2021 International Siberian Conference on Control and Communications (SIBCON)

BDD and DNF Based Algorithms for Constructing All Testability Functions of Combinational Circuit

Olga Golubeva
Tomsk State University (TSU)
Tomsk, Russian Federation

***Abstract*—Constructing testability functions of a combinational circuit line, such as: the controllability, observability and stuck-at fault detection functions, as well as the complement of the observability function is considered. Methods and algorithms for constructing testability functions based on Binary Decision Diagram (BDD) and Disjunctive Normal Form (DNF), as well as methods for constructing Conjunctive Normal Form (CNF) and obtaining testability functions using a SAT solver are proposed. Methods and algorithms for constructing testability functions for all and a subset of lines of a circuit are also proposed. Proposed methods and algorithms make it possible to significantly reduce the computational costs for constructing testability functions of a combinational circuit.**

Keywords—testability functions, stuck-at fauls, observability, Boolean difference, controllability, fault detection, BDD, DNF, CNF, SAT solver, combinational circuit

I. INTRODUCTION

Constructing Boolean testability functions of a line of a combinational circuit is considered, namely: the controllability, observability and stuck-at fault detection functions, as well as the complement (inverse) of the observability function.

Testability functions find their application in numerous tasks of analysis and synthesis of logical circuits [1 – 12], such as: identification of hard-to-detect stuck-at faults in combinational circuits and generation tests for stuck-at faults [1 – 3], detection of Trojan circuits [4], optimization of circuits [5, 6], fault tolerance [7] and calculation of testability measures [1, 3, 8].

In this paper methods and algorithms for constructing the considered functions for BDD [13] and DNF representations of functions are proposed, in which the construction process is considered in detail, showing the possibilities of efficient computations. Methods for efficient construction of CNF and obtaining testability functions using a SAT solver are proposed as well. Methods are a detailing and development of previously proposed methods [3, 7, 14, 15]. Methods and algorithms for obtaining functions for all or part of lines of a circuit are also proposed, which makes it possible to reduce the amount of computations.

Section II considers testability functions, their properties and the construction of testability functions in general. Section III describes in detail the construction of BDDs and DNFs of testability functions and presents the construction algorithms. Section IV proposes methods for constructing CNF and obtaining testability functions using a SAT solver. Section V presents the proposed methods and algorithms for constructing testability functions for all and a subset of lines of a circuit. Section VI is a conclusion.

II. TESTABILITY FUNCTIONS, THEIR PROPERTIES AND CONSTRUCTION

Consider a combinational circuit S with n inputs and m outputs. X, $X = \{x_1, ..., x_n\}$, is the set of input variables of the circuit. $\varphi_i(X)$, $i = \overline{1,m}$, is a Boolean function implemented by the i^{th} output of the fault-free circuit S.

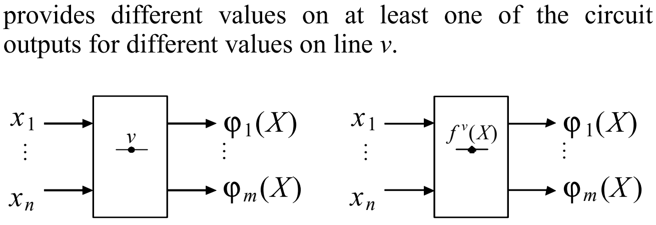
Consider a line v of the circuit S (Fig. 1a). $S^{v,1}$ is the circuit S with a stuck-at-1 fault, and $S^{v,0}$ is S with a stuck-at-0 fault on line v. $\varphi_i^{v,1}(X)$ and $\varphi_i^{v,0}(X)$, $i = \overline{1,m}$, are Boolean functions implemented by the i^{th} outputs of the circuits $S^{v,1}$ and $S^{v,0}$, respectively (Fig. 2). Note that circuits $S^{v,1}$ and $S^{v,0}$ are sub-circuits of S, which can be simplified after substituting constants 1 and 0 into line v, respectively. In these circuits, elements that are not connected with circuit outputs can be removed.

Consider the *testability functions* of line v, such as the stuck-at fault detection functions, the observability and controllability functions.

*The stuck-at-*1 (*stuck-at-*0) *fault detection function* on line v of the circuit is a Boolean function $D^{v,1}(X)$ ($D^{v,0}(X)$) such that $D^{v,1}(\alpha) = 1$ ($D^{v,0}(\alpha) = 1$) iff α is a test for the stuck-at-1 (stuck-at-0) fault on line v of the circuit.

