FACTA UNIVERSITATIS (NIŠ)
SER.: ELEC. ENERG. vol. 24, no. 3, December 2011, 303-324

Overview about Low-Level and High-Level Decision Diagrams for Diagnostic Modeling of Digital Systems

Invited paper

Raimund Ubar

Abstract: BDDs have become the state-of-the-art data structure in VLSI CAD. In this paper, a special class of BDDs is presented called Structurally Synthesized BDDs (SSBDD). The idea of SSBDDs is to establish one-to-one mapping between the nodes of SSBDDs and signal paths in the related digital circuit. Such a mapping allowed to investigate and solve with SSBDDs a lot of test and diagnosis related problems of digital circuits, which are associated explicitly with the structure. Such problems are, for example, direct representation of faults, fault collapsing and fault masking, delay testing, hazard detection, etc. The main concept of using SSBDDs is laying on the topological view on the graphs, where each path on a SSBDD can be mapped directly to a subcircuit of the related circuit. Such a topological view allowed to generalize the knowledge and methods of test synthesis and fault analysis from the Boolean level to higher register-transfer and behavior levels of digital systems by introducing High-Level DDs (HLDD). The paper gives a short historical overview of the development of SSBDDs and HLDDs.

Keywords: Binary Decision Diagrams; logic level and high level BDDs; structurally synthesized BDDs.

1 Introduction

Within the last two decades BDDs have become the state-of-the-art data structure in VLSI CAD for representation and manipulation of Boolean functions. BDDs were first introduced for logic simulation in [1], and for test generation in [2, 3]. In

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.

The author is with Computer Engineering Department Tallinn University of Technology Tallinn, Estonia (e-mail: raiub@pld.ttu.ee).

Digital Object Identifier: 10.2298/FUEE1103303U

1986, Bryant proposed a new data structure called reduced ordered BDDs (ROBDDs) [4]. He showed the simplicity of the graph manipulation and proved the model canonicity that made BDDs one of the most popular representations of Boolean functions [5–7]. Different types of BDDs have been proposed and investigated during decades such as shared or multi-rooted BDDs [8], ternary decision diagrams (TDD), or in more general, multi-valued decision diagrams (MDD) [9], edge-valued binary decision diagrams (EVBDD) [8], functional decision diagrams (FDD) [10], zero-suppressed BDDS (ZBDD) [11], algebraic decision diagrams (ADD) [12], Kronecker FDDs [13, 14], binary moment diagrams (BMD) [15], free BDDs [16], multiterminal BDDs (MTBDD) and hybrid BDDs [17], Fibonacci decision diagrams [18] etc. Overviews about different types of BDDs can be found for example in [6, 7, 19].

Traditional use of BDDs has been functional, i.e the target has been to represent and manipulate the Boolean functions by BDDs as efficiently as possible. Less attention has been devoted to represent with BDDs the structural properties of the circuits in the form of mapping between the BDD nodes and the gates, subcircuits or signal paths of the related circuit implementations. The structural aspect of logic circuits was first introduced into BDDs in [2, 20]. The idea was to establish one-to-one mapping between the nodes of BDDs and signal paths in the related digital circuit. Such a mapping allowed to investigate a lot of problems of design and test, which essentially are caused by the structural properties of the given circuit, directly and exclusively with BDDs. These BDDs were called initially alternative graphs [2], and later structurally synthesized BDDs (SSBDD) [21] to stress the way how the BDDs were synthesized directly from the gate-level network structure of logic circuits.

The difficulties in developing of analytical multi-level and hierarchical approaches of digital test generation and fault simulation are related to the need of different languages and models to handle different levels of abstractions. Most frequent examples are logic expressions for combinational circuits, state transition diagrams for finite state machines (FSM), abstract execution graphs, system graphs, instruction set architecture (ISA) descriptions, flow-charts, hardware description languages (HDL, VHDL, Verilog, System C etc.), Petri nets for complex digital systems. Most of them are not well suited for cause-effect reasoning in diagnostic modeling of systems. They also need specialized and dedicated for the given language processing and reasoning algorithms, which makes it difficult to develop uniform approaches to test synthesis, fault analysis and diagnosis. HDL based modeling methods which are efficient for simulation purposes lack the capability of analytical reasoning and analysis that is needed in test generation and fault diagnosis.

Excellent opportunities for multi-level and hierarchical diagnostic modeling of

digital systems are provided by high-level decision diagrams (HLDD). They allow uniform representation of different levels of abstraction, uniform graph-based fault analysis, and uniform effect-cause or cause-effect procedures for diagnostic reasoning of digital systems. The main goal of introducing of HLDDs was to generalize the diagnostic algorithms based on Boolean differential calculus and transformed to the graph language of BDDs for using them at higher levels of system abstraction [22]. Whereas the traditional use of BDDs is based on graph manipulation techniques, the generalization of SSBDD-based diagnosis algorithms for high-level DDs lays mainly on the topological view on the graphs.

The rest of the paper is organized as follows. In Section 2, a short historical overview of the SSBDD development is given, followed by presenting the definition and basic properties of SSBDDs together with describing the test related operations with SSBDDs. Section 3 is devoted to high-level DDs. A short historical overview is given, followed by an example of using HLDDs for representing high-level systems. Thereafter it is explained how the test related operations developed for SSBDDs were generalized to high-level graphs. Section 4 concludes the paper.

2 Overview of Structurally Synthesized SSBDDS

2.1 Short history of SSBDD developments

The BDDs introduced by Lee [1] were not an attractive model for researchers for a long time. The reason was the fast explosion of the complexity of BDDs for large Boolean functions.

In Russia, the idea of using BDDs (named as Alternative Graphs) for representing Boolean functions was mentioned the first time in [23]. In the middle of 70-s the research on BDDs was launched in the Institute of Control Problems, a leading research institute of Russia in Moscow. Several papers were published in the most prestigious Russian journal Avtomatika i Telemehanika (Avtomation and Remote Mechanics) on using binary graphs for representing digital circuits [24], evaluation of the complexity of binary programs [25], and on representing finite state machines with binary programs [26]. However, the BDDs were found to have no future in comparison with existing methods, and the research on this topic stopped.

In parallel, similar research was going on at the Tallinn University of Technology, in Estonia, however, from another point of view. The main target was to create a graph like model in a form of Binary Decision Diagrams to represent structural aspects of combinational circuits, especially, to represent possible structural faults in circuits, for test generation purposes. The graphs were synthesized directly from the topology of the gate-level network, and they were called this time alternative

graphs. The first publications about the alternative graphs (SSBDDs) were in Russian [2], in German [27, 28] in 1976, and in English [29] in 1980. An overview of SSBDDs was given also in the monographs [30, 31].

The main motivation to introduce SSBDDs was to improve the efficiency of test generation methods for combinational circuits by exploiting the possibility to reduce the complexity of the model compared to the traditional gate-level approaches. Different properties of the SSBDD model were investigated in [31–35], which allowed to develop efficient algorithms for test generation [29, 36]. A novel method for synthesizing test pairs to make test sequences robust regarding self-masking of multiple faults, was developed in [29, 37]. In [29], the first time, a general fault model, called later as conditional stuck-at-fault (SAF) model [38, 39] was introduced. Based on this idea, defect oriented fault simulation and test generation methods were developed [40, 41].

