ВЕСТНИК ТОМСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

2014 Управление, вычислительная техника и информатика № 3 (28)

UDK 004.312

Raimund Ubar

FAULT EFFECT REASONING IN DIGITAL SYSTEMS BY TOPOLOGICAL VIEW ON LOW- AND HIGH-LEVEL DECISION DIAGRAMS

To memory of my opponent Arkadi Zakrevski

In order to cope with complexity of diagnostic reasoning of today's digital systems, hierarchical multi-level approaches should be used. In this paper, the possibilities of using Decision Diagrams (DD) for uniform diagnostic modeling of digital systems at different levels of abstraction are discussed. Binary Decision Diagrams (BDD) have become the state-of-the-art data structure in VLSI CAD. 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 gate-level networks, which allows to investigate and solve with SSBDDs test and diagnosis problems directly associated with structural aspects of circuits, like fault modeling, fault collapsing, fault masking, delays, hazards. The main concept of SSBDDs lays on exploiting the topology of graphs for fault reasoning which allows to generalize the methods of test synthesis and fault analysis from the Boolean level to higher register-transfer or behavior levels of hierarchy by introducing a novel High-level DDs (HLDD).

Keywords: Binary Decision Diagrams (BDD); structurally synthesized BDD (SSBDD); shared SSBDD (SSSBDD); high level DD (HLDD); test generation and fault diagnosis.

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 1959 [1], and for logic level diagnostic modeling in [2, 3]. In 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, 6]. Different types of BDDs have been proposed and investigated during decades such as shared or multi-rooted BDDs [7], ternary decision diagrams (TDD), or in more general, multi-valued decision diagrams (MDD) [8], edge-valued BDDs (EVBDD) [7], functional decision diagrams (FDD) [9], zero-suppressed BDDS (ZBDD) [10], algebraic decision diagrams (ADD) [11], Kronecker FDDs [12], binary moment diagrams (BMD) [13], free BDDs [14], multiterminal BDDs (MTBDD) and hybrid BDDs [15], Fibonacci decision diagrams [16] etc. Overviews about different types of BDDs can be found in [5, 6, 17].

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 representing with BDDs the structural aspects of circuits. Such a goal was first set up in [2, 18] and realized by introducing a possibility for one-to-one mapping between the nodes of BDDs and signal paths in the related circuit. A special class of BDDs was introduced, called initially alternative graphs (AG) [2]. Later AG was renamed as structurally synthesized BDD (SSBDD) [18, 19] in accordance to the way how they were synthesized - directly from the gate-level network structure of logic circuits.

The direct mapping between SSBDDs and circuits allows to model different test related objectives and relations of gate level networks like signal paths, faults in gates or connections, delays on paths, fault masking, fault equivalence and dominance, etc. These issues are difficult to model and simulate explicitly with “classical” BDDs.

Whereas logic level test and diagnosis methods are well developed, this is not the case for higher level test approaches based on abstract execution graphs, system graphs, instruction set architecture (ISA)

descriptions, flowcharts, hardware description languages (HDL, VHDL, Verilog, SystemC), or Petri nets for complex digital systems. Most of these modeling tools are not well suited for reverse simulation and effect-cause reasoning in digital systems. They also need specialized and dedicated for the given language fault proccessing and reasoning algorithms, which makes it difficult to develop hierarchical approaches for test synthesis, fault analysis and diagnosis. HDL based modeling methods, which are efficient for simulation purposes, do not support analytical reasoning and analysis that is needed in test generation and fault diagnosis.

To overcome this gap, High-Level DDs [HLDD] were introduced as extension of BDDs [18,19]. The topological basis of test algorithms developed for SSBDDs allows to extend these algorithms in a rather straightforward way from logic level to high level. The class of node variables in DDs was extended from Boolean to Boolean vectors and to integer variables, whereas the class of Boolean functions was extended to data manipulation operations typically used in high-level descriptions of digital systems.

The rest of the paper is organized as follows. Section 2 presents the SSBDD as a structural model for gate-level ciruits. In Section 3, the main ideas of test generation and fault reasoning based on the topological view on SSBDDs are discussed. Section 4 demonstrates the possibility of using SSBDDs for reasoning multiple faults and for detection of fault masking. An extension of SSBDD model by introducing shared SSBDDs is discussed in Section 5. An overview of generalization of logic level DDs for higher abstraction levels to represent complex digital systems is given in Section 6. Finally, Section 7 discusses the ideas of using HLDDs for automatization of hierarchical test program synthesis for digital systems at RTL and ISA levels, and Section 8 concludes the paper.

1. S SBDD – a model for gate-level circuits

Let us have a gate level combinational circuit with fan-outs only at inputs. Consider the maximum *fan-out free region* (FFR) of the circuit with inputs at the fan-out branches and fan-out free inputs. Let the number of the inputs of FFR be *n.* For such a *tree-like sub-circuit* we can create by *superposition* of BDDs of gates an SSBDD with *n* nodes [19].