$D^{v,1}(X)$ and $D^{v,0}(X)$ represent all test vectors for single stuck-at-1 and stuck-at-0 faults on line v, respectively.

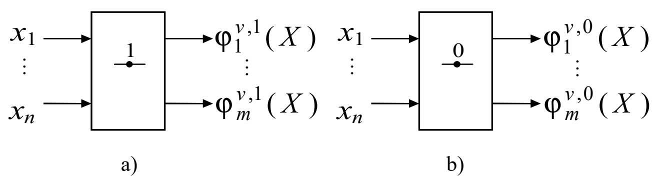
The observability function of line v is a Boolean function $B^v(X)$ such that $B^v(\alpha) = 1$ iff α is an input vector that provides different values on at least one of the circuit outputs for different values on line v.



a) b)

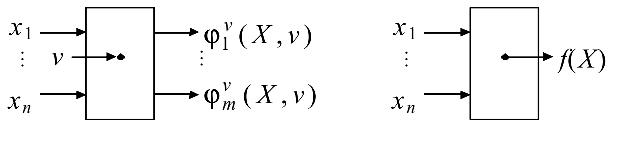
Fig. 1. a) A combinational circuit S and its line v; b) S and the function $f^v(X)$

978-1-7281-8504-0/21/$31.00 ©2021 IEEE



a) b)

Fig. 2. a) $S^{v,1}$ – the circuit S with the stuck-at-1 fault on line v; b) $S^{v,0}$ – the circuit S with the stuck-at-0 fault on line v



a) b)

Fig. 3. a) the circuit $S^{v,B}$, where the line v is associated with the input variable v; b) the circuit $S^{v,C}$, where the line v is associated with an output

Let $B_i^v(X)$ be the observability function of line v at the i^{th} output of the circuit. Formulas for the observability functions of line v are:

$$B_i^v(X) = \varphi_i^{v,1}(X) \oplus \varphi_i^{v,0}(X), \quad i = \overline{1,m}, \tag{1}$$

$$B^v(X) = B_1^v \vee \ldots \vee B_m^v . \tag{2}$$

The observability function $B_i^v(X)$ represents all input vectors that provide different values at the i^{th} output of the circuit for different values on line v.

$\overline{B}^v(X)$ (complement of $B^v(X)$) represents don't-care values for line v, i.e. all input vectors of the circuit on which the value on line v does not affect the values at the outputs of the circuit.

The 1-*controllability* (0-*controllability*) *function* $C^{v,1}(X)$ ($C^{v,0}(X)$) of line v represents input vectors providing the value 1 (0) on line v. The 1-*controllability function* is implemented by the output of the sub-circuit corresponding to line v; its inversion is the 0-*controllability function*.

Denote as $f^v(X)$ a Boolean function implemented on line v of the circuit (Fig. 1(b)). Then

$$C^{v,1}(X) = f^v(X), \; C^{v,0}(X) = \overline{f}^v(X). \tag{3}$$

Let $\varphi_i^v(X,v)$, $i = \overline{1,m}$, be a function implemented at the i^{th} output of the circuit $S^{v,B}$, in which the variable v is the input variable, associated with line v of the circuit (Fig. 3(a)). The inputs of elements connected to line v in the circuit S, are connected to the input v in the circuit $S^{v,B}$. Note that in the circuit $S^{v,B}$ all elements that are not connected with circuit outputs can be removed. The functions $\varphi_i^v(X,v)$ in some representation can be obtained from the structural description of the circuit $S^{v,B}$.

Under removing elements from the circuit, we will also mean removing connections that have at least one of the two poles removed, as well as the inputs and outputs of the circuit, if all the elements to which they are connected are removed.

Functions $\varphi_i(X)$, $\varphi_i^{v,1}(X)$ and $\varphi_i^{v,0}(X)$ can be obtained from the function $\varphi_i^v(X,v)$ as follows: $\varphi_i(X) = \varphi_i^v(X, f^v(X))$, $\varphi_i^{v,1}(X) = \varphi_i^v(X,1)$, $\varphi_i^{v,0}(X) = \varphi_i^v(X,0)$.