Based on the one-to-one mapping between the nodes in SSBDDs and the signal paths in circuits, efficient methods of deductive [42] and critical path tracing [43] fault simulation were developed. A very fast fault simulation approach based on parallel reasoning of faults on SSBDDs simultaneously for many test patterns was developed in [44], and later generalized for extended fault classes like conditional SAF [45] and X-fault model [46]. The first time, a novel algorithm for multivalued simulation based on Boolean differential algebra was implemented with using SSBDDs [21, 47]. Later, the SSBDDs were used for speeding-up timing simulation of digital circuits [48] as well.

SSBDDs have been used for optimization of fault location processes in digital circuits [49, 50], for design error diagnosis [51], for testability evaluation of circuits [52], and for optimization of SSBDDs for fast evaluation of the quality of Built-in Self-Test of digital systems [53]. A new type of SSBDDs with multiple inputs (SSMIBDD) was recently proposed to further optimize SSBDDs for fault collapsing purposes [54], and speeding up logic simulation [55, 56].

SSBDDs have been the basis of several software tools developed for test synthesis and analysis, used in the industry. For example, the first automated test pattern generator (ATPG) in the world based on BDDs was implemented in the beginning of 80-s and used in the defence and computer industries in Soviet Union. In 1986 the authors of the ATPG were awarded by the Silver Medal from the Exhibition of the National Economy in Moscow. Currently, the diagnostic software package Turbo-Tester which includes tools for test generation and fault simulation is used in many universities and institutions throughout the world [57].

2.2 SSBDDs as a structural-functional logic level model for digital circuits

Let us have a tree-like gate level combinational circuit with n inputs. For such a circuit we can create by a superposition of elementary BDDs of the gates a SSBDD with n nodes [20]. Between the paths in the tree and the nodes in the graph, there exists a one-to-one mapping. Every combinational circuit can be regarded as a network of modules, where each module represents a fan-out-free region (FFR) of maximum size. The SSBDD model for a given circuit can be regarded as a set of SSBDDs, where each of them represents such a FFR. This way of modeling the circuit by BDDs allows to keep the complexity of the model (the total number of nodes in all graphs) linear to the number of gates in the circuit.

Definition 1 *SSBDD model for a given combinational circuit is a set of SSBDDs covering all FFRs of maximum size and a set of 1-node SSBDDs covering all primary inputs which have fan-out branches.*

As a side effect of the synthesis of the SSBDD model, we have got a strict relationship between the nodes in the SSBDDs and the signal paths in the modules (FFRs) of the circuit.

SSBDDs reflect two types of mapping between the graph model and the related logic circuit: (1) the nodes in SSBDDs represent signal paths, and (2) certain groups of the nodes in SSBDDs represent certain subcircuits of the whole circuit.

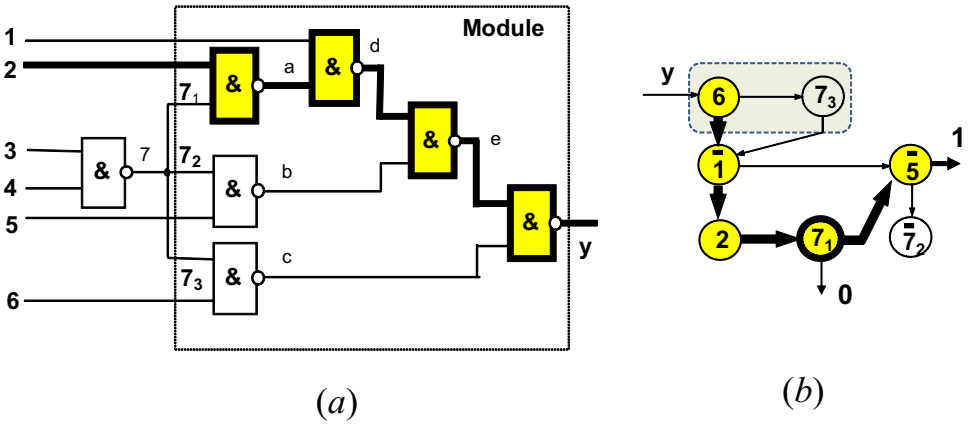


Fig. 1. Combinational circuit and SSBDD.

Example 1 In Fig.1. we have a combinational circuit with a FFR-module and a SSBDD which represents the Boolean function

$$y = (6 \wedge 7_3) \vee (\overline{1} \vee (2 \wedge 7_1))(\overline{5} \vee \overline{7}_2)$$

of the module, whereas the numbers 1,2,5,6 represent input variables, and the subscripted numbers 7_1, 7_2 and 7_3 represent the branches of the fan-out stem 7 in the circuit. The variables in the nodes of SSBDD, in general, may be inverted. The two terminal nodes of the SSBDD are labeled by Boolean constants $\equiv 1$ (truth) and $\equiv 0$ (false). The SSBDD can be interpreted as a procedure of calculating the value of the functional variable y at the given values of the node variables. The procedure is carried out by tracing a path through nodes in the SSBDD. The value 1 of the node variable means the direction to the right from the node, and the value 0 of the node variable means the direction down. Calculation begins in the root node, and the procedure will terminate in one of the terminal nodes ?1 or ?0. The value of y will be determined by the constant in the terminal node where the procedure stops. In this example, an input pattern 110000 (123456) is simulated, and the path

$$(6, \overline{1}, 2, 7, \overline{5}, \equiv 1)$$

is traced (shown by bold lines in Fig. 1), producing the result $y = 1$.

Later, for simplicity, we will omit the two terminal nodes in SSBDDs, and agree that leaving the graph to the right means entering the node #1, and leaving the graph down means entering the node #0.

To each of all 7 signal paths in the circuit, a node in the SSBDD corresponds. For example, to the path from the input of the module 71 through internal nodes *a*, *d*, and *e* in the module up to the output *y*, the node 7_1 in the SSBDD corresponds. On the other hand, for example, the group of nodes 6 and 7_3 in the SSBDD represent a subcircuit of two gates c and y in the circuit.

Corollary 1 Since all the SAF faults on the inputs of a FFR form the collapsed fault set of the FFR, and since all these faults are represented by the faults at the nodes of the corresponding SSBDD, then it follows that the synthesis of a SSBDD is equivalent to the fault collapsing procedure similar to fault folding [54].

Direct relation of nodes to signal paths and groups of nodes to subcircuits allows to handle with SSBDDs easily such problems like fault modeling, fault collapsing, fault masking, path delay or segment delay fault modeling, multi-valued simulation, hazard analysis, fault diagnosis or faulty area diagnosis. Exploring several specific properties of the SSBDDs allowed to increase the efficiency of solving different tasks of test generation, fault simulation and fault diagnosis.

2.3 Operations on SSBDDs

Consider a FFR-module of a circuit with a function $y = f(X)$ where X is set of input variables of the module. Let SSBDD G_y with a set of nodes M represent the

module. Two nodes of M , #0 and #1, are labeled with Boolean constants 0 and 1, respectively. All remaining nodes $m \in M$ are labeled by variables $x(m) \in X$, and have exactly two output edges leading to the successor nodes m^1 and m^0 of m. The variable $x(m)$ may be inverted. The edge (m, m^e) in the SSBDD where $e \in \{0,1\}$ is called activated if $x(m) = e$. A path (m,n) is called activated if all the edges which form the path are activated. To activate a path (m,n) means to assign the node variables along this path with proper values. Let m_0 be the root node of a SSBDD G_y. And, let X^t be an input vector applied at the moment t on the inputs of the module represented by SSBDD G_y. We call the vector X^t as a local test pattern for the module $y = f(X)$.