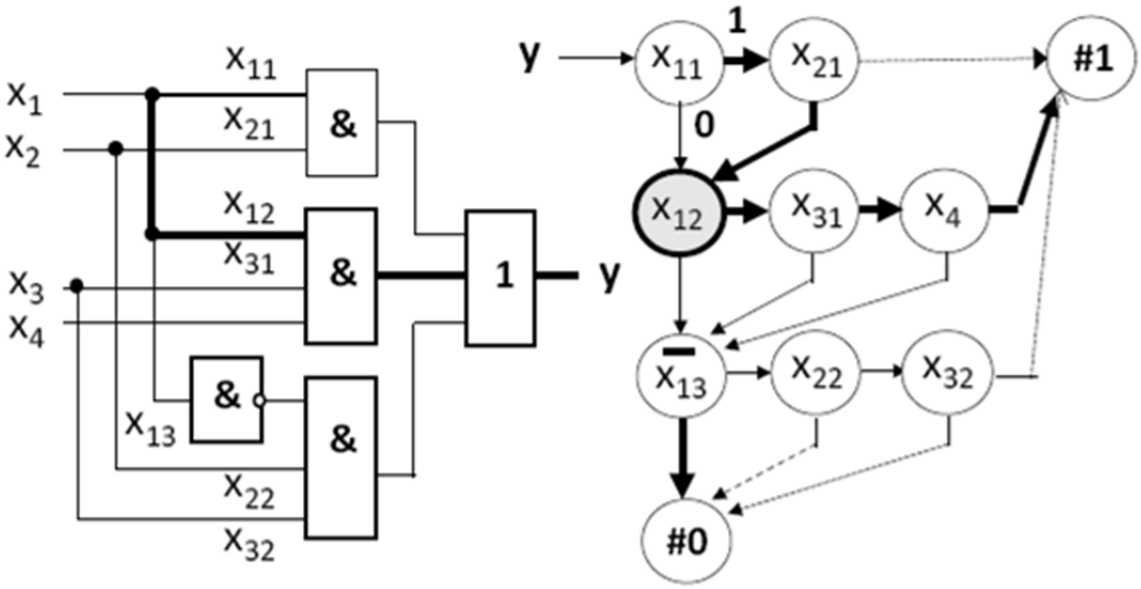

Fig. 1. Combinational circuit and its SSBDD

Example 1. In Fig. 1 we have a circuit with a FFR-module which can be represented by a Boolean expression:

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

and the SSBDD for the FFR part of the circuit. The literals with two indexes in the formula and in the SSBDD denote the branches of fan-out stems, and represent *signal paths* in the circuit. In this example, there are only two branches for each fan-out, the second index 1 is for the upper branch in the circuit, and the second index 2 is for the lower branch. For instance, the SSBDD node labeled by the variable x_u represents the bold signal path in the circuit. The variables in the nodes of SSBDD, in general, may be inverted. They are inverted when the number of invertors on the corresponding signal path in the circuit is odd. The two terminal nodes of the SSBDD are labeled by Boolean constants #1(truth) and #0(false). Let us agree that the right hand edges are always labeled by value 1, and lower hand edges by 0.

Every combinational circuit can be regarded as a network of modules, where each module represents an FFR of maximum size. 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. In Table 1, a comparison of numbers of nodes for representing ISCAS'85 circuits by ROBDD [20], FBDD [21], and SSBDD models is presented.

Table 1

Comparison of sizes of different BDDs

Circuit	#Gates	ROBDD	FBDD	SSBDD
c432	232	30200	1063	308
c499	618	49786	25866	601
c880	357	7655	3575	497
c1355	514	39858	N/A	809
c1908	718	12463	5103	866
c2670	997	N/A	1815	1313
c3540	1446	208947	21000	1648
c5315	1994	32193	1594	2712
c6288	2416	N/A	N/A	3872
c7552	2978	N/A	2092	3552

As a side effect of the synthesis of SSBDDs, we build up a strict one-to-one mapping between the nodes in SSBDDs and the signal paths in the modules (FFRs) of the circuit. The algorithm for synthesis of SSBDDs is presented in [18, 19]. The optimization issues of SSBDDs are discussed in [22]. Different properties of SSBDD which facilitate efficient processing of SSBDDs for test generation and fault diagnosis purposes are discussed in [23].

Since all the *stuck-at faults* (SAF) on the inputs of an FFR form a collapsed fault set of the FFR, and since all these faults are represented by the faults at the nodes of the related SSBDD, it follows that the synthesis of an SSBDD, described in [18, 19] is equivalent to the fault collapsing procedure similar to fault folding [24].

Direct relation of nodes to signal paths allows to handle with SSBDDs easily such problems like fault modeling, fault collapsing, and fault masking.

2. Fault reasoning using topology of S SBDD

Consider an FFR-module of a circuit which implements a function $y = f(X)$ where X is the set of input variables of the module, and is represented by SSBDD with a set of nodes M. Let $x(m) \in X$ the variable at the node $m \in M$, and let m^0 and m^1 be the neighbors of the node m for the assignments $x(m) = 0$ and $x(m) = 1$, respectively.

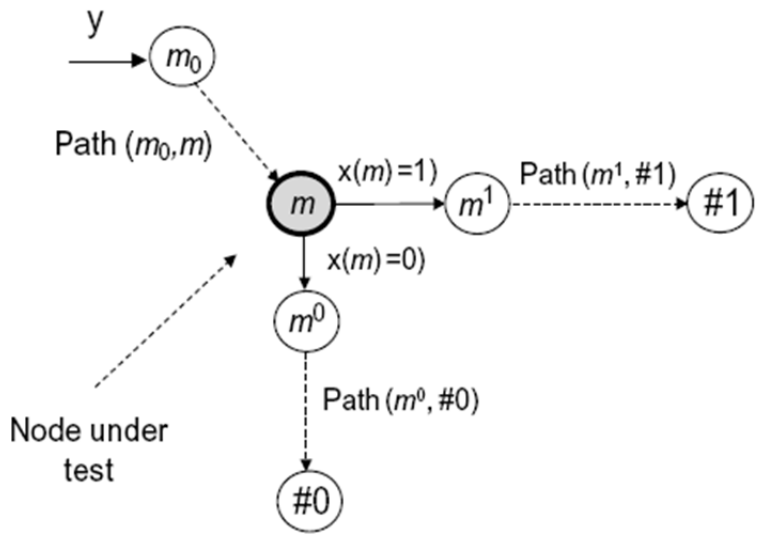

Fig. 2. Topological view on testing of nodes on the SSBDD

Activation of SSBDD paths. Let T_t be a pattern applied at the moment t on the inputs X of the module. The edge (m,m^e) in SSBDD, where $e \in \{0,1\}$, is called *activated* by T_t if $x(m) = e$. A path (m, n) is called activated by T_t if all the edges which form the path are activated. To activate a path (m, n) means to assign by T_t proper values to the node variables along this path.

