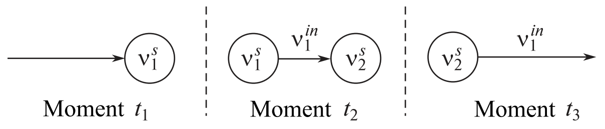

Fig. 4.

2. From the state v_1^s, under the action of the input signal v_1^{in} arriving at time t_2 (it is at this moment that the input of the combination component of the sequential circuit of the first vector v_1^{in}, v_1^s of the test pair is supplied), we go to the state v_2^s; i.e., we go to a neighboring state that differs from v_1^s in the variable z_i.
3. In the state v_2^s at time t_3, we supply the vector v_1^{in} to the circuit inputs; i.e., we ensure the arrival of the second vector v_1^{in}, v_2^s of the test pair at the inputs of the combination component. It does not matter what the next state of the circuit will be, it is only important that there should be a change in the signal at the output associated with the path under consideration.

At time t_4, we observe a delay fault at the output of the combination component corresponding to the path if a delay fault occurs.

Note that the neighboring vectors of the test pair arrive at consecutive times. The transitions at times t_1, t_2, and t_3 are illustrated in Fig. 4.

Thus, to deliver a test pair in all the situations noted above, it is necessary to get from the initial state q_0 to the state v_1^s. In this case, not all states with transitions to neighboring states are of interest but only those in which the transition is realized with respect to the input component v_1^{in} contained among the complete states represented by R_{rob}. Having provided the supply of the input component in this state, we then form the vector v_1^{in}, v_2^s by replacing the value of the variable z_i in v_1^s with the inverse one.

In the STG description, we isolate a fragment that represents transitions into neighboring states. The fragment consists of rows of the form

$$\begin{gathered} k_1 q_{i_1} \to q_{j_1} \\ k_2 q_{i_1} \to q_{j_1} \\ \dots \\ \dots \\ k_m q_{i_1} \to q_{j_1} \\ k_{m+1} q_{i_2} \to q_{j_2} \\ \dots \\ \dots \\ k_{m+s} q_{i_2} \to q_{j_2} \\ \dots \end{gathered}$$

Here the k_i are the conjunctions (ternary vectors) on the set of input variables of the scheme, and $q_{i_1}, q_{j_1}, q_{i_2}, q_{j_2} \dots$ are the states of the STG description (the states of a finite automaton) represented by Boolean vectors under encoding. The left-hand side of the considered fragment, i.e., the conjunctions of the input variables together with the states assigned to them, are represented in the form the ROBDD R_s. This graph will be useful when considering all paths from the presented set the beginnings of which are marked by the state variables of the sequential circuit. Let us pass to considering a given path α.

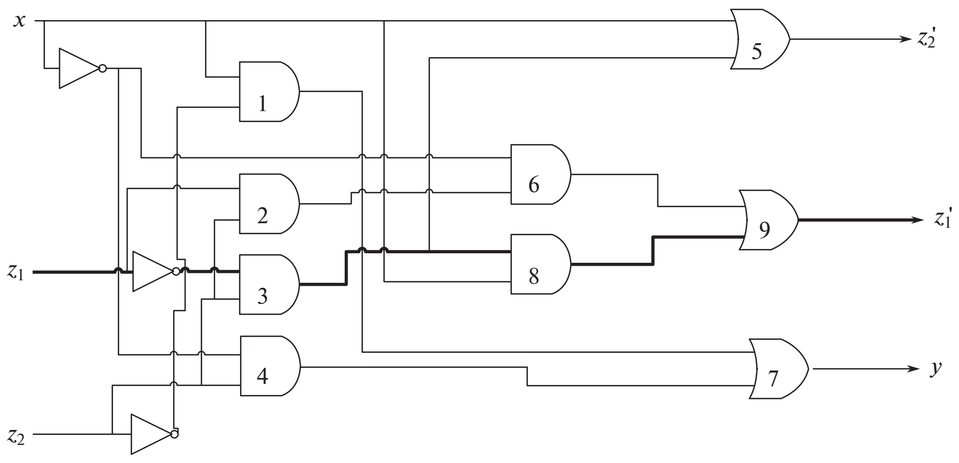

Fig. 5.