2.3.1 Logic simulation

Logic simulation on the SSBDD G_y means tracing the graph for the given test pattern X^t which is applied to the module $y = f(X)$. As the result, the value of y is calculated. The value of y will be equal to $e \in \{0,1\}$ if there is activated a path (m_0, #e) from the root node m_0 to the terminal node #e in G_y. Logic simulation can be carried out on SSBDDs in parallel for many test patterns [55].

Example 2 Consider the circuit and its SSBDD in Fig.1. Let for each node m, the right-hand edge is activated by $x(m) = 1$, and the lower-hand edge is activated by $x(m) = 0$. For simplicity, the terminal nodes with constants 0 and 1 are omitted. The bold edges on the SSBDD show the activated path for a test vector $X^t = (1,2,5,6,7) = (11001)$ which calculates the value $y = 1$.

2.3.2 Path activation

A reverse task to logic simulation is path activation which has the goal to find a proper assignment X^t to produce the needed value for y. Test generation on SSBDDs is based on path activation tasks.

2.3.3 Test generation

To generate a test pattern for testing the fault $x(m)/e$ ($x(m)$ stuck-at e) at the node m, $e \in \{0,1\}$, two paths are to be activated: (1) a path (m_0,#e^*),where $e^* \in \{0,1\}$, and $e^* \neq e$, which contains the node m, and (2) a second path (m^e, #e). Both paths should be activated consistently, i.e. by the same assignment X^t. If $x(m) = e^*$ (the correct case) the output value will be $y = e^*$, otherwise, if $x(m) = e$ (the case when the fault is present), $y = e$.

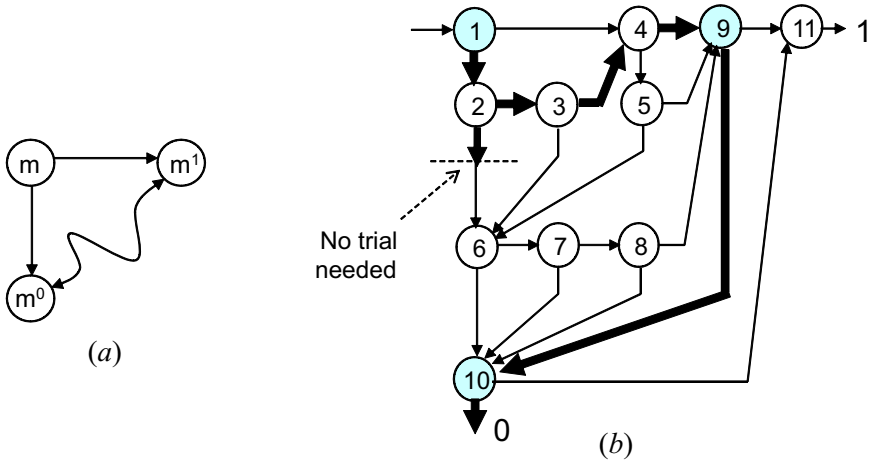


Fig. 2. Illustrations for the Properties 1 (a) and for the Property 2 (b) of SSBDD.

Example 3 As an example, consider test generation on the SSBDD for the fault $7_1/0$ (stuck-at-0 at the node 7_1) in Fig.1. When activating the path (m_0, #1), $7_1 = 1$, through nodes $6, \overline{1}, 2, 7_1, \overline{5}$ on the SSBDD to the terminal node #1, we construct the test pattern $X^t = (1,2,5,6,7) = (11001)$. The second path ($m^e$, #0) consists only of the terminal node #0, and hence, is constantly activated. Note that the test pattern generated for the node 71 of the SSBDD, detects the faults $7_1/0$, $a/1$, $d/0$, $e/1$, and $y/0$ in the related gate-level circuit.

Test generation can be accelerated by exploiting the following properties of SSBDDs.

Property 1 In SSBDDs, for each node m, there exists either a path (m^1, m^0) or a path (m^0, m^1), see Fig.2a. The proof of the Property 1 results from the method of superposition of the SSBDD synthesis [22].

When generating a test pattern for a fault in a module C_k which is a part of a circuit C, then the whole test generation task consists of three subtasks: (1) test generation for a fault in C_k, (2) fault propagation from the output y of C_k to the primary outputs of C, and (3) justification of the assigned values during the first two subtasks by the values of the primary inputs of C. Propagation of the fault from the input of the module to its output is a test generation task for the module. Justification of the value on the output of the module by the values on its inputs is a path activation task. All the listed tasks should be solved consistently by generating a single test vector for the primary inputs of C. When an inconsistency is found, a backtrack to the last assignment should be made to carry out another trial.

The following property of the SSBDDs allows to reduce dramatically the number of backtracks.

Property 2 If in SSBDD, no path (m, #e) with $x(m) = e$, can be activated, then no path (m, #e) with $x(m) = e^*$, where $e \neq e^*$, cannot be activated as well. The proof of the Property 2 is based on the Property 1 [22].

Example 4 In Fig.2b, a path (1, #1) should be activated. Assume that a path (1,9) shown by bold lines is activated. Assume as well that the values of the variables 9 and 10 have been already (during processing of other graphs) assigned to 0, and therefore, no path exists from 9 to the terminal #1. We should backtrace now to the nodes 4, and then to 3 and then to 2 to try other search possibilities.

Using Property 1, there is no need any more for these backtraces, because according to Property 1, after failing in the node 9, we can immediately return the message "No path (1, #1) can be activated".

2.3.4 Fault simulation

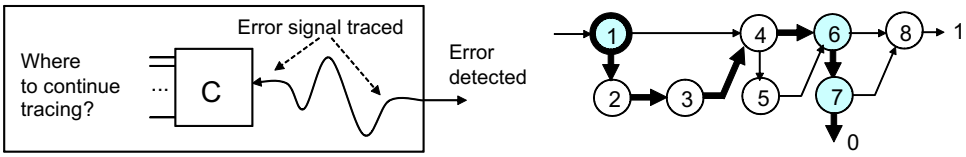
The goal of fault simulation is to determine which faults are detected by the given test pattern. On SSBDDs, first, the activated path (m_0, #e^t), $e^t \in \{0,1\}$, will be determined by logic simulation of the given pattern X^t. Let $M^t \subseteq M$ is the subset of nodes which belong to the activated path (m_0, #e^t). All the nodes in M^t should be suspected as candidate locations of a fault when an error is observed on the output y of the module. However, the set of suspected faulty nodes can be easily pruned by further simulation on the SSBDDs. The fault at the node $m \in M^t$, which changes the value of $x(m)$, is detected by the pattern X^t, if X^t activates as well a second path $(m^*, \#e^{t*})$, where m^* is the successor of m, not belonging to M^t, and $e^{t*} \neq e^t$. The following property (so called "direction rule") for SSBDDs allows to reduce the number of suspected candidate nodes in M^t without simulation, and hence, to speed up the total fault simulation process.

Property 3 If a test vector X^t which activates in the SSBDD a path $(m, \#e^t)$, then only these nodes $m \in M^t$, where $x(m) = e^t$, may be the candidate fault locations and should be checked for existing of the second activated path $(m^*, \#e^{t*})$. The proof is based on Properties 1 and 2 [22].

Example 5 Consider in Fig.1 the path through nodes $6, \overline{1}, 2, 7_1, \overline{5}$ on the SSBDD activated to the terminal with constant 1. According to Property 3, only the nodes 2, 7_1 and $\overline{5}$ may be the candidate fault locations, and should be checked if changing the direction of the path from these nodes will lead to the terminal with constant 0. For the nodes 2 and 7_1, simulation is not needed. The faults in 6 and $\overline{1}$ cannot change the value on the output at this test pattern.