Test generation with SSBDD. A test pattern T_t will detect a *single stuck-at-fault* (SSAF) $x(m) \equiv e$, $e \in \{0,1\}$, if it activates in the SSBDD three paths (Fig.2): a path (m_0,m) from the root node to the node under test, two paths $(m^0, \#0)$, $(m^1, \#1)$ for fault-free and faulty cases, and satisfies the fault activation condition $x(m) = e \otimes 1$.

Assume $e = 1$. To simulate the test experiment for T_t, generated for the fault $x(m) \equiv 1$, first, the path (m_0,m) will be traced up to the node m which will "serve as a switch". If the fault is missing, the path $(m^0, \#0)$ will be traced, and if the fault is present, the path $(m^1, \#1)$ will be activated.

Note, that a test pattern T_t for a node fault $x(m) \equiv e$ detects single SAFs on all the lines of the signal path in the circuit, which is represented by the node m in SSBDD.

Example 2. Consider the fault $x_{12} \equiv 0$ in the circuit of Fig.l, represented by the fault $x(m) = x_{12} \equiv 0$ in the SSBDD. To generate a test pattern T_t for $x_{12} \equiv 0$, we have to activate three paths: $(m_0,m) = (x_{11} = 1, x_{21} = 0)$, $(m^1, \#1) = (x_{11} = 1, x_{31} = 1, x_4 = 1, \#1)$, and $(m^0, \#0) = (x_{13} = 1, \#0)$. For the node under test we take $x_{12} = 1$ which means that the expected value of test will be $y = 1$. From this, the test pattern $X' = (x_1, x_2, x_3, x_4)' = 1011$ results. If the fault $x_{12} \equiv 0$ will be present, the path $(x_{11},x_{12},x_{13},\#0)$ will be activated, and the value #0 in the terminal node will indicate the presence of the fault.

Fault simulation and diagnosis with SSBDDs. For fault simulation of a test pattern X' on the SSBDD first, the path $l = l(m_0,m^T)$ from the root node m_0 to a terminal node $m^T \in \{\#0, \#1\}$, activated by the pattern X', is determined. Then, for each node $m \in l$, whose successor m^* does not belong to the path l, the path $l_m^* = l(m^*,m^{T*})$ will be simulated for the pattern X'. If $m^{T*} \neq m^T$ then the fault of the node m is detected by X', otherwise not. Special properties called "direction rule" allow to define the nodes $m \in l$, which can be excluded from analysis to speed up the simulation [25].

Example 3. Consider a test pattern $X' = (x_1 x_2, x_3, x_4)' = 1011$ which activates the path $l = (x_{11}, x_{21}, x_{12}, x_{31} x_4, \#1)$ shown bold in Fig. 1. According to the algorithm described, we can find that by the given test pattern the faults $x_{12} \equiv 0, x_{31} \equiv 0, x_4 \equiv 0$ can be detected.

The fault simulation procedure described can be used in fault diagnosis based on the effect-cause fault location concept to locate the fault candidates.

Fig. 3 illustrates average speed-up achieved by using SSBDDs for different simulation algorithms like logic simulation, fault simulation [26, 27], timing simulation [28] and multi-valued simulation [29] compared to gate-level simulation algorithms. The fault simulation shows the most noticeable acceleration. Other simulation algorithms vary in decreasing the runtime by 2,5 up to almost 4 times compared to algorithms working on the gate-level netlist model. This effect is possible due to shift from lower gate level to a higher macro level when working with SSBDD model (as macros, the FFRs are considered).

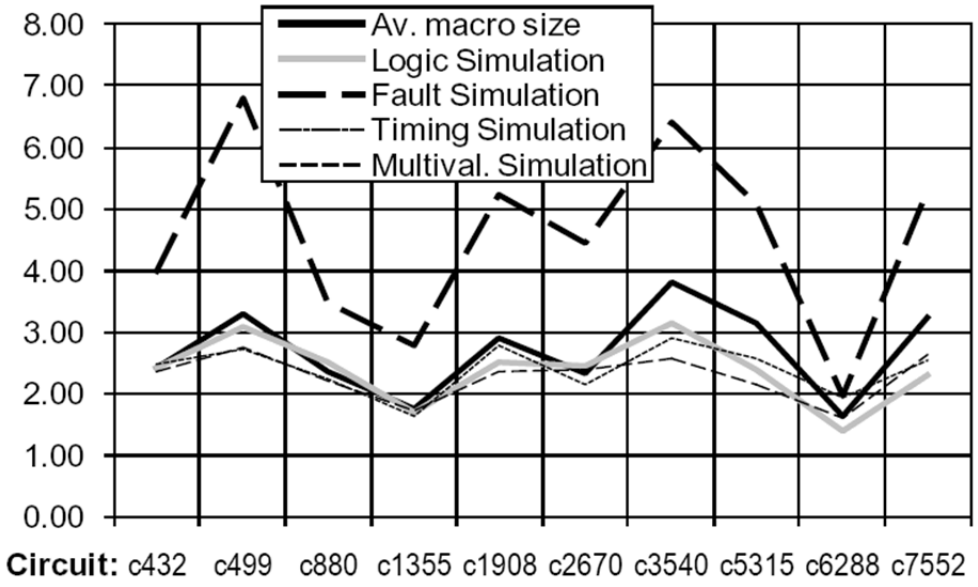

Fig. 3. Logic level simulation speed-up for different algorithms

SSBDDs facilitate well parallel processing of paths. This gave a possibility to develop extremely fast exact parallel critical path tracing algorithms which exceed the speed of commercial fault simulators, used in the industry [26, 27].

