FACTA UNIVERSITATIS (NIŠ)
SER.: ELEC. ENERG. vol. 24, no.3, December 2011, 281-301

Decision Diagrams - from A Mathematical Notion to Engineering Applications

Radomir S. Stanković, Raimund Ubar, and Jaakko T. Astola

Abstract: The paper presents a historical perspective to the theory of decision diagrams from the first definitions of the trees in mathematics related to the representations of discrete sets to the more recent definitions of different decision trees and related diagrams that are widely used in few areas of computing. Special attention has been paid to the relationships (similarities and differences) between Binary decision diagrams (BDD) and Structurally synthesized binary decision diagrams (SSBDD).

Keywords: Switching theory; logic design; decision diagrams; testing of digital devices.

1 Introduction

In mathematic, trees are used represent discrete sets. In set theory, a tree is defined as a partially ordered set (poset) $(T,<)$ such that for each $t \in T$, the set $\{s \in T | s < t\}$ is well-ordered by the relation $<$. See, for instance, [15, 21, 24].

For each $t \in T$, the order type of $\{s \in T | s < t\}$ is called the height of t. In other words, the height of a vertex is the length of the longest path to a leaf from that vertex. The depth of a vertex is the length of the path from the root vertex to this vertex. Thus, the root vertex has the depth of zero.

Single rooted trees are often used, since many problems discussed for multi-rooted trees can be reduced to related problems for single-rooted trees. At the

Manuscript received July 20, 2011. An earlier version of this paper was presented at the Reed Muller 2011 Workshop, May 25-26, 2011, Gustavelund Conference Centre, Tuusula, Finland.

R. S. Stanković is with Dept. of Computer Science, Faculty of Electronics, 18 000 Niš, Serbia, (e-mail: Radomir.Stankovic@gmail.com). R. Ubar is with Dept. of Computer Engineering, Tallinn Technical University, Tallinn, Estonia (e-mail: raiub@pld.ttu.ee). J. T. Astola is with Dept. of Signal Processing, Tampere University of Technology, FIN-33101 Tampere, Finland (e-mail: Jaakko.Astola@tut.fi)

Digital Object Identifier: 10.2298/FUEE1103281S

same time, single rooted trees are easily related to the trees in graph theory, that from another point of view can be efficiently used to represents graphically discrete sets and mappings among them, which is the train of thoughts followed in this paper leading to the decision trees and decision diagrams for representation of logic functions and logic circuits as their implementation in hardware.

It should be noticed that the notion of partially well-ordered sets, upon which the trees are defined, was introduced by Dj. Kurepa in his doctoral dissertation defended at Sorbonne, Paris, France, in 1935 [22]. The dissertation of Kurepa is reported in [21] at page 69, as the first profound study of trees in set theory. In the dissertation Kurepa used the term ramification sets for well-ordered sets (ensemble ramifié and tableaux ramifié in French). See also [23–25]. Dj. Kurepa also defined a concept that is known as the Kurepa tree and set a hypothesis called the Kurepa hypothesis. See, [16, 21, 37, 51], and references therein.

In set theory, many different trees have been introduced, as for instance, the Aronszajn tree, the Kurepa tree, the Suslin tree, that have been introduced by Kurepa, etc., see for instance [15, 21, 22, 24, 25]. Their features and relationships have been studied until the present time [16], and new concept are introduced [51].

A tree can be viewed as either a graph or as a data structure. These two interpretations of this concept are actually equivalent, since a tree as a data structure contains both a set of elements and connections between elements, which results in a tree as a graph.

In graph theory, a directed graph is said to be acyclic if it contains no cycles. An acyclic graph is said to be rooted if exactly one of its vertices, called the root, has no predecessors. In other words, the depth of this vertex is 0. A vertex in a graph with no successors is called a *leaf* or a *terminal* or *constant vertex*. A rooted, acyclic, directed graph is called a tree if each of its vertices, excluding the root, has exactly one predecessor. In other words, any connected[1] acyclic rooted directed graph is a tree.

In computing, we often deal with large sets of discrete data. Therefore, it is natural to use decision trees, and their reduced forms, decision diagrams, for representation and manipulation. These representations naturally extend to representation of tabular data [40], and representation of discrete functions viewed as mappings between discrete and often finite sets [2].

In switching theory, decision trees were used already by C.E. Shannon to represent the sets of all possible minterms of a function with a small number of variables. The same representations can be seen in a paper by Kurepa [26], discussing representations of switching functions in the broader context of mathematical logic and machine implementation of sets of computing instructions, in a presentation at

[1] A graph is described as connected if there is a route of edges and nodes between each two nodes.

the International Symposium on Theory of Switching organized by H. Aiken at his Harvard Lab on April 2-5, 1957.

As it is often the case in science, when the same or similar problems are considered by scholars in different geographic areas, and especially if communication and information flow is prohibited for whatever reasons[2], it happens that the same or closely related concepts are independently introduced and studied, and applied to solve the same or related problems, by different authors at about the same time (or even simultaneously) without being informed about the related work of the others. In this context, it is interesting to discuss relationships, meaning similarities and differences, of the concepts that have been developed independently. Such a study can often provide some new ideas and certainly brings a deeper understanding of the matter.

A particular example of such a situation can be observed when discussing Binary Decision Trees (BDT) and Diagrams (BDD), and Structurally Synthesized Binary Decision Diagrams (SSBDD) and their applications in Switching theory and Logic design.

In this paper, we will briefly discuss these two concepts, their relationships (similarities and differences).

2 Decision Trees and Diagrams in Switching Theory and Logic Design

It is commonly accepted that the profound study and applications of decision diagrams in representation of switching functions and related extensions and generalizations to other classes of functions started after the publication of the seminal paper by R. Bryant in 1986 [12]. It has been pointed out by several scholars that the subject has a much longer history, see for instance [48, 55]. Decision diagrams were used to represent switching functions by S. Lee in 1959 [30]. In 1967 and 1969, the similar concept was used by Ehrenfeticht and Orlowska in [14, 42]. In 1975 and 1977, Kuznetsov worked on synthesis of BDDs from the gate networks.

In 1978 by S.B. Akers [3–5] used BDDs for functional testing.

In 1976, a strongly related concept was introduced by R. Ubar [58] under the name of alternative graphs (AG) for the purpose of structural testing. The concept was extended into the Structural Alternative Graphs (Structural AG) in order to serve better in tasks such as testing of digital systems. The same concept was later reported as Structurally Synthesized Binary Decision Diagrams (SSBDD) and extensively used in many publications, see for instance [43, 45–47, 50, 60–67, 70].