2.3.5 Fault diagnosis

SSBDDs can be efficiently used in fault diagnosis carried out according to the effect-cause concept. Consider a circuit in Fig.3 where an error on the output is detected at the given test pattern. The erroneous signals are backtraced through the circuit in the following way. Assume that the output signal of the module *C* is erroneous. Traditional way would be now to test all the input signals of the module. If all the input signals of *C* will be correct then the fault is located in this module. Otherwise, if an error is detected on an input then the backtracing procedure will be continued from this input towards the primary inputs of the circuit. SSBDDs allow to reduce the number of measurements during the backtracing of error signals.



(*a*) (*b*)

Fig. 3. Fault diagnosis using SSBDDs.

Example 6 In Fig.3, the SSBDD of the module *C* under diagnosis is shown. The module has 8 inputs and the SSBDD has 8 nodes. Consider the path activated by the failed test pattern (bold lines on the SSBDD). The correct value on the output of the module *C* should be 0. Similarly, as during the fault simulation, according to Property 3, only three nodes 1, 6, and 7 from all the eight nodes of the SSBDD may be the fault location candidates and should be pinpointed in the case of the error on the output of *C*.

2.3.6 Multi-valued simulation

SSBDDs allow to carry out dynamic analysis of gate networks (delay fault simulation, hazard detection etc.) by using multi-valued simulation [21]. Consider, as an example, 5-valued simulation with the alphabet $S = \{0, 1, \varepsilon, h, x\}$, where the value 0 (1) represents a type of waveform having a stable logic value 0 (1), $\varepsilon(h)$ represents a waveform having a step-up transition from 0 to 1 (step-down transition from 1 to 0), and x represents unknown waveform or dont care waveform. Handling the values ε, h, x on SSBDDs has a similarity with fault simulation where we consider as well the instability of a signal in this case, correct value vs. erroneous value.

The idea of a uniform handling of the multi-valued simulation and fault simulation is based on partial Boolean derivatives. Let the SSBDD with a set of nodes

M represent a circuit with a function $y = f(X)$. The condition of detecting a fault at the node m can be represented by the Boolean differential equation $\partial y/\partial x(m) = 1$. To satisfy this equation, two paths have to be activated: (m_0, #e) which contains m, and (m^{e^*},#e^*), where $e \neq e^*$. Differently from the fault simulation where the values of variables are binary, in multi-valued simulation, it is not the case.

To solve the problem of multi-valued simulation, in [21] the concept of maximum of Boolean derivatives was introduced, where the $\max\{\partial y/\partial x(m)\}$ was calculated over all combinations of the dynamic values in the alphabet S. For SSBDDs, it means that on the path (m_0, #e), for all the nodes m with encountered values ε, h, x, these values are to be changed to e, and on the related paths (m^{e^*}, #e^*), all the encountered values ε, h, x are to be changed to e^*. Assume, $x(m) = \{\varepsilon, h, x\}$. The multi-valued simulation will have the following result: if $\max\{\partial y/\partial x(m)\} = 1$, then $y = x(m)$, otherwise, if $\max\{\partial y/\partial x(m)\} = 0$, then the value of y will be $e = e^*$. The value x on the output of the module means static hazard. For detecting dynamic hazards, the number of signal values (waveform types) should be increased [21].

Presented above methods of using SSBDDs have been implemented as different tools of test generation, fault simulation and fault analysis and are integrated in a computer aided test tool Turbo-Tester [57].

2.4 Modeling digital systems with multiple input SSBDDs (SSMIBDD)

The idea of decomposition of digital circuits into the FFR modules of maximum size was to keep the complexity of the SSBDD model measured in the number of nodes linear to the number of gates in the circuit [21].

Synthesis of the SSBDD model by the graph superposition procedure [2, 20] starts from the initial set of BDDs, where each logic gate is represented by a BDD. Each superposition step merges two graphs and removes a node from the whole SSBDD model.

In the synthesis approach for SSBDDs, the superposition stopped in the fan-out nodes of the SSBDD where each of them represents a path in the related FFR from a fan-out branch to the output of the FFR. If continuing the superposition beyond the fan-out stems of the circuit, we would have to substitute all the fanout nodes by the BDD of the fan-out gate, which would mean that instead of reducing the model its size in the number of nodes would explode.

By introducing the concept of SSBDDs with multiple inputs (SSMIBDD), we can still further compress the SSBDD model by continuing the superposition procedure beyond the fan-out stems

In [54, 55], a method was proposed for partial superposition of BDDs beyond the fan-out stems of the circuit. To avoid the explosion of the model, and to still continue removing the nodes from the model by superposition, we allow for fanout

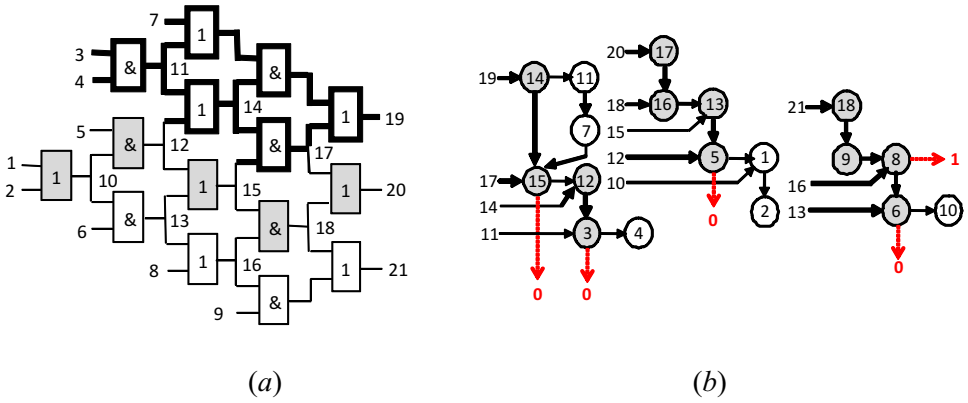


Fig. 4. SSBDDs with multiple inputs (shared SSBDDs).

Table 1. Mapping between the nodes of SSMIBDDS and the paths in the circuit

SSMIBDD for 19	
Node	Path
14	$14_0 - 19$
11	$11_0 - 19$
7	$7 - 19$
15	$15_0 - 17_0 - 19$
12	$12_0 - 14_1 - 17_0 - 19$
3	$3 - 11_1 - 14_1 - 17_0 - 19$
4	$4 - 11_1 - 14_1 - 17_0 - 19$

SSMIBDD for 20	
Node	Path
17	$17_1 - 20$
16	$16_0 - 18_0 - 20$
13	$13_0 - 15_1 - 18_0 - 20$
5	$5 - 12_1 - 15_1 - 18_0 - 20$
1	$1 - 10_0 - 12_1 - 15_1 - 18_0 - 20$
2	$2 - 10_0 - 12_1 - 15_1 - 18_0 - 20$

SSMIBDD for 21	
Node	Path
18	$18_1 - 21$
9	$9 - 21$
8	$8 - 16_1 - 21$
6	$6 - 13_1 - 16_1 - 21$
10	$10_1 - 13_1 - 16_1 - 21$

stems xi with a set of branches $B(x_i) = \{x_{i,1}, x_{i,2}, \ldots, x_{i,k}\}$ the procedure of superposition for a single node $x_{i,j} \in B(x_i)$ only. At this superposition point, an additional input to the SSBDD under construction is established. This would be another entering point for the graph to calculate the value of x_i if a demand for that will appear from the other nodes of the model labelled by $x^*_{i,j} \in B(x_i) \backslash x_{i,j}$. As the result, in the SSMIBDD model of the circuit, the total number of nodes will be smaller than in the SSBDD model, which means additional fault collapsing, whereas the mapping between the nodes in SSMIBDDs and signal paths in the circuit will remain still valid.