3. Multiple fault reasoning with SSBDDs

SSBDDs serve as an efficient tool for reasoning masking relationships in the presence of multiple faults. The basis of such a reasoning is the topological analysis of changes in the activation of paths because of faults.

A well known concept of test pairs [30–32] has been introduced for test generation which avoids fault masking in case of possible multiple faults. Let us call two test patterns TP = (T_0,T_1) a *test pair* for testing a node $x(m)$, where T_0 is for testing $x(m) \equiv 0$, T_1 is for testing $x(m) \equiv 1$, and both patterns differ from each other only in the value of $x(m)$.

Consider a topology of an SSBDD in Fig.4a with the highlighted root node m_0, two terminal nodes #0, #1, and two faulty nodes $a \equiv 0$, $c \equiv 1$. The dotted lines represent activated paths during a test pair TP = $\{T_0,T_1\}$which has the goal to test the node *a*. T_0 is for activating the correct path $L_1 = (m_0, a, \#1)$ for detecting the fault $a \equiv 0$ with expected test result #1. If the fault is present, instead of L_1, a "faulty" path $L_0 = (m_0, a \equiv 0, c, \#0)$ should be activated with faulty result #0. Thus, the fault $a \equiv 0$ should be detected.

In case of the masking fault $c \equiv 1$ on L_0, a masking path $L_M = (m_0, a \equiv 0, c \equiv 1, \#1)$ will be activated, and the fault under test $a \equiv 0$ will not be detected by T_0. The role of the second pattern T_1, according to the test pair concept, is to activate the path $L_0 = (m_0, a, \#0)$ with two goals: (1) to detect $a \equiv 1$ (in the single fault case), or (2) to detect the masking fault c $\equiv$ 1 (if the masking took place at T_0). At T_1 the path L_M should remain activated because of the masking fault $c \equiv 1$, and the wrong test result #1 will indicate the presence of a fault in the circuit.

The added value of a test pair is that if both patterns will pass then the wire under test (and the whole related signal path in the circuit) is proved to be correct at any present multiple fault.

Unfortunately, the described properties of the test pair will not always be sufficient for detecting multiple faults. The topological view on SSBDDs allows to develop efficient algorithms to avoid fault masking in multiple fault cases.

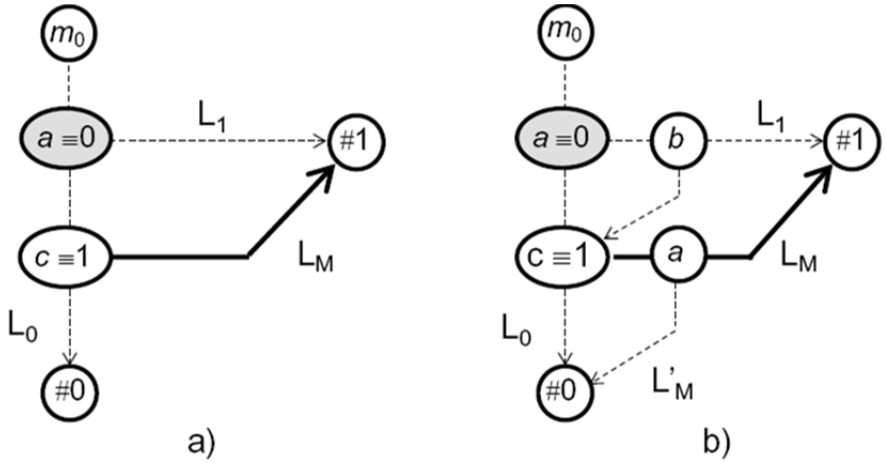

Fig. 4. Topological view on testing of nodes on the SSBDD

First, we show by simple reasoning of the topology of activated paths in SSBDDs why the test pairs are not always working as expected. Consider once more in Fig.4b the same test pair case TP = $\{T_0, T_t\}$ in the presence of the multiple fault $\{a \equiv 0, c \equiv 1\}$. Compared to Fig.4a, we have now additional node a on the masking path L_M, labeled by the same variable as the node under test $a \equiv 0$ (in grey). T_0 will again show the correct value #1 because of fault masking. At T_1 the value of a was changed from 1 to 0 compared to T_0. Because of this change, instead of the masking path $L_M = (m_0, a \equiv 0, c \equiv 1, \#1)$, a new *"demasking"* path $L'_M = (m_0, a \equiv 0, c \equiv 1, a, \#0)$ will be activated. Hence, both patterns will pass, and the multiple fault remains undetected.

The main idea of the test pair concept is to keep the activation of the masking path stable during both patterns. In Fig.4b, this condition is not fulfilled, since all the three paths, L_1 for correct case, L_0 for single fault case ($a \equiv 0$), and L_M for the multiple fault case ($a \equiv 0$, c v 1), involve the variable a under test, and the changing value of a will make the activation of the masking path L_M unstable.

In [33], a new method of test groups was developed and discussed which extends the method of test pairs and removes its drawbacks. The method of test groups essentially is based on the topological analysis of SSBDDs.

Example 4. Let us add, as an example, to TP a third pattern T_2 for testing the node b (for $b \equiv 1$) on L_1. The goal of T_2 is to keep the masking path $L_M = (m_0, a = 1, b = 0, c \equiv 1, a=1, \#1)$ again activated. The three patterns T_0 (for testing $a \equiv 0$ and $b \equiv 0$), T_2 (for testing $a \equiv 1$) and T_3 (for testing $b \equiv 1$) can be regarded as a *test group* for testing two nodes a and b. In this example, the first two patterns will pass, however, the pattern T_2 will not pass, and detect the given multiple fault.