Many different decision diagrams have been defined for the purpose of compact representations of different classes of discrete functions, and the relationships

[2] In the past often caused by political reasons.

among them have been investigated, see for instance [8] for a particular example of such discussions. For a classification of decision diagrams, we refer to [55, 56]. The spectral interpretation provided a unified view to various essentially different decision diagrams [54, 55]. Interestingly, although a lot of work has been done in this area, SSBBDs have remained uninvolved in such considerations. Their place among a variety of decision diagrams used in Switching theory and Logic design is still undetermined. For that reason, this task has been undertaken by the authors of this paper. We will focus on relationships of SSBDD to the most closely related concept of BDD.

3 Binary Decision Diagrams

Binary decision diagrams (BDD) can be viewed as a data structure tailored especially for representation of switching (Boolean) functions. In this context they can be related to the application of the Shannon expansion with respect to all the variables in a function f to be represented. Recall that the Shannon expansion allows to decompose a function f into co-factors $f_0 = f(x_1, \ldots, x_i = 0, \ldots, x_n)$ and $f_1 = f(x_1, \ldots, x_i = 1, \ldots, x_n)$, with respect to each variable x_i, $i = 1, 2, \ldots, n$, as

$$f = \overline{x} f_0 \vee x_i f_1,$$

where $\vee$ is the logic OR.

Note that in the Shannon expansion, that is traditionally viewed as an AND-OR expansion, the logic OR can be replaced by EXOR, since the terms in the expansion are disjoint. Due to that, the Shannon expansion can also be viewed as an AND-EXOR expansion which is an interpretation often used when discussing decision diagrams. Thus, BDD are graphical representations of the functional expressions derived by the recursive application of the Shannon decomposition rule.

BDD can be alternatively discussed as a particular example of directed acyclic graphs consisting of a set of non-terminal vertices, and a set of constant vertices connected with edges. They are obtained from the Binary Decision Trees (BDT) by removing redundant information about the function represented. This redundancy is seen as appearance of isomorphic subtrees, that are subsequently removed by the application of suitably defined reduction rules, see for instance [49]. The problem of reducing the size of BDD was also discussed in [58] by listing the rules for removing redundant parts from the BDD.

For a formal definition of BDD, we refer, for instance, to [48, 49], while for the purpose of considerations in this paper, BDD will be introduced by the Example 1.

4 Complexity of BDD

In this section, we will briefly repeat and discuss some well-known features of BDD in order to explain and justify introduction of many other decision diagrams besides BDD. We will also point out rationales leading to the definition of SSBDD.

A different picture about the compactness of decision diagrams is obtained when representing particular functions that are often met in engineering practice and, on the other hand, arbitrary switching functions.

For many practical functions, BDD are compact in the number of vertices. There are, however, functions, as for instance multipliers, that have BDD of exponential complexity. The size of a BDD is sensitive to the order of the variables, and can range from polynomial to exponential complexity for the same function depending on the selected order of the variables. There are functions that are invariant to the order of the variables, as for instance symmetric functions by definition, and also functions that have BDD of exponential complexity for any order of the variables. Again, multipliers are such an example. For arbitrary switching functions, called general switching functions, the following conclusions are presented in [32].

1. A tight upper bound for the worst case size of reduced OBDD's, is $(2^{n+1}/n)(1+\varepsilon)$ where n is the number of variables and ε is an arbitrarily small positive number.
2. The two reduction rules have disparate reduction capacity. The merging rule alone contributes the reduction factor $1/n$ in the worst case size $O(2^n/n)$, while the deletion rule contributes no more than 1% for large n.
3. Almost all Boolean functions require at least $2^n/2n$ vertices even in the optimal variable ordering. Moreover almost all switching functions are not sensitive to variable ordering.

As pointed out in [32], these complexity measures correspond to those derived by Miller in 1956 for multi-level networks [39], and already in 1949 by Shannon [52] and improved by Lupanov in 1955 [33] for contact networks. For more details on this subject, see the discussion in [20].

It should be noted that BDD provide compact representations for many functions useful in various applications. Nevertheless, there are functions whether the BDD have an exponential complexity in the number of vertices.

That was a reason to define various classes of decision diagrams aimed at compact representation of certain classes of functions.

These diagrams represent functions, that can be viewed as outputs of logic circuits.

We cannot say that these decision diagrams represent circuits, although functions represented by decision diagrams can be easily mapped into circuits [7, 48]. Such circuits will have layout derived from the shape of the used decision diagrams and will consists of modules corresponding to the decomposition rules used in the definition of the diagrams. In the case of BDD, the modules are (2×1) multiplexers, since they can be viewed as a hardware realization of the Shannon decomposition rule.

A desire to derive graphical representations of logic circuits directly, instead of those of the functions they implement, led to the definition of Structurally Synthesized Binary Decision Diagrams [58], that will be briefly introduced in the following sections. It is interesting to note that SSBDD were introduced in 1976 [58] for the purpose of testing of logic networks, which is the same problem to the solution of which BDD were used in 1978 [4].

4.1 Typed Decision Graphs

The introduction of this kind of decision diagrams was driven by certain problems that were noticed in certain concrete industrial applications that were developed in the Bull Corporation in France around 1987. The staff of the Research Center of this corporation, managed by Gerard Roucairol, was occupied among the other things by an extensive simulation of processors, to which task a lot of computer power had to be devoted. François Anceau, a member of the processor-development team of the Center, suggested to prove the correctness of processors versus the VHDL specifications rather than to perform simulations. As noted in [6], the task was transferred to researchers J.P. Billon, and Jean-Christophe Madre, to whom latter joined Alain Coudert, a student at that time. Typed Decision Graphs (TDG) were introduced in [9], and were used as the underlying data structure to represent large Boolean functions.