Example 7 Consider as an example a combinational circuit and a set of three SSMIBDDs in Fig.4. The BDDs represent disjoint parts of the circuit highlighted in Fig.4. Table 1 describes the mapping between the signal paths in the circuit and the nodes in the SSMIBDD model. We use the notation where the subscripts 0 and 1 of the node numbers in the circuit correspond to the higher and lower fanout branches, respectively. The activated paths are highlighted on the graphs by bold arrows for the input pattern $X^t = (1,2,3,4,5,6,7,8,9) = (100100111)$. The output values of the circuit calculated on the SSMIBDDs are $y_{19} = 0$, $y_{20} = 0$, $y_{21} = 1$.

3 Overview of High Level DDS

3.1 Short history of HLDD developments

High-level approaches to diagnostic analysis of digital systems lay on different languages and models. Most frequent examples are state transition diagrams for finite state machines (FSM), abstract execution graphs, register transfer level (RTL) flowcharts, system graphs, instruction set architecture (ISA) descriptions, hardware description languages (HDL, VHDL, Verilog, System C), Petri nets for system level description. All these models need different dedicated algorithms, which makes it difficult to create a uniform high-level approach to diagnostic analysis of digital systems. Existing high-level modeling methods which are efficient for simulation lack the capability of analytical reasoning that is needed for formalizing test generation and fault diagnosis problems. Promising opportunities for multi-level and hierarchical diagnostic modeling of digital systems provide decision diagrams (DD) because of their uniform cover of different levels of abstraction, and because of their capability for uniform graph-based fault analysis and diagnostic reasoning principles. The most important impact of the high-level DDs (HLDD) is the possibility of generalization and extending of the methods developed for logic circuits using BDDs to higher abstraction levels of digital systems on the uniform graph based formalism.

The first attempt in generalization of BDDs for representing RTL digital systems by introducing vector alternative graphs (VAG) was done in [58,59]. The data path RTL circuits were represented as a set of functionally linked vector HLDDs (called VAGs) where the nodes of graphs were labeled by logic control variables, state variables of FSMs, and n-bit word register variables (instructions or microinstructions). A method was presented for automated test microprogram synthesis for data paths. In [60, 61], a method was proposed for automated synthesis of test programs with HLDDs for microprocessors. A short overview of SSBDDs and HLDDs was given in the All-Union Workshop "Technical Diagnostics" [62], which was the most important event in this field in the former Soviet Union, and

an extended description of the joint theory of DDs was presented in the dissertation [22] which included the implementation results of the SSBDDs and HLDDs in the computer industry of SU.

During 1987 1995, after the collapse of SU and as the result of subsequent economic instabilities in Estonia the scientific research was not the first priority among the students and university staff. The first overview paper in English about the joint presentation of DDs was published in 1996 [20].

During the last decade, the HLDDs have been used in different fields of high-level and hierarchical test. As the result, new promising algorithms, techniques and prototype tools have been developed, which allowed to improve the efficiency of RT level cycle based simulation [63, 64], hierarchical test program automated synthesis [65, 66], hierarchical fault simulation [67, 68], and fault diagnosis [69].

3.2 HLDD as a structural-functional high-level model for digital systems

The goal for introducing of HLDDs was to generalize the logic level methods and algorithms of fault simulation, test generation and fault diagnosis from logic level to higher RTL and functional levels. For this purpose, the class of variables was extended from Boolean ones to the Boolean Vector, and integer variables, and the class of Boolean functions was extended to the data manipulation operations typically used in high-level descriptions of digital systems.

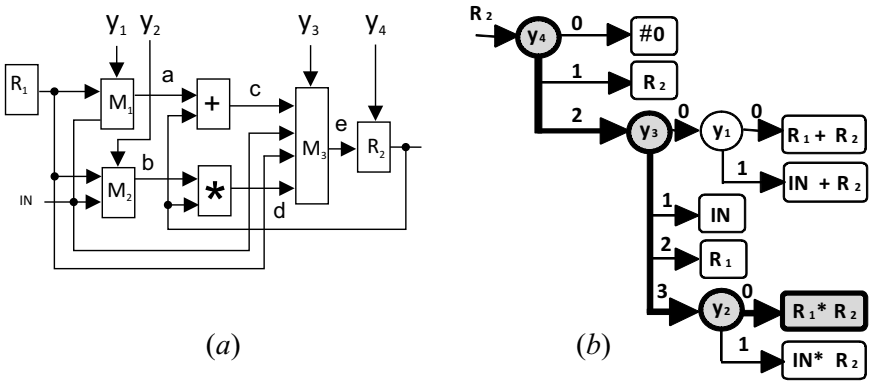


Fig. 5. Representing a register transfer level data path by a HLDD.

In Fig.5, an example of a RTL data-path and its HLDD is presented. The variables R_1 and R_2 represent registers, IN denotes the input bus, the integer variables y_1, y_2, y_3, y_4 represent control signals, M_1, M_2, M_3 are multiplexers, and the functions $R_1 + R_2$ and $R_1 * R_2$ represent the adder and multiplier, correspondingly. Each node in the DD represents a subcircuit of the system (e.g. the nodes y_1, y_2 , y_3,

y_4 represent multiplexers and decoders). The whole DD describes the behavior of the input logic of the register R_2. To test a node in the DD means to test the corresponding to the node component or subcircuit.

Depending on the class of the system (or its representation level), we may have various HLDDs where the nodes have different interpretations and relationships to the system structure. In the RTL descriptions, we usually partition the system into control and data paths. In this case, the nonterminal nodes in the HLDDs correspond to the control path, and they are labeled by state or output variables of the control part, interpreted as addresses or instruction words. On the other hand, the terminal nodes in the HLDDs correspond to the data path, and they are labeled by the data words or functions of data words, which correspond to buses, registers, or data manipulation blocks. The state transfer and output functions of control circuits are represented as well by HLDDs. When using HLDDs for describing complex digital systems, we have to represent the system by a suitable set of interconnected components (combinational or sequential subcircuits). Thereafter, we have to describe the components by their functions which can be represented by HLDDs.

Two methods for synthesis of HLDDs for representing digital systems were described in [65, 70]. The first method is based on symbolic execution of procedural descriptions, which corresponds to the functional representation of systems. The method can be used in cases when the system is given functionally as a procedure in a hardware description language. The second method is based on iterative superposition of HLDDs, and the created model corresponds to the high-level structural representation of the system. The method can be used in cases when the system is given structurally as a network of components (subsystems), and for each component its HLDD is already given. The second method can be regarded as a generalization of the superposition procedure for BDDs [20].

3.3 Operations on HLDDs

The methods for test generation and fault simulation developed for SSBDDs can be easily generalized for using at higher abstraction levels of systems. The possibility of such a generalization results from the topological similarity of DDs at lower and higher levels (Fig.6). In case of SSBDDs, each node has two output edges, and the graph has two terminal nodes with constants 0 and 1. HLDDs differ from SSBDDs in having more edges from nodes and more terminal nodes, whereas the terminal nodes in general case may be labeled by functions. Both graphs represent a mapping into the structure of the system they describe. In both cases, the faults in the system can be modeled similarly by errors in the nodes or in the interconnections between the nodes. In both cases, for both types of graphs, test generation for a given node m is carried out by activating a path from the root node to m and from

all successor nodes of m to different terminal nodes.