The test groups are targeting not a single SSBDD node, rather a selected subset of SSBDD nodes [33] which represent a subcircuit of the given gate-level network. The main property and the main importance of test groups is that they are robust with respect to multiple faults. In other words, from passing of a given test group the correctness of the related subcircuit results. The test pair can be considered as a special case of the test group which works in "special" cases.

4. Shared *SSBDDs*

In [34], a new type of SSBDDs called *Structurally Synthesized Multiple Input BDDs* or Shared SSBDDs (SSSBDD) was introduced. The goal was to further compress the SSBDD model by exploiting the effect of superposition of SSBDDs.

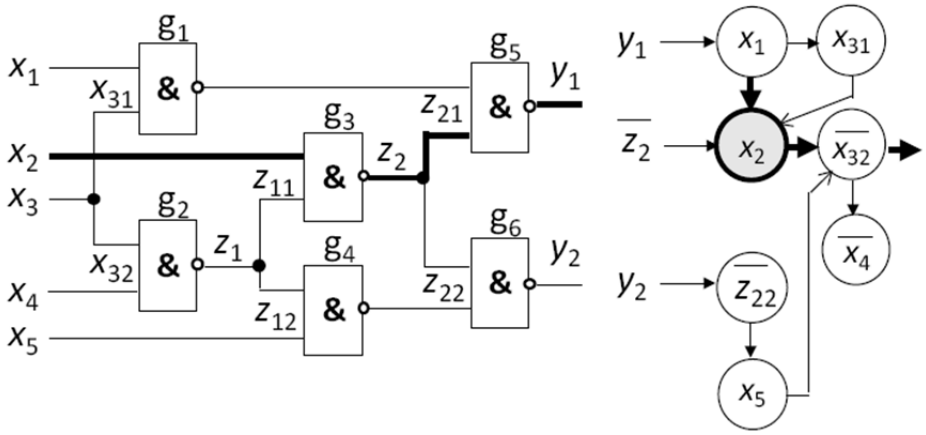

Fig. 5. Digital circuit c17 and its SSSBDD

Example 5. An example of a combinational circuit and its SSSBDD is presented in Fig.5. For simplicity we omit the terminal nodes, and agree that leaving the graph to the right (down) means entering the terminal node #1 (#0). The graph with 7 nodes represents only 14 collapsed stuck-at faults as targets for test generation instead of 32 faults in the lines of the original gate-level circuit. The graph joins three subgraphs for outputs y_x and y_2, and for internal node z_2 with the shared subgraph with nodes $\neg x_{32}$ and $\neg x_4$. The nodes x_1 and x_5 represent signal paths in the circuit from x_1 to y_1 and from x_5 to y_2, respectively. The node x_{31} represents the path from the lower input of *gi* to y_x. The node x_2 represents the path from the upper input of g_1 to y_1. The nodes $\neg x_{32}$ and $\neg x_4$ represent the paths from the inputs of g_2 to both outputs y_1 and y_2. And, finally, the node $\neg x_{22}$ represents the path from the upper input of g_6 to y_2. By bold lines in SSSBDD a path is shown, which leads to assignments $x_1 = 0$ and $x_3 = 0$ for testing the bold path in the circuit from x_2 to y_1. More detailed discussion of using SSSBDDs where the graphs for different output functions are merged into the same graph can be found in [34].

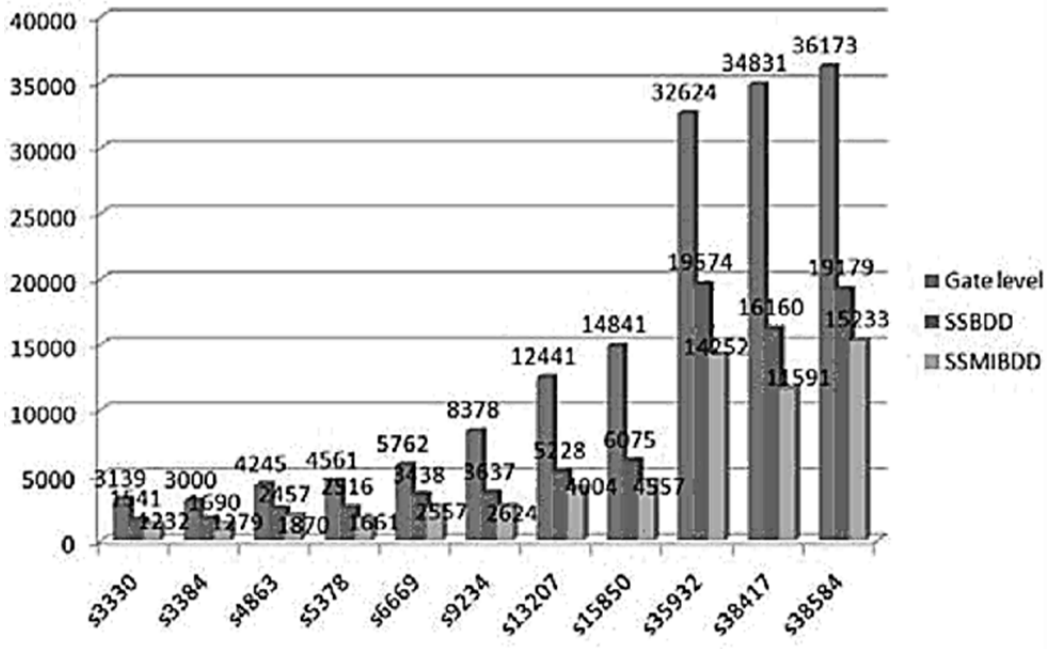

Fig. 6. Comparison of complexities of SSBDDs and SSSBDDs

A comparison of the reduction of complexities for gate-level circuits, SSBDDs and SSSBDDs in terms of fault collapsing was carried out for ISCAS'89 circuits. The differences in the number of nodes are shown

in Fig. 6. The average minimization gained for SSSBDDs in the number of nodes (and also in the size of collapsed fault sets) is up to 2.4 times compared to the gate level, and up to 1.4 times compared to the SSBDD model. The results prove that the SSSBDD model is more compact than the previously discussed SSBDD or gate level models, and as the result allows better fault collapsing which in its turn has the influence on the efficiency and speed of test generation and fault simulation.