Thus, two ways of constructing B^v by (1) and (2) were considered. In these two ways, functions $\varphi_i^{v,1}(X)$ and $\varphi_i^{v,0}(X)$ are obtained as follows. 1) From the function $\varphi_i^v(X,v)$: $\varphi_i^{v,1}(X) = \varphi_i^v(X,1)$ and $\varphi_i^{v,0}(X) = \varphi_i^v(X,0)$; 2) $\varphi_i^{v,1}(X)$ and $\varphi_i^{v,0}(X)$ are obtained in some representation from the structural description of circuits $S^{v,1}$ and $S^{v,0}$.

$B_i^v(X)$ is the Boolean difference of the function $\varphi_i^v(X,v)$ with respect to the variable v. $B^v(X) = \mathop{\vee}\limits_{i=1}^{m} B_i^v(X) = \mathop{\vee}\limits_{i=1}^{m} \dfrac{\partial \varphi_i^v(X,v)}{\partial v}$.

The stuck-at fault detection functions $D^{v,1}(X)$ and $D^{v,0}(X)$ can be obtained using the observability and controllability functions as follows:

$$\begin{aligned} D^{v,1}(X) &= B^v(X) \cdot C^{v,0}(X), \\ D^{v,0}(X) &= B^v(X) \cdot C^{v,1}(X). \end{aligned} \tag{4}$$

Let $N = \{e_1, e_2, \ldots, e_t\}$ denote the set of all elements of the circuit S.

Let's define the relation "$\prec$" between circuit elements e_i and e_j. $e_i \prec e_j$, if: a) the output of e_i is connected to the input of e_j or b) the circuit contains a subset of elements $e_{k_1}, \ldots, e_{k_p}$, such that the following relations hold $e_i \prec e_{k_1} \prec \ldots \prec e_{k_p} \prec e_j$. In other words, $e_i \prec e_j$, if there is a path in the circuit from the output of e_i to the input of e_j.

Let's call an element e_i *the direct predecessor* of an element e_j, and e_j – *the direct successor* of e_i, if the output of e_i is connected to an input of e_j. An element e_i is called the *predecessor* of an element e_j, and e_j is called the *successor* of e_i, if $e_i \prec e_j$.

Similarly, the relation "$\prec$" between lines, inputs and outputs of the circuit can be defined.

Let *on*(g) be the on-set of a Boolean function $g(x_1, \ldots, x_n)$, $on(g) = \{(a_1, \ldots, a_n) \in \mathbf{B}^n\!: g(a_1, \ldots, a_n) = 1\}$, and *off*($g$) be the off-set of $g(x_1, \ldots, x_n)$, $off(g) = \{(a_1, \ldots, a_n) \in \mathbf{B}^n\!: g(a_1, \ldots, a_n) = 0\}$.

In [7], we proposed the method for constructing fault detection functions based on constructing the observability function and binary simulation of the sub-circuit $S^{v,C}$ (Fig. 3(b)) using its structural description. Note that the circuit $S^{v,C}$ contains only elements connected with its single output.

The method uses a formula that follows from (3) and (4):

$$B^v(X) = D^{v,1}(X) \vee D^{v,0}(X) . \qquad (5)$$

In this method, the set $on(B^v)$ is obtained first. It can be obtained in different ways. Then sets $on(D^{v,1})$ and $on(D^{v,0})$ are obtained by binary simulation of the sub-circuit $S^{v,C}$ on vectors from $on(B^v)$. Simulation is performed on each vector from the set $on(B^v)$. If on some vector $\alpha \in on(B^v)$ on line v of the circuit the value 0 is obtained, then α is included in $on(D^{v,1})$, otherwise – in $on(D^{v,0})$.

Note that this method can find not only test vectors, but also values on each predecessor of line v for each test vector. The values on the predecessors can be obtained as the projection of the vector of values on each circuit line, obtained by binary simulation, onto the considered subset of variables.

Example

Consider a circuit $S^{v,C}$ with three input variables $X = \{x_1, x_2, x_3\}$. w_1, w_2, w_3, w_4 are lines that are predecessors of v. Let's $on(B^v) = \{(010),(100),(101)\}$.