Consider a system S as a network of components (or subsystems) where each component is represented by a function $y = f(X)$ where X is a set of variables (Boolean, Boolean vectors or integers), and $V(x)$ is a finite set of possible values of $x \in X$. Let HLDD G_y with a set of nodes M represent this component. The terminal nodes $m^{T,i} \in M^T$ may be labeled either by variables $x(m^{T,i}) \in X$, digital functions $x(m^{T,i}) = f_m^{T,i}(X)$, or constants $x(m^{T,i}) \equiv c_i$. All the remaining nodes $m \in M \backslash M^T$ are labeled by variables $x(m) \in X$, and have $|V(x(m))|$ output edges leading to the successor nodes m^e where $e \in V(x(m))$. The edge (m, m^e) in the HLDD is called activated if $x(m) = e$. A path (m,n) is called activated if all the edges which form the path are activated. To activate a path (m,n) means to assign the node variables along this path with proper values. Let m_0 be the root node of a HLDD G_y. Let X^t be an input vector applied at the moment t on the inputs of the module represented by G_y. We call the vector X^t as a local test pattern for the component $y = f(X)$.

3.3.1 Logic simulation

It is easy to see that the SSBDD defined earlier can be regarded as a special case of HLDDs. Similarly, as we defined the operations of logic simulation and path activation in Section 2.3 for SSBDDs we can do it for HLDDs.

Example 8 In test pattern simulation, a path is traced in the graph, guided by the values of input variables of X until a terminal node is reached, similarly as in the case of SSBDDs. In Fig.5, the result of simulating the vector Xt = (y1, y2, y3, y4, R1, R2, IN) = -,0,3,2,10,6,- is R2 = R1*R2 = 60 (here - means dont care, the bold arrows mark the activated path, and the grey node R1*R2 is reached by simulation). Instead of simulating by a traditional approach all the components in the circuit, in the DD only 3 control variables are visited during simulation (y4, y3, y2,), and only a single data manipulation R2 = R1*R2? MT is carried out.

3.3.2 Test generation

The differences in test generation result from the fact that the number of output edges from nodes and the number of terminal nodes in HLDDs are in general case larger than 2. The fault model in the case of HLDDs is also more complex than in the case of SSBDDs, and will be not discussed here. Without going into details regarding fault handling, consider the following simplified idea of test generation for the nodes of HLDD.

To generate a test pattern for testing a node m, $(n+1)$ paths are to be activated: first, a path (m_0, m), and second, n paths $l_e = (m^e, m^{T,e})$ for all $e \in V(x(m))$, and

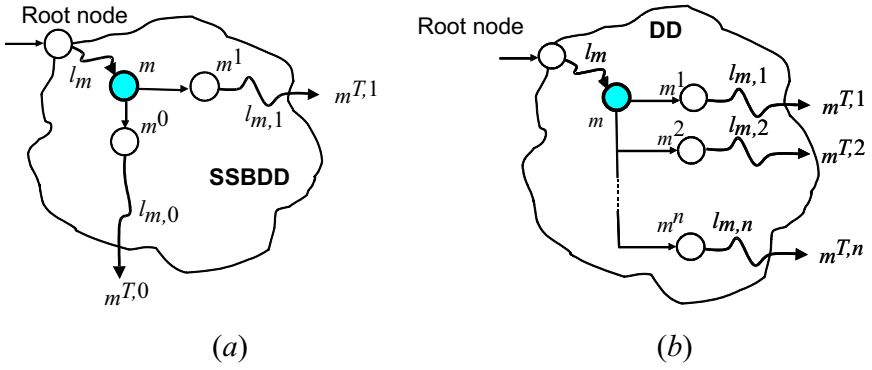


Fig. 6. Path activation and test generation on the SSBDD and the HLDD.

$n = |V(x(m))|$, so that

$$x(m^{T,1}) \neq x(m^{T,2}) \neq \cdots \neq x(m^{T,n}) \tag{1}$$

All paths should be activated consistently by the same assignment X^t. The test vector X^t includes as well the data found by solving the inequality (1). The test for the node m consists of n experiments where $x(m)$ will have all the values of $V(x(m))$. Respectively, the observed function variable y should take the expected values calculated by solving the inequality (1).

Example 9 As an example, consider test generation for testing the multiplexer M_3 represented by the node y_3 in the HLDD in Fig.5. We activate, first, the path from the root node y_4 to the node y_3 under test by assigning $y_4 = 2$. Second, we activate 4 paths from the successors of y_3, for each value $e = 0,1,2,3$ of y_3. Two of the paths, l_1, l_2, for values $e = 1$ and $e = 2$, respectively, are activated "without action", since the successors of y_3 for these values are terminal nodes. Other two paths l_0 and l_3 may be activated, for example, by $y_1 = 0$ and $y_2 = 0$, respectively (the number of other possibilities is three). The test data $R_1 = D_1$, $R_2 = D_2$, $IN = D_3$ are found by satisfying the inequality $R_1 + R_2 \neq IN \neq R_1 \neq R_1 * R_2$.

4 Conclusion

An overview was given about two types of Decision Diagrams Structurally Synthesized BDDs and High-Level DDs for diagnostic modeling of digital systems. The main focus of both models is on the topological view on the graphs and on representing in DDs besides the functions the implementation details of the structure of the system as well. A short insight was given to the history of the development of these models.

SSBDDs are synthesized directly from the topology of the gate-level network of a digital circuit by iterative superposition of partial SSBDDs. SSBDDs reflect two types of mapping between the graph model and the related logic circuit: single nodes in SSBDDs represent single signal paths, and groups of nodes in SSBDDs represent certain subcircuits of the circuit. Such mappings allow to represent by SSBDDs the internal structural details of the circuit, which is precondition for solving the structure related tasks like logic hazard detection, signal timing and signal path delay analysis, fault modeling, test generation and fault diagnosis. An overview about the main properties of SSBDDs and about the diagnostic modeling related algorithms was given.

To overcome the difficulties of high-level diagnostic reasoning of complex digital systems when using traditional hardware description languages, High-Level DDs were introduced. HLDDs provide a basis for analytical cause-effect and effect-cause reasoning that is needed in automated test program synthesis and fault diagnosis in digital systems. Introducing of HLDDs allowed to generalize the diagnostic algorithms developed for SSBDDs for using them at higher levels of system abstraction. While the traditional use of BDDs is based on graph manipulation techniques, the generalization of diagnosis algorithms from logic level SSBDDs to high-level DDs lays mainly on the topological properties of the graphs. SSBDDs used for representing logic circuits can be regarded as a special case of HLDDs for representing digital systems on higher abstraction levels. In a similar way, we can regard the stuck-at-fault model defined for SSBDDs as a special case of the node fault model for HLDDs.

Acknowledgment

The authors are grateful to the Reviewers whose comments helped to improve the presentation

The work has been supported by Estonian SF grant 7483, FP7 IST project DIAMOND, and Research Centre CEBE funded by EU Structural Funds.

References

[1] C. Y. Lee, “Representation of switching circuits by binary decision programs,” *The Bell System Technical Journal*, pp. 985–999, July 1959.

[2] R. Ubar, “Test generation for digital circuits with alternative graphs,” *Proceedings of Tallinn Technical University*, no. 409, pp. 75–81, 1976, in Russian.

[3] S. B. Akers, “Functional testing with binary decision diagrams,” *J. of Design Automation and Fault-Tolerant Computing*, Oct. 1978.