The main idea behind TDG if that a switching function f and its logic negation f' can be represented by the BDD that differ in the order of values of constant vertices. For instance, if the constant vertices in the BDD for f are 0, and 1, then constant vertices in the BDD for f' will be 1 and 0, otherwise these BDD are identical. This feature enables to increase the number of isomorphic subtrees in a BDD, by assigning a minus sign at the front of a subdiagram representing a subsunction f' that is the logical complement of a subfunction f in the BDD. In this way, if the sign is added to the BDD, the canonicity can be questioned, since the constant value 0 can be represented as either $(+, 0)$ or $(-, 1)$. The problem is resolved when the choice is fixed and applied consistently. For instance, it is pointed out, see for instance [34], that a good choice with respect to the number of vertices in the graph

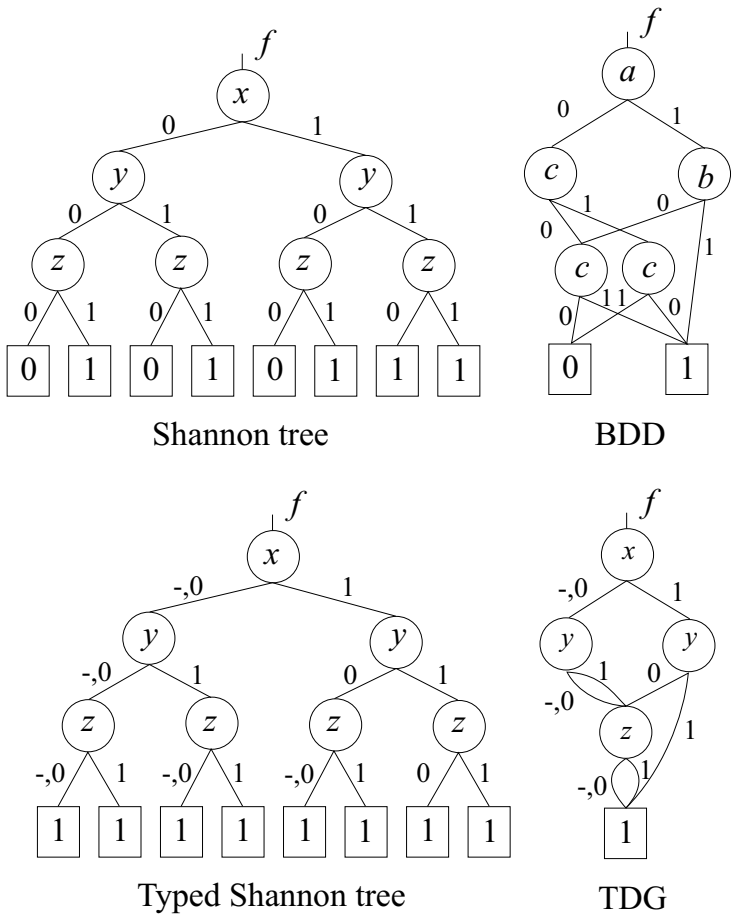

Fig. 1. BDT, BDD, TDG, and reduced TDG for f in Example 1.

is the following [36]

$$Tst(f(x_1,\ldots,x_n))(a1,\ldots,a_i) \stackrel{def}{=} \begin{cases} (+,St(f(x_1,\ldots,x_n))(a_1,\ldots,a_i)) \\ \text{if } f(a_1,\ldots,a_i,1,\ldots,1)=1 \\ (-,St(\overline{f}(x_1,\ldots,x_n))(a_1,\ldots,a_i)), \text{otherwise} \end{cases}$$

It follows that TDG are closely related to the BDD with complemented (also called negated) edges, that are suggested as a way to simplify BDD, see for instance [11], [38].

In TDG, the minus signs are assigned to the left outgoing edges, as this is determined by the corresponding function values, and the values of constant vertices are

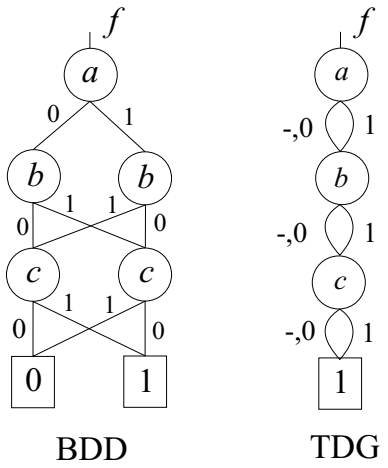

Fig. 2. BDD and TDG for the function in Example 2.

set to 1. Due to these features, they can be related to the Edge-valued binary decision diagrams (EVBDD) with multiplicative attributes, as Factored EVBDD [71].

The following example, taken from [36] illustrates the concept of TDG and links then to BDD.

Example 1 *Fig. 1 shows the Binary decision tree (BDT), Binary decision diagram (BDD), the Typed Shannon tree, and reduced form of it, the Typed Decision graph (TDG) for the function* $f(x,y,z) = xy + y\bar{z} + z\bar{y}$.

The efficiency of TDG compared to BDD is well illustrated by the following example taken from [35]. Further details on TDG can be found in [13].

Example 2 *Fig. 2 shows BDD and TDG for the the function* $f(a,b,c) = a \oplus b \oplus c$. *From this figure, the comment about the link to EVBDD [31] becomes obvious.*

5 Structurally Synthesized Binary Decision Diagrams

In [58], two classes of graphs called *Alternative graphs* (AG) and *Structural alternative graphs* (Structural AG) for the purpose of modeling and testing digital devices were introduced. These graphs have been reported latter as *Structurally Synthesized Binary Decision Diagrams* (SSBDD) in connection with the wide use of BDD after the appearance of the seminal paper by Bryant [12].

Formally, a SSBDD is defined as follows.

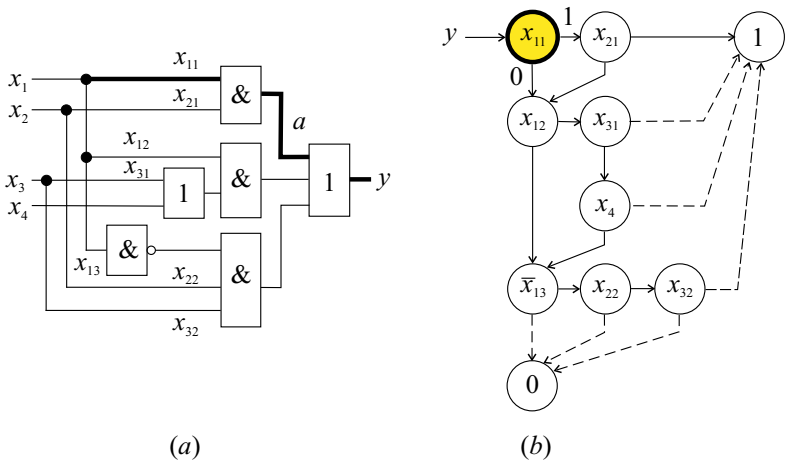

Fig. 3. A circuit and a SSBDD which correspond to the Boolean formula (1).