The sequence of steps for a falling transition of the path α:

1. From R_{rob}, we form a_p-test tuples (tuples v_1^{in}, v_1^s) of the test pair that are associated with the state variable z_i, representing them by an appropriate ROBDD graph $R_{\text{rob}/z_i} = R_{\text{rob}} \,\&\, z_i$.
2. Of the a_p-test tuples, we select those which generate transitions into the neighboring states in the variable z_i, $R_{a/s} = R_{rob/z_i} \,\&\, R_s$; i.e., we obtain the set of all vectors of the form v_1^{in}, v_1^s associated with the time t_2 when testing for a path delay.
3. In $R_{a/s}$, we isolate the set of internal states and represent it by a graph R^{s_0}.
4. Having obtained the vector v_1^{in}, v_1^s, we then form a vector v_1^{in}, v_2^s that is generated by R_{rob} and supply it at time t_3. The vector v_2^s is different from the vector v_1^s in the variable z_i.

When finding a test pair for a rising transition, we perform the following steps of the algorithm for the b_p-test tuples.

The sequence of steps for a rising transition of the path α:

1. From R_{rob}, we select b_p-test tuples (tuples v_1^{in}, v_1^s) of the test pair that are associated with the input letter $\overline{z}_i$, representing them by an appropriate ROBDD graph $R_{\text{rob}/\overline{z}_i} = R_{\text{rob}} \,\&\, \overline{z}_i$.
2. Of the b_p-test tuples, we select those which generate transitions into the neighboring state in the variable z_i, $R_{b/s} = R_{rob/\overline{z}_i} \,\&\, R_s$; i.e., we obtain the set of all vectors of the form v_1^{in}, v_1^s associated with time t_2 when testing for a path delay.
3. In $R_{b/s}$, we isolate the set of internal states and represent it by a graph R^{s_0}.
4. Having obtained the vector v_1^{in}, v_1^s, we then form a vector v_1^{in}, v_2^s that is generated by R_{rob} and supply it at time t_3. The vector v_2^s is different from the vector v_1^s in the variable z_i.

Thus, in all the above algorithms (step 3), a suitable set R^{s_0} is formed to find test pairs.

We find a sequence [15] that sets the circuit from the initial state q_0 to one of the states of the set R^{s_0} corresponding to the desired pair, i.e., to the state v_1^s. Having obtained this state, we use a ROBDD graph from the set $\{R_{a/p}, R_{a/s}, R_{b/p}, R_{b/s}\}$ for a test pair of the same type, and using this graph, we find the first vector of the test pair and then the second vector by the method specified in step 4 of the algorithm corresponding to the desired pair.

Consider an example. In the circuit, we are given the path $z_1 \to 3 \to 8 \to 9 \to z_1'$ (see Fig. 5).

The beginning of the path is marked by the variable z_1.

We calculate the Boolean difference for this path,

$$D_{U_9}/D_{U_8} = (U_8 \vee U_6)|_{U_8=0} \oplus (U_8 \vee U_6)|_{U_8=1} = U_6 \oplus 1 = \overline{U}_6;$$

$$D_{U_8}/D_{U_3} = (U_3x)|_{U_3=0} \oplus (U_3x)|_{U_3=1} = x;$$
$$D_{U_3}/D_{z_1} = (\overline{z}_1z_2)|_{z_1=0} \oplus (\overline{z}_1z_2)|_{z_1=1} = z_2;$$
$$D_{\text{path}} = (\overline{\overline{x}z_1z_2})xz_2 = (x \vee \overline{z}_1 \vee \overline{z}_2)xz_2 = xz_2.$$

Thus, the Boolean difference is representable in the form $D_{\text{path}} = xz_2$.

Further, we calculate $D(R_{\text{rob}})$,

$$D(R_{\text{rise}}) = x\overline{z}_1z_2;$$
$$D(R_{\text{fall}}) = xz_1z_2;$$
$$D(R'_{\text{rise}}) = xz_2;$$
$$D(R'_{\text{fall}}) = xz_2;$$
$$D(R_{\text{rob}}) = xz_2.$$

We have obtained a unique conjunction that does not contain the variable marking the beginning of the considered path. The conjunction generates a unique test pair. We have obtained $(\overset{x}{1}\overset{z_1}{1}\overset{z_2}{1}, \overset{x}{1}\overset{z_1}{0}\overset{z_2}{1})$.

From Table 3, we isolate transitions representable by neighboring vectors on the set of variables z_1, z_2. We set conditions ensuring these transitions in the form of the DNF

$$D(R_s) = x\overline{z}_1\overline{z}_2 \vee x\overline{z}_1z_2 \vee xz_1z_2 \vee \overline{x}z_1z_2 \vee \overline{x}z_1\overline{z}_2.$$

Consider a falling transition. Choose a vector $\overset{x}{1}\overset{z_1}{0}\overset{z_2}{1}$. This is a a_p-test tuple, since it turns the DNF of a single-output subcircuit in which the path in question lies into unity.