Let the following values be obtained on lines of the circuit $S^{v,C}$ by binary simulation.

	x_1	x_2	x_3	w_1	w_2	w_3	w_4	v
1)	0	1	0	1	0	0	1	1
2)	1	0	0	1	0	1	1	1
3)	1	0	1	0	0	1	1	0

Then input vectors (010) and (100) are tests for the stuck-at-0 fault, as the value of the variable v is equal to 1; (101) is a test for the stuck-at-1 fault, as $v = 0$.

On the stuck-at-0 fault test vector (010) internal lines $\{w_1, w_2, w_3, w_4\}$ take on values (1001), and on the test vector (100) they take on values (1011). Whereas, if to consider the subset of lines $\{w_1, w_2\}$, it takes only one value (10) on all tests for stuck-at-0 fault.

To line v of the circuit S corresponds an incompletely specified function that takes values on the set {0, 1, *}. $D^{v,1}(X)$ and $D^{v,0}(X)$ represent the on-set and off-set of the incompletely specified function. $\overline{B}^v(X)$ represents don't care set of the incompletely specified function.

Constructing testability functions based on (1) – (4) that used DNFs, BDDs or SAT solvers was previously considered in our papers [3, 7, 14, 15].

In this paper methods are proposed to reduce the computational costs of constructing testability functions. These methods are based on our previous research. The process of constructing functions is considered in detail, showing the possibilities of efficient calculations.

Next, consider in detail the construction of testability functions using BDDs and DNFs. Also consider the construction of CNF and obtaining testability functions using SAT solvers.

III. Constructing Testability Functions in BDD and DNF

Further, the local Boolean function of the element will also be called simply the function of the element, and the global function of the element will also be called the function implemented at the output of the element.

As it is known, BDDs (ROBDDs) of Boolean functions implemented by a combinational circuit can be obtained from the structural description of the circuit as follows [16]. First, the BDDs of all input variables of the circuit are constructed. Then, by levels, from the circuit inputs to the outputs, the BDDs of all intermediate subfunctions are constructed. When constructing the BDD of a circuit element, the local Boolean function implemented by that element is applied to the BDDs at inputs of the element. BDDs of elements, the outputs of which are outputs of the circuit, are Boolean functions implemented by the circuit.

Similarly, DNFs (ODNFs) of Boolean functions implemented by a combinational circuit can be obtained from the structural description of the circuit. When constructing DNFs, in addition to obtaining DNFs of functions implemented at outputs of circuit elements, it is necessary to obtain DNFs of inversions of these functions. When constructing the DNF of a circuit element, DNFs of the local Boolean function implemented by this element and DNF of its inversion are obtained, and then DNFs of functions implemented at the inputs of the element and their inversions are substituted in them.

BDDs (ROBDDs) and DNFs (ODNFs) of functions implemented by a circuit can also be obtained from the structural description of the circuit by considering elements in such an order that all predecessors of an element are considered before the element [14] (it is not necessary to consider elements exactly by levels). Let's order the circuit elements in a sequence $e_{k_1}, e_{k_2}, \ldots, e_{k_t}$ so that all relations "$\prec$" between the elements are fulfilled in it, i.e. if $e_{k_i} \prec e_{k_j}$, then e_{k_i} precedes e_{k_j} in the sequence. Then, moving through the elements from the first to the last in the obtained sequence and applying the local functions of the elements, to the BDDs or DNFs of functions implemented at inputs of elements, obtain BDDs or DNFs of functions implemented at the output of each element and at outputs of the circuit.

Above, two ways to obtain the functions $\varphi_i^{v,1}(X)$ and $\varphi_i^{v,0}(X)$, $i = \overline{1,m}$, in some representation were shown. Next, consider Algorithm 1 for obtaining testability functions of line v of the circuit S in the BDD (ROBDD) or DNF (ODNF) for the two considered ways of obtaining functions $\varphi_i^{v,1}(X)$ and $\varphi_i^{v,0}(X)$. Some steps of the algorithm are common for these two ways, while others are different, in the latter case the two ways are separated by sub-steps A) and B).

This algorithm is based on the method considered in [14] and described above for obtaining BDDs and DNFs of functions implemented by the circuit, as well as on the algorithms from [15]; a more general algorithm than previously is proposed in this paper.

Let O^v denote the subset of numbers of outputs that are successors of line v. Note that to obtain all testability functions, it is necessary to consider only a sub-circuit of the circuit S, which implements outputs that are successors of line v, since $B_i^v(X) = 0$ for $i \notin O^v$.