Definition 1 *A BDD is a rooted directed acyclic graph* $G = (M, \Gamma, X)$ *with a vertex set* $M = M_N \cup M_T$. Γ *is a relation on* M *where* $\Gamma(m) \subset M$ *denotes the set of successors of* m*, and* $\Gamma^{-1}(m) \subset M$ *denotes the set of predecessors of* m*. The nonterminal vertices* $m \in M_N$ *are labeled by Boolean variables* $x(m) \in X$*, and have exactly two successor vertices* $m^e \in \Gamma(m)$*,* $e \in \{0,1\}$*. The graph has two terminal vertices* $m_T \in M_T$ *labeled by a constant* $e(m_T) \in \{0,1\}$ *and called* leafs*. For the terminal vertices* m_T *we have* $\Gamma(m_T) = \emptyset$*, and for the root vertex* $m_0 \in M_N$ *we have* $\Gamma^{-1}(m_0) = \emptyset$*. If there exists an assignment* $x(m) = e$*, then we say that the edge*(m, m^e) *in* G *is activated. Activated edges connecting vertices* m_i *and* m_j *form an activated path* $l(m_i, m_j)$*. An activated path* $l(m_0, m_T)$ *is called* full activated path*.*

Definition 2 *A BDD* $G_y = (M, \Gamma, X)$ *represents a Boolean function* $y = f(X)$ *where* $X = (x_1, x_2, \ldots, x_n)$*, iff for all the possible vectors* $X^t \in \{0,1\}^n$ *there is a path* $l(m_0, m_T)$ *activated in* G_y *so that* $y = f(X^t) = e(m_T)$*.*

Definition 3 *A BDD* $G_y = (M, \Gamma, X)$ *is called SSBDD if it represents a Boolean function* $y = f(X)$ *in the form of equivalent parenthesis expression with* $|IN|$ *literals, which describes a gate-level tree-like combinational circuit* C_y *with a set of inputs IN, and is composed on the basis of AND, OR, and NOT gates, where* $|M| = |IN|$*,* $|X| \leq |M|$*, and there exist a bijection* $M \rightarrow IN$ *and a surjection* $X \rightarrow M$*.*

Example 3 *An example of a SSBDD for the Boolean parenthesis expression*

$$y = (x_{11}x_{12}) \vee x_{12}(x_{31} \vee x_4) \vee \overline{x}_{13}x_{22}x_{32}, \tag{1}$$

which is equivalent to the gate-level circuit in Fig. 3a is represented in Fig. 3b. Here $|M| = |IN| = 8$, *and* $|X| = 4$. *By convention, the right-hand edge from a vertex corresponds to the value* 1, *and the down-hand edge corresponds to the value* 0 *of the vertex variable. Note that vertex variables in a SSBDD may also be inverted (see Section 6). We call this graph structurally synthesized because it is derived from and represents the structure of the formula (1) and the corresponding circuit in Fig. 3a. There exists a one-to-one mapping between the vertices in the graph in Fig. 3b and the inputs of the fan-out free subcircuit in Fig. 3a (and the literals in the formula (1)).*

The SSBDD are generated by the superposition of the BDD of gates or subcircuits of the given circuit, which allows to represent in the model both functionality and structural information of the circuit [58], [66]. Superposition is carried out for a given circuit in the direction from the outputs to the inputs. A similar way is used for creating the BDD for a given logic expression as a process of iterative logic operations with BDD starting from the trivial BDD for variables [49].

The SSBDD model for the given circuit has a linear complexity. This results from the fact that digital circuits are represented as systems of SSBDD, where for each fanout-free region (FFR) a separate SSBDD is generated. On the other hand, the use of the SSBDD model is equivalent to representing of the traditional gate-level network as a higher level module (FFR) network, whereas the gate-level structure of each module is represented more concisely by the corresponding SSBDD.

The following example illustrates how an SSBDD is constructed for the given FFR circuit.

Example 4 *Consider the FFR circuit in Fig. 4. The analysis of the circuit shows that it can be described by the set of equations:* $y = e + f$; $e = a \cdot b$; $f = d \cdot c$; $d = \overline{a}$. *Fig. 5 shows the train of thoughts to construct a SSBDD for this circuit.*

The output y depends on the inputs e and f, *and since this is an OR circuit, if* $e = 0$, *the output depends on* f *(Fig. 5(a)). Since e depends on a and b, we do not show it explicitly, but replace with a and later examine the influence of b. If* $a = 0$, *since the circuit is AND, whatever is the value for b, y will depend just on* f *and we draw an edge from a to* f. *If* $a = 1$, *we ask for the value of b and draw a right edge to b. If* $b = 1$, *we go to the input of the circuit, which is a constant vertex* 1 *in the graph which by the convention is not shown explicitly. If* $b = 0$, *then again y depends solely on* f. *The value of* f *is the output of an AND circuit and depends on its inputs d and c. Thus,* f *is replaced by d and if* $d = 1$ *we ask for the value of c by following the right-hand edge (Fig. 5(c)). The values for c are determined by the inputs of the circuit. Thus, they are constant vertices and not shown. Since d is*

the output of an inverter whit the input a, then $d = \overline{a}$*, and we get the graph as in Fig. 5(d).*

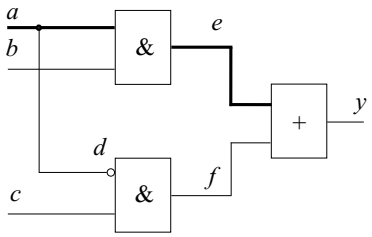

Fig. 4. Circuit realizing the function f in Example 4.

6 Relationships between BDD and SSBDD

BDD and SSBDD are both single rooted binary acyclic graphs. In this way, they belong to the large family of rooted binary acyclic graphs including trees that are used in mathematics for representation of discrete sets and mapping among them. Binary means that there are two outgoing edges of each non-terminal vertex. There are, however, some differences of SSBDD compared to the traditional BDD that will be briefly summarized in the following statements.

BDD and SSBDD can be viewed as function-based and structure-based decision diagrams, respectively, and the most significant difference between these two notions is in the way they are generated.

For a given function, the BDD is constructed by the recursive application of the Shannon expansion rule to all the variables in function to be represented. In the same way, various other decision diagrams are defined by using different expansion rules, as for instance the positive and negative Davio expansion rules [49], [55].

6.1 The meaning of vertices

In a BDD, the vertices correspond to MUX(2×1) modules, that can be viewed as hardware realizations of the Shannon expansion rule. The decision variables assigned to the vertices are used as control variables in the multiplexers. Otherwise, the vertices in BDD have only the functional meaning as the points where decisions of type ”if-then-else” are made.