For a falling transition, we perform steps 1–3 of the algorithm to obtain the state $D(R^{s_0}) = \overset{z_1}{0}\overset{z_2}{1}$,

1. $D(R_{\text{rob}/\overline{z}_1}) = x\overline{z}_1z_2$.
2. $D(R_{a/s}) = x\overline{z}_1z_2$
3. $D(R^{s_0}) = \overline{z}_1z_2$.

To ensure the delivery of a test pair, we need to get from the initial state q_0 represented by the vector $\overset{z_1}{0}\overset{z_2}{0}$ into the state $\overset{z_1}{0}\overset{z_2}{1}$. It was shown earlier that it is possible to get into the state $\overset{z_1}{0}\overset{z_2}{1}$ from the initial state $\overset{z_1}{0}\overset{z_2}{0}$ based on the input symbol $\overset{x}{1}$ ($D(R^{s_1}) = xz_1 \vee x\overline{z}_2 \vee \overline{x}\overline{z}_1z_2$.). In the state $\overset{z_1}{0}\overset{z_2}{1}$, we form the test pairs $\overset{x}{1}\overset{z_1}{0}\overset{z_2}{1}$ and $\overset{x}{1}\overset{z_1}{1}\overset{z_2}{1}$. This situation can be observed in the transition diagram (Fig. 2).

Consider a rising transition. Choose a vector $\overset{x}{1}\overset{z_1}{1}\overset{z_2}{1}$. This is a b_p-test tuple, since it turns the DNF of a single-output subcircuit where the considered path lies into zero.

Let us try to find a sequence of length one by multiplying the DNFs corresponding to the state variables: $D(R^{s_1}) = x\overline{z}_1z_2$. The resulting DNF in the initial state $\overset{z_1}{0}\overset{z_2}{0}$ does not turn into unity. Consequently, there does not exist a sequence of length 1 delivering the desired test pair. Let us try to find a sequence of length two. This means that we need to find conditions for the transition from the initial state into the state $\overset{z_1}{0}\overset{z_2}{1}$. These conditions have already been found and represented in the form of a DNF. Under the action of the input symbol $x = 1$, we get from the initial state $\overset{z_1}{0}\overset{z_2}{0}$ into the state $\overset{z_1}{0}\overset{z_2}{1}$. Further, from the state $\overset{z_1}{0}\overset{z_2}{1}$, under the action of the input symbol $\overset{x}{1}$, we fall into the state $\overset{z_1}{1}\overset{z_2}{1}$. Thus, we have obtained an input sequence $\overset{x}{1}$, $\overset{x}{1}$ sending the circuit into the state $\overset{z_1}{1}\overset{z_2}{1}$. This can be seen from the transition diagram (Fig. 2). We form the first vector of the test pair $\overset{x}{1}\overset{z_1}{1}\overset{z_2}{1}$. Under the action of the input signal $x = 1$, we pass into the state $\overset{x}{1}\overset{z_1}{0}\overset{z_2}{1}$; i.e., we ensure supplying the test pair for a rising transition. This situation can also been seen from the transition diagram (Fig. 2).

Table 4. Results of experiments

name	i	s	xf	xr	zf	zr	xob	zob	ob	N	%
s27	4	3	8	8	18	2	8	2	10	58	17.2%
s208	11	8	0	0	0	3	0	0	0	643	0%
s298	3	14	40	0	0	0	0	0	0	1 179	0%
s386	7	6	45	57	320	572	0	160	160	1 351	11.8%
s510	19	6	0	0	906	724	0	575	575	1 373	41.9%
s820	18	5	34	451	1 860	1 392	0	1 313	1 313	3 055	42.87%
s832	18	5	34	450	1 859	1 391	0	1 312	1 312	3 052	42.98%
s1488	8	6	0	58	2 578	2 209	0	1 612	1 612	5 384	29.94%
s1494	8	6	0	58	2 628	2 247	0	1 626	1 626	5 351	30.39%

For each path from a given set and each type of signal value excursions, we construct the corresponding sequences that detect a path delay, combine them, and obtain a test sequence for a given set of paths. Note that the resulting test sequence consists of fragments starting in the state q_0.

5. RESULTS OF EXPERIMENTS

The algorithms described above were implemented as a program. The CUDD library was used for operations on ROBDD graphs. Experiments were carried out on some benchmarks, the results of which are compiled in Table 4. Only the longest paths were considered in the circuits presented in the benchmarks. The length of the homing sequence was limited to a thousand input vectors; however, during the experiment, the homing sequences did not reach even a hundred input vectors. The last column in the table shows the proportion (in percent) of paths for which a test pair was delivered from the initial state of the sequential circuit that detects a robustly testable path delay fault. This fraction is determined with respect to all paths for which there are test pairs for the same faults under the conditions of the availability of inputs mapped to the state variables of the combination component.