[4] R. E. Bryant, “Graph-based algorithms for boolean function manipulation,” *IEEE Trans. on Computers*, vol. C-35, no. 8, pp. 667–690, 1986.

[5] S. Minato, *BDDs and Applications for VLSI CAD*. Kluwer Academic, 1996.

[6] T. Sasao, *Representations of Discrete Functions*, M. Fujita, Ed. Kluwer Academic, 1996.

[7] R. Drechsler and B. Becker, *Binary Decision Diagrams*. Kluwer Academic, 1998.

[8] S. Minato, N. Ishiura, and S. Yajima, "Shared binary decision diagrams with attributed edges for efficient boolean function manipulation," in *Proc. 27th ACM/IEEE Design Automation Conference, IEEE/ACM ICCAD90*, Orlando, FL, USA, June 24–28, 1990, pp. 52–57.

[9] A. Srinivasan, T. Ham, S. Malik, and R. K. Bryanton, "Algorithms for discrete function manipulation," in *Proc. International Conference on Computer-Aided Design. ICCAD-90. Digest of Technical papers*, 11–15 1990, pp. 92–95.

[10] U. Kebschull, E. Schubert, and W. Rosenstiel, "Multilevel logic synthesis based on functional decision diagrams," in *3rd European Conference on Design Automation, IEEE EDAC92*, Mar. 16–19, 1992.

[11] S. Minato, "Zero-suppressed bdds for set manipulation in combinational problems," in *Proc. 30th Conference on Design Automation ACM/IEEE DAC*, June 14-18, 1993, pp. 272–277.

[12] R. I. Bahar, E. A. Frohm, C. M. Gaona, G. D. Hachtel, E. Macii, A. Pardo, and F. Somenzi, "Algebraic decision diagrams and their applications," in *Int. Conf. on Computer Aided Design. ICCAD-93. Digest of technical papers*, Nov. 7–11, 1993, pp. 188–191.

[13] A. Sarabi, P. F. Ho, K. Iravani, W. R. Daasch, and M. A. Perkowski, "Minimal multilevel realization of switching functions based on kronecker functional decision diagrams," in *Proc. of IEEE International Workshop on Logic Synthesis, IWLS '93*, Tahoe City, CA, USA, May 1993, pp. P3a–1 – P3a–6.

[14] R. Drechsler, A. Sarabi, M. Theobald, B. Becker, and M. A. Perkowski, "Efficient representation and manipulation of switching functions based on ordered kronecker functional decision diagrams," in *31st Conference on design Automation, DAC-1994*, June 6–10 , 1994, pp. 415–419.

[15] R. E. Bryant and Y.-A. Chen, "Verification of arithmetic functions with binary moment diagrams," in *Proc. 32nd ACM/IEEE DAC*, 1995.

[16] J. Bern, C. Meinel, and A. Slobodova, "Efficient obdd-based boolean manipulation in cad beyond current limits," in *32nd Conference on Design Automation, DAC'95*, San Francisco, CA, USA, 1995, pp. 408–413.

[17] E. M. Clarke, M. Fujita, and X. Zhao, *Representations of Discrete Functions*, multi-terminal binary decision diagrams and hybrid decision diagrams ed. Kluwer Academic, 1996, pp. 93–108.

[18] R. S. Stanković, J. Astola., M. Stanković, and K. Egiazarian, "Circuit synthesis from fibonacci decision diagrams," *VLSI Design, Special Issue on Spectral Techniques and Decision Diagrams*, vol. 14, pp. 23–34, 2002.

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

[20] R. Ubar, "Test synthesis with alternative graphs," *IEEE Design and Test of Computers*, pp. 48–57, 1996.

[21] ——, "Multi-valued simulation of digital circuits with structurally synthesized binary decision diagrams," *Multiple Valued Logic*, vol. 4, pp. 141–157, 1998.

[22] ——, "Diagnostics of complex digital systems," Ph.D. dissertation, Latvian Academy of Sciences, Riga, Latvia, 1986.

[23] B. N. Shneider, "O realizatsii bulevyx funktsii alternativnymi grafami," in *II Vsesojuznoje soveshtshanije po teorii releinyh ustroistv I konetshnyh avtomatov*, Riga, Latvia, 1971, p. 1.

[24] O. P. Kuznetsov, "Grafy logitsheskih avtomatov i ih preobrazovanija," *Avtomatika I telemehanika*, no. 9, pp. 149–158, 1975.

[25] V. A. Kuzmin, *Otsenka slozhnosti realizatsii funktsii algebry logiki prosteishimi vidami binarnyh programm*, 1976, vol. 29, pp. 11–39, sb. Trudov I nstituta Matematiki SO AN SSSR.

[26] O. P. Kuznetsov, "O programmnoi realizatsii logitsheskih funktsii i avtomatov. i. analiz i sintez binarnyh programm," *Avtomatika I telemehanika, year=*.

[27] R. Ubar, "Berechnung von tests für die fehlerdiagnose in digitalen systemen," in *Proc. of 21. Int. Wiss. Koll.* Ilmenau, Germany: Technical University of Ilmenau, Oct. 1976, pp. 33–35.

[28] ——, "Beschreibung digitaler einrichtungen mit alternativen graphen für die fehlerdiagnose," *Nachrichtentechnik/Elektronik*, no. 3, pp. 96–102, 1980.

[29] M. Plakk and R. Ubar, "Digital circuit test design using the alternative graph model," in *Automation and Remote Control*. Plenum Publishing Corporation, USA, Nov. 1980, vol. 41, no. 5, pp. 714–722.

[30] R. Ubar, *Diagnosis of Digital Devices*. Tallinn Technical University, 1980, vol. I and II, in Russian.

[31] A. Seleznev, B. Dobriza, and R. Ubar, *Design of Automatic Test Equipments*. Moscow, USSR: Mashinostrojenie, 1983, in Russian.

[32] R. Ubar, "Description of digital devices by alternative graphs," in *Proc. of Tallinn Technical University*, no. 474, Tallinn, Estonia, 1979, pp. 11–33.

[33] A. Jutman, A. Peder, J. Raik, M. Tombak, and R. Ubar, "Structurally synthesized binary decision diagrams," in *6th International Workshop on Boolean Problems*, Freiberg, Germany, Sept. 2004, pp. 271–278.

[34] A. Jutman, J. Raik, and R. Ubar, "On efficient logic-level simulation of digital circuits represented by the ssbdd model," in *23rd Int. Conf. on Microelectronics*, vol. 2, Niš, Serbia, May 12–15, 2002, pp. 621–624.

[35] ——, "Ssbdds: Advantageous model and efficient algorithms for digital circuit modeling, simulation and test," in *5th Int. Workshop on Boolean Problems*, Freiberg, Germany, Sept. 19–20, 2002, pp. 157–166.

[36] R. Ubar, "Combining functional and structural approaches in test generation for digital systems," *Journal of Microelectronics and Reliability*, vol. 38, no. 3, pp. 317–329, 1998.

[37] M. Plakk and R. Ubar, "Synthesis of test pairs for combinational circuits," in *Proceedings of Tallinn Technical University*, no. 474, Tallinn, 1979, pp. 45–68, in Russian.

[38] K. B. Keller, "Hierarchical pattern faults for describing logic circuit failure mechanisms," Patent 5 546 408, Aug. 13, 1994.

[39] R. D. Blanton and J. P. Hayes, "On the properties of the input pattern fault model," *ACM Trans. Des. Automat. Electron. Syst.*, vol. 8, no. 1, pp. 108–124, Jan. 2003.