Statement 1 *The vertices in a SSBDD correspond to signal paths in the gate-level circuit on the basis of AND, OR, and NOT gates, it represents. This is the decisive*

and most important property of SSBDD, and this property results from the method of synthesis of SSBDD.

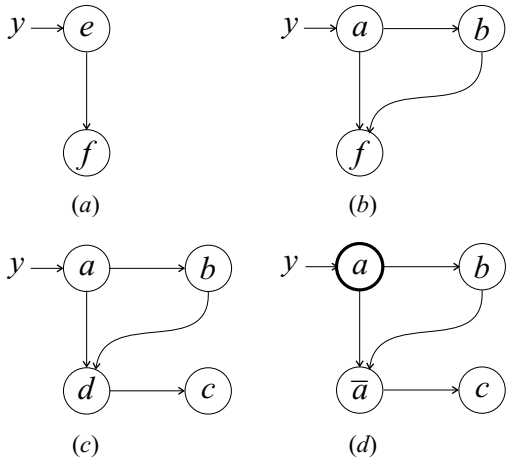

Fig. 5. SSBDD for the function f in Example 4.

Example 5 *For example, the vertex x_{11} denoted by the bold circle in the SSBDD in Fig. 3b, corresponds to the signal path (x_{11}, a, y) shown by bold lines in Fig. 3a. On the other hand, Fig. 6 shows the network realizing the function f in Example 4 derived from the BDD.*

The fact that the vertices in SSBDD represent signal paths in the original circuit allows SSBDD to explicitly model the faults in the circuit. With BDD the faults can be modeled only implicitly by providing the fault lists, and creating a faulty BDD for each fault. As exception, only the faults on the primary inputs of a circuit can be modeled by the vertices of the BDD which represents the function of the circuit.

Statement 2 *SSBDD can represent different structural characteristics of the circuit in a compressed way, similarly as BDD can represent the function of the circuit in a compressed way. Examples of such characteristics are: structural faults on the lines of the circuit, delays on the signal paths, different types of delay faults, static and dynamic hazards in logic. The listed characteristics of logic circuits cannot be simulated explicitly with BDD which represent only the function.*

Example 6 *Since the vertex x_{11} in Fig. 3b corresponds to the signal path (x_{11}, a, y) in Fig. 3a, then the fault stuck-at-1 at the vertex x_{11}, for example, models all three stuck-at-1 faults on the path (x_{11}, a, y).*

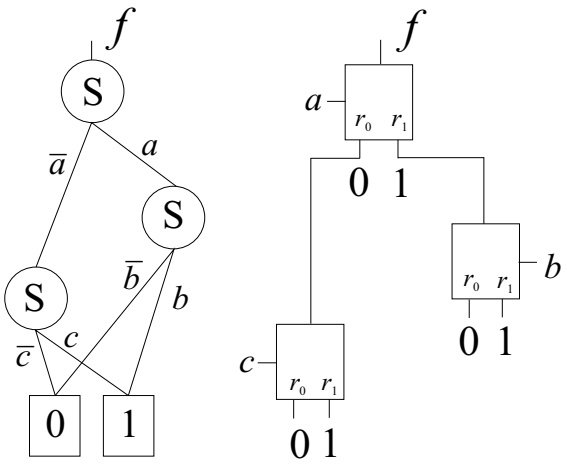

Fig. 6. Network derived from the BDD to realize the function f Example 4.

The compression of fault sets and selecting of representative faults is called *fault collapsing* [57], [69]. Fault collapsing is a side effect of the procedure of SSBDD synthesis. The number of stuck-at-faults processed in the SSBDD model of the given circuit is twice the number of vertices in the model. Typically, the number of representative faults for the SSBDD model is half the number of faults in gate-level circuits [19]. This is nearly the same result achieved by other structural fault collapsing methods like the folding method [57]. However, the SSBDD model represents explicitly the collapsed fault set by the set of vertices, whereas the other gate-level fault diagnosis methods do it implicitly by separate lists of selected representative faults. Recently, an extension of SSBDDS in a form of structurally synthesized multiple input (multiple-rooted) BDD (SSMIBDD) was proposed, which allows additional compression of the model which leads automatically to additional fault collapsing [69].

Since each vertex in the SSBDD represents a signal path in the circuit, it is possible also to represent in a compact way the path delays as vertex delays in SSBDD, and model different delay faults in an efficient way on the SSBDD model [68].

The possibility to model delays on signal paths by SSBDD led to find different applications in simulation of structural aspects of logic circuits like hazard analysis with multi-valued simulation [67], delay fault and timing simulation [18]. The SSBDD achieved higher speed of simulation or analysis compared to the methods that work on the gate level which can be explained by raising the simulation from gate to macro (FFR) level.

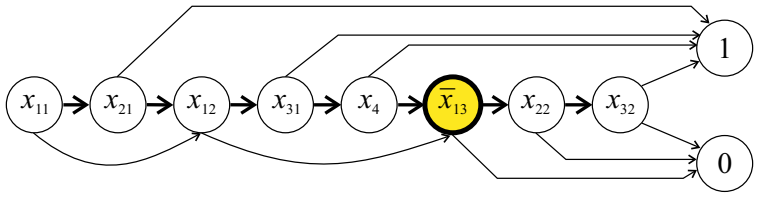

Fig. 7. Ordering of vertices in the SSBDD in Fig. 3.

6.2 Special properties of SSBDD

Statement 3 *In SSBDD all the vertices except the terminal ones with constant labels* 1 *and* 0 *are ordered whereas in the general case of BDD the vertices are only partially ordered.*

Example 7 *The SSBDD represented in Fig. 3, can be stretched out into a string of vertices as shown in Fig. 7, which highlights the ordering of vertices.*

Consider a fully activated path in the given SSBDD which traverses a subset of vertices $M' \subseteq M$. Let us call a vertex m a 1-vertex if there is an assignment $x(m) = 1$, and a 0-vertex if there is an assignment $x(m) = 0$. Note that the variable $x(m)$ may be inverted.

Statement 4 *SSBDD with a function $y = f(X)$ has the following property: (1) in every path from the root vertex to the* 1*-terminal, the variables in the* 1*-vertices form a conjunction of the disjunctive normal form of $f(X)$; (2) in every path from the root vertex to* 0*-terminal, the inverted variables in* 0*-vertices form a conjunction of the disjunctive normal form of the inverted $f(X)$.*

To create SSBDD which have such properties, sometimes it is needed to use inverted variables in the vertices of SSBDD. These properties are useful in using SSBDD for fault simulation and fault diagnosis.