In the algorithm assume for brevity that v is the line connected to the output pole of an element. The algorithm can be easily extended to cases of other circuit lines, circuit inputs and outputs.

In Algorithm 1 (case B) only BDDs and DNFs of elements that are successors of line v in circuits $S^{v,1}$ and $S^{v,0}$ are obtained separately for the stuck-at-1 and stuck-at-0 faults.

Algorithm 1. Obtaining testability functions: $C^{v,1}$, $C^{v,0}$, B_i^v, B^v, $D^{v,1}$, $D^{v,0}$, and $\overline{B}^v$ in BDD or DNF.

1) Denote by S^v the sub-circuit of the circuit S, consisting of elements, connections between them and primary inputs and outputs connected with them, which are predecessors of outputs of the set O^v.

2) Obtain the BDD or DNF of the input variables of the circuit S^v.

3) Obtain the functions (in BDD or DNF), implemented at the outputs of elements of the circuit S^v.

Each element is considered after all its predecessors. The BDD or DNF of the function implemented at the output of an element is obtained by applying the local function of that element to the BDDs or DNFs of the functions implemented at inputs of the element. If the element is not a successor of v, obtain (in the BDD or DNF) a function implemented at its output. For elements that are successors of v, obtain functions as follows.

A) Obtain functions (in BDD or DNF), implemented at outputs of elements that are successors of line v. In this case, if one of the inputs of the element in the circuit S is connected to line v, then associate this input with the BDD or DNF of the input variable v.

B) For each element that is a successor of v, obtain two functions (in BDD or DNF): one for the value 1, the second for the value 0 on line v. Before obtaining the function at the output of an element, simplify the local function of the element, if possible, by substituting into it the corresponding constants from the inputs.

When constructing the DNF, in addition to obtaining DNFs of functions implemented at outputs of elements, also obtain DNFs of inversions of these functions.

4)

A) At the outputs of circuit elements, which are circuit outputs, functions $\varphi_i^v(X, v)$ in BDD or DNF are obtained for outputs from the set O^v. Exclude from consideration functions that do not depend on the variable v, since for them $B_i^v(X) = 0$. For each of the remaining functions of the set O^v, obtain two functions (in BDD or DNF): $\varphi_i^{v,1}(X) = \varphi_i^v(X,1)$ and $\varphi_i^{v,0}(X) = \varphi_i^v(X,0)$.

B) At the output of each circuit element, which is the circuit output, two functions are obtained (in BDD or DNF), which represent functions $\varphi_i^{v,1}(X)$ and $\varphi_i^{v,0}(X)$, $i \in O^v$.

5) Perform XOR of the obtained functions $\varphi_i^{v,1}(X)$ and $\varphi_i^{v,0}(X)$ according to (1).

6) Perform the disjunction of functions obtained as a result of XOR at step 5, according to (2); if the i-th output does not belong to the set O^v, then we skip it, since $B_i^v(X) = 0$. As a result, obtain the observability function B^v of line v of the circuit in BDD or DNF.

7) Obtain fault detection functions $D^{v,0}$ and $D^{v,1}$ for line v of the circuit in BDD or DNF according to (4). For this, perform the conjunction of B^v with the 1-controllability function $C^{v,1}$ and the 0-controllability function $C^{v,0}$, respectively. $C^{v,1}$ is implemented on line v and is obtained at step 3 of this algorithm. BDD $C^{v,0}$ is obtained from BDD $C^{v,1}$ by inverting values of terminal vertices.

If for line v it is necessary to obtain only the observability functions and (or) function $\overline{B}^v$, step 1 of Algorithm 1 is replaced by the following.

1) Denote by $S^{v,B}$ the sub-circuit of the circuit S, in which the variable v is the input variable associated with line v of the circuit S (Fig. 3(a)). The inputs of elements connected to line v in the circuit S are connected to the input v in the circuit $S^{v,B}$. $S^{v,B}$ consists of elements of the circuit S, connections between them and primary inputs and outputs connected with them, which are successors of input v or predecessors of these successors.

Next, steps 2 – 6 of the Algorithm 1 are performed for the circuit $S^{v,B}$.