5. Overview of high level DDs

The most important impact of the high-level DDs (HLDD) is the possibility of generalization and extension of the methods for test generation, fault simulation and diagnosis, developed for logic level circuits, to higher abstraction levels of digital systems using the uniform graph topology based formalism. For this purpose, the class of variables was extended from Boolean ones to Boolean vectors or integer variables, and the class of Boolean functions was extended to the data manipulation operations typically used in high-level descriptions of digital systems.

Example 6. In Fig. 7, 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, respectively. 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 non-terminal 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 as HLDDs.

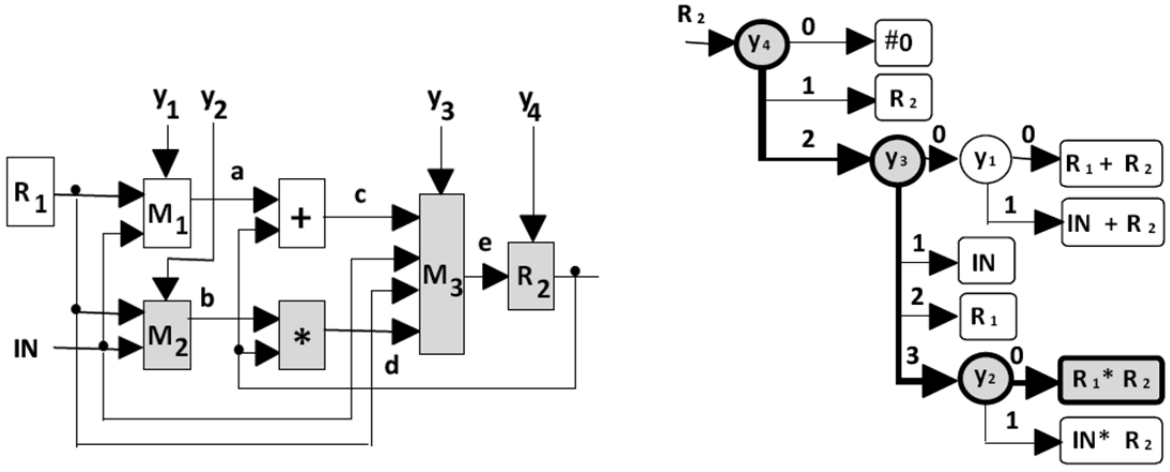

Fig. 7. Representing a register transfer level data path by a HLDD

Two methods for synthesis of HLDDs for representing digital systems were described in [19, 35]. The first one 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 [19].

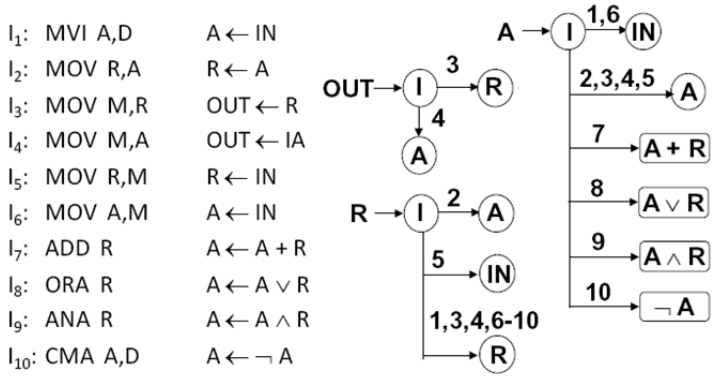

Fig. 8. HLDDs for a hypothetical microprocessor on the ISA level

Example 7. An example of behavior level HLDDs is shown in Fig. 8 for representing a hypothetical microprocessor given at the Instruction Set Architecture (ISA) level by its instruction set list. The model consists of three DDs: OUT, A, and R for representing the processor's output behavior, accumulator A, and register R, respectively.

6. Diagnostic modeling of digital systems with HLDDs

The methods for test generation and fault simulation developed for SSBDDs can be easily generalized for using at higher abstraction levels of systems [36]. The possibility of generalization results from the topological similarity of DDs at lower and higher levels (Fig. 9). In case of SSBDDs, each node has two output edges, and the graph has two terminal nodes $m^{T,0}$ and $m^{T,1}$ with constants 0 and 1, respectively. HLDDs differ from SSBDDs in having more edges from nodes and more terminal nodes $m^{T,1}$, $m^{T,2}$, ..., $m^{T,n}$, whereas the terminal nodes in general case may be labeled by constants, register variables or functional expressions. Both graphs represent a mapping into the structure of the system they describe.