Example 8 *Consider how these properties support fault diagnosis in the given circuit. Let us have an input test pattern* 0111 *applied to the circuit in Fig. 3a which produces on the output $y = 1$. The* 1*-vertices of the full activated path in the SSBDD in Fig. 3b form the conjunction $\overline{x}_{13}x_{22}x_{32}$. If the output value at this pattern will be erroneous $y = 1$, then the conjunction shows the possible causes of the detected error.*

Statement 5 *Each SSBDD is a BDD. The converse is generally not valid. It would be interesting to determine if a given BDD is a SSBDD or not.*

This question was investigated in [44] where the necessary and sufficient conditions were found to say when a BDD is also a SSBDD. SSBDD A gate-level circuit can always be straightforwardly derived from a SSBDD.

6.3 Complexity

Statement 6 *Not all possible* full paths *from the root vertex to terminals in a SSBDD which represents a function $y = f(X)$ can be activated by the assignments of the variables $x \in X$. If a full path cannot be activated we say this full path is* infeasible.

The reason for the infeasibility of a full path in a SSBDD is in the fact that a full path can have the same variable in different vertices both, as inverted and not inverted. Infeasibility of a full path in a SSBDD refers to the inherent redundancy in SSBDD.

Example 9 *Consider the SSBDD in Fig. 7, which is the same SSBDD as in Fig. 3b, where the longest path is stretched out as a string of vertices showing explicitly their ordering. Every SSBDD can always be represented as a similar stretched out string of vertices which represents the longest path in the SSBDD. It is easy to see in Fig. 7 that for activating this longest path for example up to the* 1*-terminal the following condition should be fulfilled:*

$$x_1\overline{x}_2x_1\overline{x}_3\overline{x}_4\overline{x}_1x_2x_3 = 1,$$

which, however, is inconsistent which means that such a full activated path is infeasible. The breakpoint on the highlighted longest path is the vertex $\overline{x}_{13}$. Because of the assignment $x_1 = 1$ made at the root vertex, the traversing of the path after the vertex $\overline{x}_{13}$ ends in 0*-terminal.*

In the BDD generated by recursive use of the Shannon expansion, also have all the full paths feasible. After minimizing the BDD generated for a given logic expression as a process of iterative logic operations [49], the resulting BDD will have also all the full paths feasible.

Statement 7 *SSBDD in general case are redundant. It means that a full path in a SSBDD may have the same variable at different vertices traversed by the path. This is not the case in BDD.*

Having this kind of redundancy in SSBDD is needed for preserving all the important structural properties of the circuit in the model, which allow for the analysis,

for example, of static and dynamic hazards in the given circuit. BDD which are generated from logic expressions of the circuit by iterative logic operations [49], and are optimized afterwards, loose the possibility of analyzing the mentioned structural properties of the given circuit.

The problem with BDD regarding the complexity was discussed in the previous sections. From Statement 7 it results that the SSBDD are even more complex than BDD. Let us compare first the estimations of the complexities of SSBDD and BDD when representing the FFRs. Consider a Boolean function with n variables which represents a tree-like circuit with $|IN|$ inputs.

Statement 8 *The number of vertices $|M|$ in a BDD is $n \leq |M| \leq |IN|$, and in the SSBDD is $|M| = |IN|$, hence, $n \leq |M_{BDD}| \leq |M_{SSBDD}| = |IN|$.*

From the statement it results that the number of vertices in BDD are always equal or less than in SSBDD. The reason is that in SSBDD we have to model explicitly all the signal paths in the original circuit represented by the vertices in SSBDD, and hence, the minimization of the number of vertices in SSBDD is not allowed as in the case of BDD.

Table 1. Comparison of the complexity of BDD and SBDD.

Circuit	In	Out	Gates	ROBDD [3]	FBDD [1]	SSBDD
c432	36	7	232	30200	1063	308
c499	41	32	618	49786	25866	601
c880	60	26	357	7655	3575	497
c1355	41	32	514	39858	N/A	809
c1908	33	25	718	12463	5103	866
c2670	233	140	997	N/A	1815	1313
c3540	50	22	1446	208947	21000	1648
c5315	178	123	1994	32193	1594	2712
c6288	32	32	2416	N/A	N/A	3872
c7552	207	108	2978	N/A	2092	3552

The explosion of the complexity in the case of SSBDD is avoided by using them for representing only FFR subcircuits. The whole circuit is handled as a network of FFR modules where each module is a SSBDD. As a result, the gate-level networks will be substituted by module level networks with less complexity. In Table 1, a comparison of the number of vertices as a measure of complexity of the model for different classes of BDD is shown: ROBDD [3], FBDD [1] and SSBDD taken from [19]. The comparison is given for the benchmarks ISCAS'85.

As can be seen, the complexity of SSBDD is in a linear relation with the complexity of original circuits in the number of gates.

7 Closing Remarks

Switching (Boolean) functions are a particular class of logic functions (binary logic functions), and logic functions can be further viewed as functions over discrete sets. Switching functions are implemented by logic networks that can be viewed as facets of digital systems.

Binary decision diagrams BDD and SSBDD are graphical representations of switching functions and their circuit implementations, respectively. In this context, they are particular examples of decision diagrams used in computing for representations of tabular data (where a hierarchy among data is implicitly assumed through the order of rows and columns) and other ordered sets of data. Due to the existence of some order relations, there is a direct link to trees that are a powerful tool in set theory. Single rooted trees in set theory are directly related to the trees in graph theory via the concept of directed acyclic graphs expressing both sets of elements and connections between elements. Since BDD and SSBDD are both formally defined as particular examples of single rooted acyclic graphs, the train of links among these essential mathematical concepts is completed.

BDD and SSBDD are used in different tasks in digital system representation and modeling, design, verification, testing, and in general the study of digital system.

Acknowledgments

The authors are grateful to the Reviewers whose comments improved the presentation in the paper.

This work was supported by the Academy of Finland, Finnish Center of Excellence Programme, Grant No. 213462, and by Estonian Science Foundation grant 7483, and Research Centre CEBE funded by EU Structural Funds.

References

[1] Abramovici, M., Breuer, M.A., Friedman, A.D., *Digital Systems Testing and Testable Design*, New York, IEEE Press, 1990, 652p.

[2] Akers, S.B., "On a theory of Boolean functions", *Journal of the Sociaety of Industrial and Applied Mathematics*, Vol. 7, No. 4, 1959, 487-498.

[3] Akers, S.B., "Binary decision diagrams", *IEEE Trans. Computers*, Vol. 27, No. 6, 1978, 509-516.