IV. CONSTRUCTING CNF FOR OBTAINING TESTABILITY FUNCTIONS USING SAT SOLVERS

The proposed Algorithm 1 for obtaining testability functions in BDD or DNF can also be used with the necessary changes to construct the CNF for obtaining testability functions using a SAT solver. In this section methods for constructing CNFs and obtaining testability functions using a SAT solver are proposed.

It is known that the CNF for obtaining functions implemented by a combinational circuit using a SAT solver can be constructed as follows [17]: its own variable v_i is assigned to the output of each element e_i; then the CNF of the function $v_i \sim f_i$ is constructed, where f_i is the local function of the element e_i; CNFs obtained for the elements are united by the conjunction ($\wedge$) operation. Elements of the circuit can be considered in any order.

Thus, in order to obtain testability functions of a combinational circuit line, construct the CNF using an algorithm similar to Algorithm 1 described above, but instead of obtaining functions implemented at the outputs of elements by the superposition of functions at inputs of elements, we obtain CNFs of elements of the circuit under consideration and perform their conjunction. For some formula $A = F$, we obtain the CNF for the function $A \sim F$. CNFs for formulas are also united by the conjunction with the CNF under construction. For each element e_i that is a successor of line v, CNFs for two functions are constructed:

one is $v_{i,1} \sim f_i$ for the value 1 on line v, the other is $v_{i,0} \sim f_i$ for the value 0 on line v.

Note that in this method, we used both the circuit and formulas to construct the CNF for obtaining testability functions using a SAT solver.

The algorithm for constructing the CNF $P^{v,B,1}$ for obtaining only the observability function is presented in detail in [15].

To construct the CNF P^v for obtaining all testability functions of line v, steps 1 – 6 of Algorithm 1 are performed with changes described above. Step 7 is not performed and the variable B^v is not assigned the value 1. The CNF P^v is constructed using the circuit S^v described at step 1 of the Algorithm 1 and formulas (1) and (2).

Note that the variable B^v is assigned the value 1, if it is not necessary to obtain the on-set of the function $\overline{B}^v$.

Vectors from on-sets of fault detection functions are obtained from the satisfying assignments obtained for the CNF P^v by a SAT solver in a way similar to that described above for the method using binary simulation. However, here, by the satisfying assignment, we also determine whether the input vector belongs to the on-sets of each of testability functions of line v, based on the values of variables v and B^v of the satisfying assignment.

Let $\alpha|_X$ denotes the projection of vector α on the set of input variables X.

Consider a satisfying assignment α. Denote by X^v input variables of the circuit S^v. Then the belonging of $\alpha|_{X^v}$ to on-sets of different testability functions is determined as follows:

1) if $\alpha|_v = 1$, then $\alpha|_{X^v} \in on(C^{v,1})$;

2) if $\alpha|_v = 0$, then $\alpha|_{X^v} \in on(C^{v,0})$;

3) if $\alpha|_{B_i^v} = 1$, $i \in O^v$, then $\alpha|_{X^v} \in on(B_i^v)$;

4) if $\alpha|_{B_i^v} = 0$, $i \in O^v$, then $\alpha|_{X^v} \in on(\overline{B}_i^v)$;

5) if $\alpha|_{B^v} = 1$, then $\alpha|_{X^v} \in on(B^v)$;

6) if $\alpha|_{B^v} = 0$, then $\alpha|_{X^v} \in on(\overline{B}^v)$;

7) if $\alpha|_{B^v} = 1$ and $\alpha|_v = 1$, then $\alpha|_{X^v} \in on(D^{v,0})$;

8) if $\alpha|_{B^v} = 1$ and $\alpha|_v = 0$, then $\alpha|_{X^v} \in on(D^{v,1})$.

Note that if the set $X^v \subset X$, where X is the set of input variables of the circuit S, then variables $X \setminus X^v$ of the obtained on-sets of testability functions are set to '-' and cubes from on-sets of corresponding functions are obtained.

Values on predecessors as well as on successors of line v for test vectors from $on(D^{v,1})$ and $on(D^{v,0})$ are obtained from satisfying assignments in a way similar to that described above for the fault detection functions construction method using binary simulation.