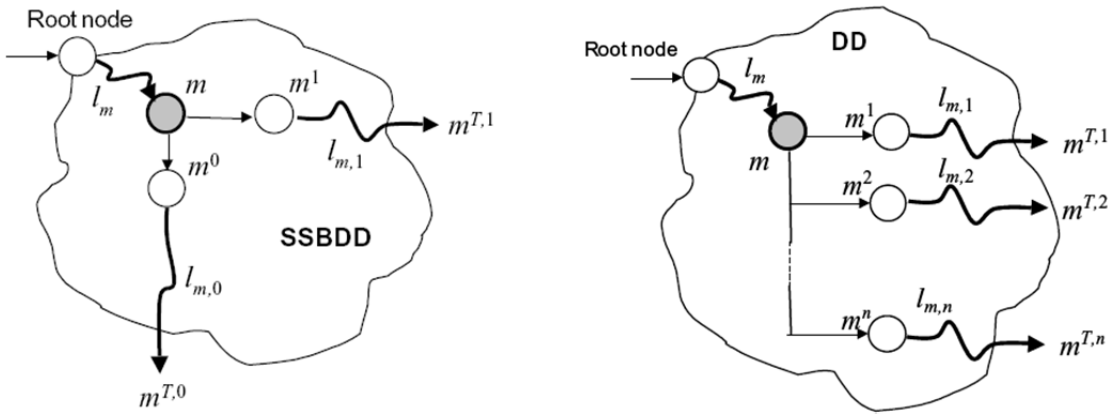

Fig. 9. Topological similarities of SSBDDs and HLDDs

In both cases, the faults in the system can be modeled similarly by errors at the nodes, and for both types of graphs, test generation for a given node те is carried out by activating a path from the root node to те and

from all successor nodes of те to correponding terminal nodes. It is easy to see that the SSBDD can be regarded as a special case of HLDD. Similarly, as we defined the operations of logic simulation and path activation for SSBDDs we can do the same for HLDDs.

Example 8. In test pattern simulation, a path is traced in the graph, guided by the values of system variables until a terminal node is reached, similarly as in the case of SSBDDs. For example, in Fig.7, the result of simulating the vector $X' = (y_1, y_2, y_3, y_4, R_1, R_2, IN) =$ -,0,3,2,10,6,- is $R_2 = R_1 * R_2 = 60$ (here "-" means *don't care,* the bold arrows in Fig.7 highlight the simulated path, and the grey node $R_1 * R_2$ is reached by simulation).

The advantage of HLDDs compared to the traditional methods of simulation of systems, lays in the fact that instead of processing of all the components in the RTL network for the given input pattern, in the HLDD only 3 control variables y_4, y_3, y_2, were visited in this particular case during simulation, and only a single data manipulation operation $R_2 = R_1 * R_2$ was carried out.

Fault model on HLDDs. Each path in the HLDD describes the behavior of the system in a specific working mode. The faults having effect on the behavior can be associated with nodes along the path. A fault causes incorrect leaving the path activated by a test. From this point of view the following abstract fault model for nodes те with node variables $x(m)$ in HLDDs can be defined:

Dl: the output edge for $x(m) = i$ of a node m is always activated, $x(m) = i$ (analog to logic level stuck-at-1);

D2: the output edge for $x(m) = i$ of a node m is broken (analog to logic level stuck-at-0);

D3: instead of the given edge for $x(m) = i$ of a node m, another edge for $x(m) = j$, or a set of edges $\{j\}$ is activated (analog to logic level multiple stuck-at-fault).

The fault model is directly related to the nodes m , and is an abstract one. It will have a semantic meaning only when the node has a particular physical interpretation. As an example, in Table 2 the correspondence of the HLDD-based fault model to different microprocessor fault classes [37], and RTL fault classes [38, 39] is shown.

Table 2

Comparison of High-Level fault models

Microprocessor faults [37]	**DD faults**	
F1: No source is selected F4: No destination selected	D1, D2	Internal nodes
F3: More than one source is selected F5: Instead of, or in addition to the selected destination, one or more other destinations are selected	D3	Internal nodes
F2: A wrong source is selected; F5: One or more other destinations are selected	D3	Internal nodes
F9-F14: Data storage, communication or manipulation faults	D3	Terminal nodes
RTL faults [38,39]	**DD faults**	
F15-F20: Control, addressing, timing, condition, decoding faults)	D1,D2,D3	Internal nodes
F21-F23: Data storage, communication or manipulation faults	D3	Terminal nodes

Test generation. 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 an internal node те in HLDD, $(n + 1)$ paths are to be activated: first, a path (m_0, m), and second, n paths $l_e = (m^e, m^{T,e})$ for all values e of the variable $x(m)$, so that

$$x(m^{T,1}) \neq x(m^{T,2}) \neq \ldots \neq x(m^{T,n}).$$

All paths should be activated consistently by the same test pattern (or sequence) T'. The test T' includes as well the data found by solving the inequality.

The test program for an internal node m (*conformity test* of the control part), consists of n experiments to excercise all the possible n values of $x(m)$ [40].

To test a terminal node $m^{T,i}$ a path $(m_0, m^{T,i})$ is activated. The test program *(scanning test)* generated for $m^{T,1}$ will be repeated for all local test patterns for testing the module with function $x(m^{T,i})$. The local test patterns may be generated at lower level using for instance SSBDDs [40].

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. 7. 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 test data $R_1 = D_1$, $R_2 = D_2$, $IN = D_3$ are found by satisfying the inequality $R_1 + R_2 \neq IN \neq R_1 \neq R_1 * R_2$.

Note, by the described procedure, a test pattern is created for earring out the test for a selected HLDD node (a structural unit of a system) at the given state of the system (i.e. content of the system registers). In the full test sequence, the needed load operations as well as the operations for reading out the test result should be included. These operations can be formally generated as well from the HLDD model of the system [19].

From above, the following test program results:

Test program for control part:

Fore = 1,2,3,4

BEGIN

Load the data registers:

$R_1 = D_1, R_2 = D_2$;

Carry out the tested working mode at:

$y_3 = e, y_1 = 0, y_2 = 0, y_4 = 2$ and $IN = D_3$;

Read the test response $R_{2:e}$

END.