[4] Akers, S.B., "Functional testing with binary decision diagrams", *Proc. 8th Ann. IEEE Conf. Fault-Tolerant Comput.*, 1978, 75-82.

[5] Akers, S.B. , "Functional testing with Binary Decision Diagrams", *J. of Design Automation and Fault-Tolerant Computing*, Vol. 2, 1978, 311-331.

[6] Anceau, F., Private communication 2010.

[7] Astola, J.T., Stanković, R.S., *Fundamentals of Switching Theory and Logic Design*, Springer, 2006.

[8] Becker, B., Drechsler, R., Werchner, R., "On the relation between BDD's and FDD's", *Inform. Comput.*, Vol. 123, No. 2, 1995, 185-197.

[9] Billon, J.P., *Perfect Normal Forms for Discrete Programs*, Technical Report 87039, BULL, France, June 1987.

[10] Boute, R.T., "The Binary Decision Machine as a programmable controller", *EUROMICRO Newsletter*, Vol. 1, No. 2, January 1976, 1622.

[11] Brace, K., Rudell, R., Bryant, R., "Efficient implementation of a BDD package", *Proc. 27th ACM/IEEE Design Automation Conference*, 1990, 40-45.

[12] Bryant, R.E., "Graph-based algorithms for Boolean functions manipulation," *IEEE Trans. Comput.*, Vol. C-35, No. 8, 1986, 667-691.

[13] Coudret, O., Madre, J.C., "Towards an interactive fault tree analyser", *IASTED International Conference on Reliability, Quality Control and Risk Assessment, IASTED'92*, Washington DC, USA, November 1992.

[14] Ehrenfeticht, A., Orlowska, E., "Mechanical Proof Procedure for Propositional Calculus",*Bulletin de l'Academie Polonaise des Sciences*, Serie des sciences math., astr. et phys., - Vol. XV, No. 1, 1967, 25-30.

[15] Jech, T., "Trees", *The Journal of Symbolic Logic*, Vol. 36, 1971, 1-14.

[16] Jin, R., "Some independence results realted to the Krepa tre", *Notre Dame Journal of Formal Logic*, Vol. 32, No, 3, Summer 1991, 448-457.

[17] Jutman, A., Raik, J., Ubar, R., "On efficient Logic-level simulation of digital circuits represented by the SSBDD model", *23rd Int. Conf. on Microelectronics*, Vol. 2, May 2002, 621-624.

[18] Jutman, A., Ubar, R., Peng, Z., "Algorithms for speeding-up timing simulation of digital circuits", *DATE*, Munich, March 13-16, 2001, 460-465.

[19] Jutman, A., Raik, J., Ubar, R., "SSBDD: Advantageous model and efficient algorithms for digital circuit modeling, simulation & test", *5th Int. Workshop on Boolean Problems*, Freiberg, Germany, September 19-20, 2002, 157-166.

[20] Karpovsky, M.G., Stanković, R.S. Astola, J.T., *Spectral Logic and Its Applications for the Design of Digital Devices*, Wiley & Sons, 2008.

[21] Kunen, K., *Set Theory, An Introduction to Independence Proofs*, North-Holland, Amsterdam, 1980.

[22] Kurepa, Dj., *Ensembles ordonnés et ramifiés*, PhD. Thesis, Paris, Sorbonne, 1935. Reprinted in *Publ. Math. Univ. Belgrade*, Vol. 4, 1935, 1-138, and republished in A. Ivić, Z. Mamuzić, Ž. Mijajlović, S. Todorčević (eds.), *Selected papers of Djuro Kurepa*, Matematički institut SANU, Belgrade, Serbia, 1996.

[23] Kurepa, Dj., "L' hypothése de remification", *Comptes Rendus Hebodomaires des Séances de l'Academie des Sciences de Paris*, 1936.

[24] Kurepa, Dj., "Ensembles lineaires et une classe de tableaux ramifiés", *Publ. Math. Univ. Belgrade*, Vol. 6, 129-160.

[25] Kurepa, Dj., ”A propos d’une generalization de la notion d’ensembles bien ordonnés”, *Acta Mathematica*, Vol. 75, 1942, 139-150.

[26] Kurepa, Dj. R., ”Sets-Logics-Machines”, *Proc. Int. Symp. Theory of Switching*, Harvard University, Cambridge, 1957, Part 1, 137-146.

[27] Kurepa, Dj., ”On A-trees”, *Publications de l’Institut Mathematique*, Vol. 8 , No. 22, 1968, 153-161.

[28] Kuznetsov, O.P., ”Grafy logitsheskih avtomatov i ih preobrazovanija”, *Avtomatika i telemehanika*, No. 9, 1975, 149-158.

[29] Kuznetsov O.P, ”O programmnoi realizatsii logitsheskih funktsii i avtomatov, I, Analiz i sintez binarnyh program”, *Avtomatika i telemehanika*, No. 7, 1977, 163-174.

[30] Lee, C.Y., ”Representation of switching circuits by binary-decision programs”, *Bell. Syst. Tech. J.*, Vol. 38, July 1959, 985-999.

[31] Lai, Y.-T., Pedram, M., Vrudhula, S.B.K., ”EVBDD-based algorithms for integer linear programming, spectral transformation, and function decomposition”, *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol. 13, No. 8, 1994, 959-975.

[32] Liaw, H.T., Lin, C.S., ”On the OBDD-representation of general Boolean functions”, *IEEE Trans. Computers*, Vol. 41, No. 6, 1992, 661-664.

[33] Lupanov, O.B., ”On the possibilities of synthesis of networks from different types of elements”, *Dokl. Akad. Nauk SSSR*, 103, 1955, 561-563, (in Russian).

[34] Madre, J.C., Billon, J.P., ”Proving circuit correctness using formal comparison between expected and extracted behavior”, *Proc. 25th DAC*, Anaheim, California, USA, June 12-15, 1988, 205-201.

[35] Madre, J.-C., Coudert, O., Currat, M., Debreil, A., Berthet, C., ”The formal verification chain at BULL”, *Euro ASIC ’90*, 1990, 474-479.

[36] Mauborgne, L., ”Binary decision graphs”, in A. Cortesi, G. Filé, (eds.), *Static Analyis Symposium (SAS99)*, Vol. 1694 of *Lecture Notes in Computer Science*, Springer-Verlag, 1999, 101116.

[37] Mijajlović, Ž., ”Djuro Kurepa”, *Publlications de L’ Institut Mathématique, Nouvelle série*, tome 57, No. 71, 1995, 13-18.