Example

Consider a circuit S^v with three input variables $X^v = \{x_1, x_2, x_3\}$. For simplicity, suppose $X = X^v$. $\{w_1, w_2, ..., w_k\}$ are lines that are predecessors of v. Let a SAT solver be used to obtain satisfying assignments, and three of the obtained satisfying assignments are as follows.

	x_1	x_2	x_3	…	w_1	w_2	...	w_k	v	…	B^v
α_1:	0	1	0	…	1	1	…	0	1	…	0
α_2:	1	0	0	…	1	0	…	1	1	…	1
α_3:	1	0	1	…	0	0	…	1	0	…	1

Consider the input vector $\alpha_1|_{X^v} = (010)$. $(010) \in on(C^{v,1})$ since $\alpha_1|_v = 1$; $(010) \in on(\overline{B}^v)$ because $\alpha_1|_{B^v} = 0$. Since $(010) \in on(\overline{B}^v)$, (010) is not a test vector.

The input vector $\alpha_2|_{X^v} = (100) \in on(C^{v,1})$ since $\alpha_2|_v = 1$ and $(100) \in on(B^v)$ since $\alpha_2|_{B^v} = 1$, consequently (100) is a test vector for the stuck-at-0 fault. On the test vector (100) for the stuck-at-0 fault, internal variables of the circuit w_1, w_2, .. w_k take values $\alpha_2|_{w_1 w_2 ... w_k} = (10…1)$.

The input vectors $\alpha_3|_{X^v} = (101) \in on(C^{v,0})$ since $\alpha_3|_v = 0$ and $(101) \in on(B^v)$ since $\alpha_3|_{B^v} = 1$, consequently (101) is a test vector for the stuck-at-1 fault. On the test vector (101) for the stuck-at-1 fault, lines of the circuit w_1, w_2, .. w_k take values $\alpha_3|_{w_1 w_2 ... w_k} = (00…1)$.

In the same way, values on successors of v, as well as on any lines of the circuit, can be obtained from satisfying assignments.

From all satisfying assignments obtained for CNF using a SAT solver, on-sets of all testability functions can be obtained.

The proposed method makes it possible to obtain using a SAT solver all testability functions using only one CNF, the size of which is comparable to the size of the CNF of the circuit S. In this paper, the proposed method is considered in general. We plan to consider it in detail in future work.

Note that on-sets of testability functions can be obtained in a similar way using only binary simulation of the circuit S^v. For this, during the binary simulation, obtain two values on successors of line v: one for the value 1, the second for the value 0 on line v. Then, the obtained values are substituted into (1) and (2). Thus, we obtain a vector α of values on all lines of the circuit and values of observability functions. On-sets of testability functions are obtained in the similar way as for satisfying assignments for CNF P^v.

V. CONSTRUCTING ALL TESTABILITY FUNCTIONS OF A COMBINATIONAL CIRCUIT

Further, consider an algorithm for constructing testability functions for each line of the circuit S in BDD (ROBDD) or DNF (ODNF).

Algorithm 2. Constructing testability functions for all lines of a circuit in BDD or DNF.

1. Obtain functions (in BDD or DNF), implemented at the output of each element of a fault-free circuit S, in the way described above.

2. Construct testability functions for each line v_i as follows. For each line v_i, obtain BDD or DNF only for successors of line v_i (their obtaining was considered at step 3 of Algorithm 1); for all the rest lines use BDDs or DNFs obtained for the fault-free circuit. Then perform operations according to (1) – (4) and thus construct testability functions of line v_i.

Constructing BDDs or DNFs only for successors of each line can significantly reduce the computational cost when constructing testability functions for all lines of a circuit.

If only a subset of circuit lines is considered and it is necessary to obtain testability functions for this subset, step 1 in Algorithm 2 is replaced by the following one.

1) Consider a circuit consisting of elements of the circuit S, connections between them and primary inputs and outputs connected to them, which are predecessors of outputs that are successors of at least one of the lines under consideration. Obtain functions (in BDD or DNF), implemented at the output of each element of this circuit, as described above.

Then, for the obtained circuit, step 2 of Algorithm 2 is performed, where lines v_i from the given subset are considered.

Similarly, CNFs can be constructed to obtain testability functions using a SAT solver for all or part of lines of a circuit. First, CNFs are constructed for elements of a fault-free circuit S and functions $B_i^v \sim (v_{i,1} \oplus v_{i,0})$, $i = \overline{1,m}$, and $B^v \sim \bigvee_{i=1}^{m} B_i^v$ in the way described above; here v_i are variables corresponding to outputs of elements connected to outputs of the circuit. Then, for each line for which testability functions are constructed, these CNFs are used for elements that are not successors of line v, and for successors of line v, these CNFs are duplicated and adjusted according to the above. CNFs for functions $B_i^v \sim (v_{i,1} \oplus v_{i,0})$, $i \in O^v$, and $B^v \sim \bigvee_{i \in O^v} B_i^v$ are united by the conjunction with the CNF under construction.