The results of experiments carried out on reference circuits showed that the fraction of paths for which it is possible to deliver test pairs that detect robustly testable path delay faults is at best forty-odd percent. Of the nine schemes, for two it was not possible to deliver test pairs for any of the considered paths at all. This means that it is impractical to abandon scanning technologies despite the associated high hardware costs.

The notation is as follows:

name	The name of the benchmark.
i	The number of inputs.
s	The number of state variables.
xr	The number of paths whose beginning is marked by an input variable and for which a test pair can be delivered from the initial state of the circuit. The test pair is intended for a rising transition.
xf	The number of paths whose beginning is marked by an input variable and for which a test pair can be delivered from the initial state of the circuit. The test pair is intended for a falling transition.

zr	The number of paths whose beginning is marked by a state variable and for which a test pair can be delivered from the initial state of the circuit. The test pair is intended for a rising transition.
zf	The number of paths whose beginning is marked by a state variable and for which a test pair can be delivered from the initial state of the circuit. The test pair is intended for a falling transition.
xob	The number of paths whose beginning is marked by an input variable and for which test pairs can be delivered for both excursions of signal values from the initial state of the circuit.
zob	The number of paths whose beginning is marked by a state variable and for which test pairs can be delivered for both excursions of signal values from the initial state of the circuit.
ob	The total number of robustly testable paths for each of which both signal excursions are detectable.
N	The number of paths for which there exist nonempty graphs R_{rob}.
%	The fraction (in percent) of robustly testable paths for which it has been possible to deliver a test pair among the paths for which there exist nonempty graphs R_{rob}.

6. CONCLUSIONS

Earlier, a method was proposed for finding all test pairs consisting of adjacent tuples of Boolean vectors that detect robustly testable path delay faults when they are delivered to the inputs of the combination component of a sequential circuit. In this paper, we propose a method for delivering such test pairs from the initial state q_0 of a sequential circuit under constraints on the sequence length based on operations over ROBDD graphs. The work is focused on investigating the possibility of abandoning the use of scanning techniques that involve large hardware costs. In the paper, the classes of automata and methods of coding states are isolated for which delivery of test pairs existing in the combination component is impossible. The experiments carried out on benchmarks have shown that it is far from being always possible to deliver at least one test pair, no matter which one, for a path that has test pairs for the combination component under the conditions of availability of its state variables. This means that for some sequential circuits, it may turn impossible to detect robustly testable path delay faults without considerable hardware costs.

APPENDIX

Proof of Theorem 1. We will bear in mind that, in the general case, the a_p-test tuple of the test pair for a robustly testable path delay fault turns several nonempty ENF conjunctions into unity (there are no mutually inverse variables in nonempty ENF conjunctions). Each of these conjunctions contains an ENF literal associated with the path α. (The ENF literal is a variable symbol with a suitable inversion sign and a sequence of indices representing the path α.) The variable associated with the literal is present only once in such a conjunction. This is because the b_p-test tuple of the (b_p, a_p) test pair is generated by a conjunction that does not contain literals that differ from the literal associated with path α only by a sequence of indices mapped to another path in the circuit. At the same time, both test tuples of the pair under consideration are generated by the same nonempty ENF conjunction (condition 4). Note that the test tuple for the a_p-fault (by construction) is orthogonal to each ENF conjunction that does not contain the literal associated with path α. The test tuple for a b_p-fault (by construction) is orthogonal to all ENF conjunctions. Let us take into account the fact that any test tuple is a Boolean vector, and therefore, it is orthogonal to the empty conjunction. This means that the minimum-cover interval u generated by the neighboring vectors (b_p, a_p) is orthogonal to each ENF conjunction that does not contain a literal associated with the path α. Thus, for the considered neighboring sets, the conditions of being a test pair for a robustly testable path delay fault listed earlier in the paper are satisfied. Bearing in mind Theorem 5 in the paper [7], we arrive at the conclusion that the test pair found can be used to detect opposite excursions in the signal values of the path under consideration. The proof of the theorem is complete. ■