[38] Minato, S., Ishiura, N., Yajima, S., ”Shared binary decision diagrams with attributed edges for efficient Boolean function manipulation”, *Proc. 27th IEEE/ACM DAC*, June 1990, 52-57.

[39] Muller, D.E., ”Complexity in electronic switching circuits”, *IRE Trans. Electron. Comput.*, Vol. EC-5, No. 1, 1956, 15-19.

[40] Miyakawa, M., ”Optimum decision trees - An optimal variable theorem and its related applications”, *Acta Inf.*, Vol. 22, No. 5, 1985, 475-498.

[41] Miyakawa, M., ”Criteria for selecting a variable in the construction of efficient decision trees”, *IEEE Trans. Computers*, Vol. 38, No. 1, 1989, 130-141.

[42] Orlowska, E., ”Mechanical theorem proving in a certain class of formulae of the predicate calculus”, *Studia Logica: An International Journal for Symbolic Logic*, T. 25, 1969, 17-29.

[43] Pall, M., Ubar, R.R., ”Computer-aided module-level test generation for digital devices on the basis of their alterantive-graph-model”, *Preprints SOCOCO-79,Proc. 2nd IFAC/IFIP Symp. on Software for Comp. Contr.*, Prague, 1979, Vol. 2.

[44] Peder, A., Tombak, M., ”Superpositional Graphs”, *Acta et Commentationes Universitatis Tartuensis de Mathematica*, 13, 2009, 51-64.

[45] Plakk, M., Ubar, R., "Digital circuit test design using the Alternative graph model", *Automation and Remote Control*, Vol. 41, No. 5, Part 2, 1980, Plenum Publishing Corporation, USA, 714-722.

[46] Raik, J., Ubar, R., "Feasibility of Structurally synthesized BDD models for test generation", *Proc. of the IEEE European Test Workshop*, 1998, 145-146.

[47] Raik, J., Ubar, R., "Fast test pattern generation for sequential circuits using decision daigram representations", *Journal of Electronic Testing - Theory and Applications*, Vol. 16, 2000, 213-226.

[48] Sasao, T., *Switching Theory for Logic Synthesis*, Kluwer Academic Publishers, 1999.

[49] Sasao, T., Fujita, M., (ed.), *Representations of Discrete Functions*, Kluwer Academic Publishers, 1996.

[50] Seleznev, A., Dobriza, B., Ubar, R., *Design of Automatic Test Equipments*, Mashinostrojenie, Moscow, USSR, 1983, 224 pp., (in Russian).

[51] Shelah, S., Jin, R., "Planting Kurepa trees and killing JechKunen trees in a model by using one inaccessible cardinal", *Fundamenta Mathematicae*, 141, 1992, 287-296.

[52] Shannon, C.E., "The synthesis of two-terminal switching circuits", *Bell System Tech. J.*, Vol. 28, No. 1, 1949, 59-98.

[53] Somenzi, F., "Efficient manipulation of decision diagrams", *Int. Journal on Software Tools for Technology Transfer*, Vol. 3, 2001, 171-181.

[54] Stanković, R.S., "Unified view of decision diagrams for representation of discrete functions", *Multi. Val. Logic*, Vol. 8, No. 2, 2002, 237-283.

[55] Stanković, R.S., Astola, J.T., *Spectral Interpretation of Decision Diagrams*, Springer, 2003.

[56] Stanković, R.S., Sasao, T., "Decision diagrams for discrete functions: classification and unified interpretation", *Proc. Asian and South Pacific Design Automation Conference, ASP-DAC'98* Yokohama, Japan, February 13-17, 1998, 439-446.

[57] To, K., "Fault folding for irredundant and redundant combinational circuits", *IEEE Trans. Computers*, Vol. C-22, No. 11, 1973, 1008-1015.

[58] Ubar, R., "Test generation for digital circuits using alterantive graphs", *Proc. Tallinn Technical University*, Estonia, No. 409, 1976, 75-81 (in Russian).

[59] Ubar, R., "Description of models of digital devices by altrantive grpahs", *Proc. of the Tallinn Polytechnic Institute*, No. 474, 1979, 11-33.

[60] Ubar, R.R., "Beschreibung digitaler Einrichtungen mit alternative Graphen fur die Fehlerdiagnose", *Nachrictentechnik/Elektronik*, 1980, 30, H. 3, 96-102.

[61] Ubar, R., "Desription of computers by vector alternative graphs for diagnostic microprogram synthesis", *Proc. of Tallinn Technical University*, No. 497, 1980, Tallinn, 11-20 (in Russian).

[62] Ubar, R., "Vektorielle Alternative Graphen und Fehlerdiagnose fr digitale Systeme", *Nachrichtentechnik/Elektronik*, Vol. 31, H.1, 1981, 25-29.

[63] Ubar, R., "Test generation for digital systems on the vector alternative graph model", *Proc. of the 13th Annual Int. Symp. on Fault Tolerant Computing*, Milano, Italy, 1983, 374-377.

[64] Ubar, R., "Test generation for microprocessors", *Proc. of the 6th Conf. on Fault-Tolerant Systems and Diagnostics*, Brno, Czechoslovakia, 1983, 209-215.

[65] Ubar, R., "General approach to test synthesis for digital circuits and systems", *Proc. of the 10th All-Union Workshop on Technical Diagnostics*, Tallinn, Oct., 1984, 75-81 (in Russian).

[66] Ubar, R., ”Test synthesis with alterantive graphs”, *IEEE Design & Test of Computers*, 1996, 48-57.

[67] Ubar, R., ”Multi-valued simulation of digital circuits with Structurally synthesized binary decision diagrams”, *Multiple-Valued Logic*, Vol. 4, 1998, 141-157.

[68] Ubar, R., Devadze, S., Raik, J., Jutman, A., ”Parallel X-fault simulation with critical path tracing technique”, *IEEE Conf. Design, Automation & Test in Europe - DATE-2010*, Dresden, Germany, March 8-12, 2010, 1-6.

[69] Ubar, R., Mironov, D., Raik, J., Jutman, A., ”Structural fault collapsing with linear complexity for test generation in digital circuits”, *IEEE Int. Symposium on Circuits and Systems - ISCAS’2010*, Paris, France, May 30-June 2, 2010, 1-6.

[70] Viilup, A., Lohuaru, T., Ubar, R., ”Fault localization in digital circuits with automatic test equipments”, *Proc. of Tallinn Technical University*, No.432, 1977, Tallinn, pp.37-45 (in Russian).

[71] Vrudhula, S.B.K., Pedram, M., Lai, Y.-T., “Edge valued binary decision diagrams,” in: [49], 109-132.