VI. CONCLUSION

Efficient methods and algorithms for constructing testability functions and the complement of the observability function of a combinational circuit line based on BDD and DNF are proposed. Efficient methods for constructing CNF and obtaining testability functions using a SAT solver are also proposed. They are a detailing and development of previously proposed methods. Methods for constructing testability functions for all and for a subset of lines of a circuit are also proposed, which significantly reduces computational costs.

REFERENCES

[1] P. Bardell, W. McAnney, and J. Savir, Buid-In Test for VLSI: Pseudorandom Techniques. John Wiley & Sons, 1987.

[2] F. F. Sellers, M. Y. Hsiao, and L. W. Bearnson, "Analyzing errors with the boolean difference", *IEEE Trans. on Computers*, vol. C-17, no. 7, pp. 676–683, July 1968.

[3] O. Golubeva, "Detection of Hard-to-Detect Stuck-at Faults and Generation of their Tests Based on Testability Functions", *2018 IEEE Intern. Conf. on Automation, Quality and Testing, Robotics* (*AQTR*), May, 2018.

[4] M. Priyadharshini and P. Saravanan, "An Efficient Hardware Trojan Detection Approach adopting Testability based Features", *in Proc. of the 2020 IEEE International Test Conference India* (*ITC India*), July 2020.

[5] A. Mishchenko and R.K.Brayton, "SAT-Based Complete Don't-Care Computation for Network Optimization", *in Proc. of the Design, Automation and Test in Europe Conference and Exhibition* (DATE'05), March 2005, pp. 412–417.

[6] Yu. Matsunaga and M. Fujita, "Multi-Level Logic Optimization Using Binary Decision Diagrams", *IEEE International conference on computer-aided design* (ICCAD-89), 1989, pp. 556–559.

[7] O. Golubeva, "Construction of Permissible Functions and their Application for Fault Tolerance", *2019 IEEE International Siberian Conference on Control and Communications (SIBCON)*, April 2019. 5 pp.

[8] R. Krieger, B. Becker, and C. Okmen, "OBDD-based Optimization of Input Probabilities for Weighted Random Pattern Generation", *Proc. Fault Tolerant Computing Conference*, 1995, pp. 120–129.

[9] M. Venkatasubramanian and V.D. Agrawal, "A New Test Vector Search Algorithm for a Single Stuck-at Fault using Probabilistic Correlation", *in Proc. of the 2014 IEEE 23rd North Atlantic Test Workshop* (*NATW*), pp. 57–60, August 2014.

[10] M. L. Bushnell and V. D. Agrawal, Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits. Kluwer Academic Publishers, 2000. P. 690.

[11] M. Abramovici, M.A. Breuer, and A.D. Friedman, Digital Systems Testing and Testable Design. New York: Computer Science Press, 1990.

[12] P.K. Lala, "An Introduction to Logic Circuit Testing", Synthesis Lectures on Digital Circuits and Systems 3, no. 1, pp. 1-100, 2008.

[13] R.E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation", *IEEE Trans. on Computers*, vol. C-35, no. 8, pp. 677–691, 1986.

[14] O. Golubeva and D. Ruday, "Construction of Controllability, Observability and Stuck-at Fault Detection Functions", *in Proc. of V Scientific Practical Conf. "Contemporary Problems of Physical and Mathematical Sciences*", September 2019, pp. 281–284.

[15] O. Golubeva, "Algorithms for Obtaining Testability Functions of an Element Pole of a Combinational Circuit", *in Proc. of VI Scientific Practical Conf. "Contemporary Problems of Physical and Mathematical Sciences*", December 2020. (In Russian)

[16] Karpov Yu.G. MODEL CHECKING. Verification of Parallel and Distributed Systems. St. Petersburg, BHV-Petersburg, 2010. 551 p. (In Russian)

[17] A.G. Dyakov, Satisfiability problem (modern solution algorithms), Moscow, 2006. 52 p. (In Russian)