[40] R. Ubar, W.Kuzmicz, W. Pleskacz, and J. Raik, "Defect-oriented fault simulation and test generation in digital circuits," in *2nd Int. Symp. on Quality of Electronic Design*, San Jose, California, USA, Mar. 26–28, 2001, pp. 365–371.

[41] J. Raik, R. Ubar, J. Sudbrock, W. Kuzmicz, and W. Pleskacz, "Dot: New deterministic defect-oriented atpg tool,."

[42] R.Ubar, "Fault simulation in digital systems using alternative graphs," in *36. Int. Wiss. Koll.*, Ilmenau, Germany, Oct. 21–24, 1991, pp. 737–742.

[43] J. Raik, R. Ubar, S. Devadze, and A. Jutman, *Efficient Single-Pattern Fault Simulation on Structurally Synthesized BDDs*. Berlin, Heidelberg, New York: Springer Verlag, 2005, vol. 3463, pp. 332–344.

[44] R. Ubar, S. Devadze, J. Raik, and A. Jutman, "Fast fault simulation in digital circuits with scan path," in *13th Asia and South Pacific Design Automation Conference ASP-DAC*, Seoul, Korea, Jan. 21–24, 2008, pp. 667–672.

[45] ——, "Fast fault simulation for extended class of faults in scan-path circuits," in *5th IEEE Int. Symposium on Electronic Design, Test and Applications DELTA 2010*, Ho Chi Minh City, Vietnam, Jan. 13–15, 2010, pp. 14–19.

[46] ——, "Parallel x-fault simulation with critical path tracing technique," in *IEEE Conf. Design, Automation and Test in Europe DATE-2010*, Dresden, Germany, Mar. 8–12, 2010, pp. 1–6.

[47] V. Alango, T. Kont, and R. Ubar, "New test design techniques for fault detection in digital objects," in *Proc. of Tallinn Technical University*, no. 708, Tallinn, Estonia, 1990, pp. 45–61.

[48] A. Jutman, R. Ubar, and Z. Peng, "Algorithms for speeding-up timing simulation of digital circuits," in *IEEE Conf. Design, Automation and Test in Europe DATE-2001*, Munich, Germany, Mar. 13–16, 2001, pp. 460–465.

[49] A. Viilup, T. Lohuaru, and R. Ubar, "Fault localization in digital circuits with automatic test equipments," in *Proc. of Tallinn Technical University*, no. 432, Tallinn, Estonia, 1977, pp. 37–45.

[50] R. Ubar, "Fault diagnosis in digital devices," in *Proceedings of the Estonian Academy of Sciences, Engng.*, 1995, no. 1/1, pp. 51–67.

[51] ——, "Design error diagnosis with resynthesis in combinational circuits," *Journal of Electronic Testing: Theory and Applications*, vol. 19, pp. 73–82, 2003.

[52] R. Ubar, J. Heinlaid, J. Raik, and L. Raun, "Calculation of testability measures on structurally synthesized binary decision diagrams," in *Proc. of the 6th Baltic Electronics Conference*, Tallinn, Estonia, Oct. 7–9, 1998, pp. 179–182.

[53] R. Ubar, T. Vassiljeva, J. Raik, A. Jutman, M. Tombak, and A. Peder, "Optimization of structurally synthesized bdds," in *The 4th IASTED International Conference on Modelling, Simulation and Optimization*, Kauai, Hawaii, USA, Aug. 17–19, 2004, pp. 234–240.

[54] R. Ubar, D. Mironov, J. Raik, and A. Jutman, "Structural fault collapsing by superposition of bdds for test generation in digital circuits," in *IEEE 11th International Symposium on Quality Electronic Design*, San Jose, CA, USA, Mar. 22-24, 2010, pp. 250–257.

[55] D.Mironov, R. Ubar, S. Devadze, J. Raik, and A. Jutman, "Structurally synthesized multiple input bdds for speeding up logic-level simulation of digital circuits," in *Euromicro Conf. on Digital System Design DSD'2010*, Lille, France, Sept. 1–3, 2010, pp. 658–663.

[56] G. Jervan, A. Markus, P. Paomets, J. Raik, and R. Ubar, "Turbo tester: A cad system for teaching digital test," in *Microelectronics Education*. Kluwer Academic Publishers, 1998, pp. 287–290.

[57] [Online]. Available: http://www.pld.ttu.ee/tt/

[58] R. Ubar, "Description of computers by vector alternative graphs for diagnostic microprogram synthesis," in *Proc. of Tallinn Technical University*, Tallinn, Estonia, 1980, no. 497, pp. 11–20, in Russian.

[59] ——, “Vektorielle alternative graphen und fehlerdiagnose für digitale systeme,” *Nachrichtentechnik/Elektronik*, vol. 31, no. 1, pp. 25–29, 1981.

[60] ——, “Test generation for digital systems on the vector alternative graph model,” in *Proc. of the 13th Annual Int. Symp. on Fault Tolerant Computing*, Milan, Italy, year=.

[61] ——, “Test generation for microprocessors,” in *Proc. of the 6th Conf. on Fault-Tolerant Systems and Diagnostics*, Brno, Czechoslovakia, 1983, pp. 209–215.

[62] ——, “General approach to test synthesis for digital circuits and systems,” in *Proc. of the 10th All-Union Workshop on Technical Diagnostics*, Tallinn, Estonia, Oct. 1984, pp. 75–81, in Russian.

[63] R. eveugle and R. Ubar, “Modeling vhdl clock-driven multi-processes by decision diagrams,” *J. of Electron Technology*, vol. 32, no. 3, pp. 282–287, 1999.

[64] A. Morawiec, R. Ubar, and J. Raik, “Cycle-based simulation algorithms for digital systems using high-level decision diagrams,” in *IEEE Proc. of Design Automation and Test in Europe DATE’2000*, Paris, France, Mar. 27–30, 2000, p. 743.

[65] J. Raik and R. Ubar, “Fast test pattern generation for sequential circuits using decision diagram representations,” *Journal of Electronic Testing: Theory and Applications*, vol. 16, no. 3, pp. 213–226, 2000.

[66] G. Jervan, R. Ubar, Z. Peng, and P. Eles, *Test Generation: A Hierarchical Approach*.

[67] R. Ubar, J. Raik, E. Ivask, and M. Brik, “Multi-level fault simulation of digital systems on decision diagrams,” in *IEEE Workshop on Electronic Design, Test and Applications DELTA’02*, Christchurch, New Zealand, Jan. 29–31, 2002, pp. 86–91.

[68] R. Ubar, S. Devadze, M. Jenihhin, J. Raik, G. Jervan, and P. Ellervee, “Hierarchical calculation of malicious faults for evaluating the fault-tolerance,” in *4th IEEE International Symposium on Electronic Design, Test and Applications DELTA 2008*, Hong Kong, Jan. 23–25, 2008, pp. 222–227.

[69] J. Raik, U. Repinski, R. Ubar, M. Jenihhin, and A. Chepurov, “High-level design error diagnosis using backtrace on decision diagrams,” in *The 28th IEEE NORCHIP Conference*, Tampere, Finland, Nov. 5–16, 2010, pp. 1–4.

[70] R. Ubar, J. Raik, A. Karputkin, and M. Tombak, “Synthesis of high-level decision diagrams for functional test pattern generation,” in *16th Int. Conference MIXDES 2009*, Lodz, Poland, June 25–27, 2009, pp. 519–524.