Proof of Theorem 2. Let u be the minimum-cover interval of tuples (b_p, a_p) of a test pair for a robustly testable path delay fault. Consider the a_p-test tuple. This tuple (by construction) is orthogonal to each conjunction that does not contain a literal associated with the path α and possibly meets some other conjunctions of the considered ENF. Let us construct the set b' adjacent in the variable x_i for a_p. Here x_i is the variable marking the beginning of the path for which the test pair (b_p, a_p) was constructed. The set b' is absorbed by the interval u; therefore, it is also orthogonal to each conjunction that does not contain a literal associated with the path α. Moreover, b' is orthogonal to all conjunctions containing a literal associated with the path α with respect to the variable x_i; that is, b' is orthogonal to the conjunctions of the ENF as a whole. This means that it belongs to the domain of zero values of the function represented by the ENF and is a b_p-test tuple. The proof of the theorem is complete. ■

REFERENCES

1. Matrosova, A., Ostanin, S., and Chernyshov, S., Masking robust testable PDFs, *Proc. 2019 IEEE East-West Des. & Test Symp. (EWDTS)* (Batumi, September 13–16, 2019), Kharkov: IEEE, 2019, pp. 420–423.

2. Matrosova, A.Yu., Andreeva, V.V., Tychinskiy, V.Z., and Goshin, G.G., Applying ROBDDs for delay testing of logical circuits, *Russ. Phys. J.*, 2019, vol. 62, no. 5, pp. 615–621.

3. Lindgren, P., Kerttu, M., Thornton, M., and Drechsler, R., Low power optimization technique for BDD mapped circuits, *Proc. 2001 Asia South Pac. Des. Autom. Conf. (ASP-DAC '01). Assoc. Comput. Mach.* (New York), pp. 615–621.

4. Shelar, R.S. and Sapatnekar, S.S., An efficient algorithm for low power pass transistor logic synthesis, *Proc. ASP-DAC/VLSI Design 2002. 7th Asia South Pac. Des. Autom. Conf. and 15h Int. Conf. VLSI Des.* (Bangalore, India, 2002), pp. 87–92.

5. Gekas, G., Nikolos, D., Kalligeros, E., and Kavousianos, X., Power aware test-data compression for scan-based testing, *ICECS 2005. 12th IEEE Int. Conf. Electron. Circuits Syst.* (Gammarth, Tunisia, 2005), pp. 1–4.

6. Tudu, J.T., Larsson, E., Singh, V., and Agrawal, V.D., On minimization of peak power for scan circuit during test, *Test Symp. 2009 14th IEEE Eur.* (2009), pp. 25–30.

7. Kotasek, Z., Skarvada, J., and Strnadel, J., Reduction of power dissipation through parallel optimization of test vector and scan register sequences, *IEEE Int. Symp. Des. Diagn. Electron. Circuits Syst.* (2010), pp. 364–369.

8. Eichelberger, E.B. and Williams, T.W., A logic design structure for LSI, *Proc. 14th Des. Autom. Conf.* (1977), pp. 462–468.

9. Agrawal, V.D., Cheng, K.T., and Johnson, D.D., Designing circuits with partial scan, *IEEE Des. Test Comput.* (1988), pp. 8–15.

10. Matrosova, A.Yu. and Lipskii, V.B., Properties of pairs of test vectors detecting path delay faults in high performance VLSI logical circuits, *Autom. Remote Control*, 2015, no. 4, pp. 135–148.

11. Sapozhnikov, V.V., Sapozhnikov, Vl.V., and Lykov, A.A., Analysis theorems for time-delay fault detection, *Elektron. Model.*, 2004, vol. 26б, no. 3, pp. 83–93.

12. Matrosova, A.Yu., Andreeva, V.V., and Nikolaeva, E.A., Finding test pairs for PDFs in logic circuits based on using operations on ROBDDs, *Russ. Phys. J.*, 2018, vol. 61, no. 5, pp. 994–999.

13. Efanov, D., Sapozhnikov, V., Sapozhnikov, Vl., and Nikitin, D., Sum code formation with minimum total number of undetectable errors in data vectors, *Proc. 13th IEEE East-West Des. & Test Symp. (EWDTS'2015)* (Batumi, Georgia, September 26–29, 2015), pp. 141–148.

14. Efanov, D.V., Sapozhnikov, V., and Sapozhnikov, Vl., Two-modulus codes with summation of one-data bits for technical diagnostics of discrete systems, *Autom. Control Comput. Sci.*, 2018, vol. 52, no. 1, pp. 5–21.

15. Matrosova, A., Andreeva, V., and Melnikov, A., ROBDDs application for finding the shortest transfer sequence of sequential circuit or only revealing existence of this sequence without deriving the sequence itself, *Proc. IEEE East-West Des. & Test Symp. (EWDTS'2016)*, Yerevan: IEEE Comput. Soc., 2016, pp. 513–516.

This paper was recommended for publication by M.F. Karavai, a member of the Editorial Board