Example 10. As another example, consider test generation for testing the multiplier M_3 represented by the node $R_1 * R_2$ in Fig. 7. By activating the path to this node (shown in bold in Fig. 7) we generate a control word $(y_2, y_3, y_4) = (0, 3, 2)$. To find the proper values of R_1 and R_2 we need to descend to the lower abstraction level of hierarchy (e.g. to the gate level) and generate test patterns by a low level ATPG for the low level implementation of the multiplier. Let us have got a low level test set of n data patterns ($D_{1,1}$, $D_{2,1}$; $D_{1,2}$, $D_{2,2}$; $D_{1,1}$, …, $D_{1,n}$, $D_{2,n}$,) generated for the multiplier with input registers R_1 and R_2.

From above, the following test program results:

Test program for data part:

For all the values of $t = 1,2,..., n$

BEGIN

Load the data registers:

$R_1 = D_{1,t}$, $R_2 = D_{2,t}$,

Carry out the tested working mode at the

control values $(y_2, y_2, y_4) = (0,3,2)$;

Read the test response of $R_{2:t}$

END.

HLDDs have been used in different fields of high-level and hierarchical test and verification. As the result, new promising algorithms, techniques and prototype tools have been developed, which allowed to improve the efficiency of RTL cycle based simulation [41, 42], hierarchical test program automated synthesis [40, 43], hierarchical fault simulation [44], high-level verification [46], fault diagnosis [47, 48], and automated design error correction [48, 49].

Conclusion

An overview was given about two types of Decision Diagrams - SSBDDs and HLDDs for diagnostic modeling of digital systems, particularly for fault simulation and test generation. 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.

Acknowledgment: The work has been supported by FP7 1ST project DIAMOND, and Research Centre CEBE funded by EU Structural Funds.

REFERENCES

1. *Lee C.Y.* Representation of Switching Circuits by Binary Decision Programs // The Bell System Technical Journal. 1959. P. 985-999.
2. *Ubar R.* Test Generation for Digital Circuits with Alternative Graphs // Proceedings of Tallinn Technical University. 1976. No. 409. P. 75-81.
3. *Akers S.B.* Functional Testing with Binary Decision Diagrams // J. of Design Automation and Fault-Tolerant Computing. 1978. V. 2. P. 311-331.
4. *Bryant R.E.* Graph-based algorithms for Boolean function manipulation // IEEE Trans on Comp. 1986. V. C-35, No. 8. P. 667-690.
5. *Sasao T., Fujita M. (eds.).* Representations of Discrete Functions. Kluwer Academic Publishers. 1996.
6. *Drechsler R., Becker B.* Binary Decision Diagrams. Kluwer Academic Publ., 1998.
7. *Minato S., Ishiura N.* Shared binary decision diagrams with attributed edges for efficient Boolean function manipulation // Proc. 27th IEEE/ACM ICCAD. 1990. P. 52-57.
8. *Srinivasan A., Kam T., Malik S., Bryant R.E.* Algorithms for discrete function manipulation // Proc. of Informations Conference on CAD –ICCAD. 1990. P. 92-95.
9. *Kebschull U., Schubert E., Rosenstiel W.* Multilevel logic synthesis based on functional decision diagrams. IEEE EDAC, 1992.
10. *Minato S.* Zero-suppressed BDDs for set manipulation in combinational problems // Proc. 30th DAC. 1995. P. 272-277.
11. *Bahar R., Frohm E., Gaona C., Hachtel G., Macii E., Pardo A., Somenzi F.* Algebraic decision diagrams and their applications // Int. Conf. on CAD. 1993. P. 188-191.
12. *Drechsler R., Sarabi A., Theobald M., Becker B., Perkowski M.A.* Efficient representation and manipulation of switching functions based on Ordered Kronecker Functional Decision Diagrams // Proc. DAC. 1994.
13. *Bryant R.E., Chen Y.-A.* Verification of arithmetic functions with binary moment diagrams // 32nd ACM/IEEE DAC. 1995.
14. *Bern J., Meinel C., Slobodova A.* Efficient OBDD-based manipulation in CAD beyond current limits // 32-nd DAC. 1995. P. 408-413.
15. *Clarke E., Fujita N., Zhao X.* Multi-terminal binary decision diagrams and hybrid decision diagrams, In: T. Sasao, M. Fujita (eds.), Representations of Discrete Functions. Kluwer Academic Publishers, 1996. P. 93-108.
16. *Stankovic R., Astola J., Stankovic M., Egiazarjan K.* Circuit synthesis from Fibonacci decision diagrams // VLSI Design, Special Issue on Spectral Techniques and Decision Diagrams. 2002. V. 14. P. 23-34.
17. *Karpovsky M.G., Stankovic R.S., Astola J.T.* Spectral Logic and Its Applications for the Design of Digital Devices. Wiley-Interscience, 2008.
18. *Ubar R.* Test Synthesis with Alternative Graphs // IEEE Design&Test of Computers. Spring, 1996. P. 48-57.
19. *Ubar R., Raik J., AJutman, Jenihhin M.* Diagnostic Modeling of Digital Systems with Multi-Level DDs // Design and Test Technology for Dependable SoC, R. Ubar, J. Raik, H.Th. Vierhaus (Eds.). 2011. P. 92-118.
20. *Brace K.S., Rudell R.L., Bryant R.E.* Efficient Implementation of a BDD Package // Proc. of the 27th DAC. June 1990. P. 40-45.
21. *Giinther W., Drechsler R.* Minimization of Free BDDs // Proc. of Asia and South Pacific Design Automation Conf. Hong Kong, Jan 1999. P. 323-326.
22. *Ubar R., Vassiljeva T., Raik J., AJutman, Tombak M., Peder A.* Optimization of Structurally Synthesized BDDs // IASTED Conf. on Modelling, Simulation and Optimization, Kauai, Hawaii, USA, August 17-19. 2004. P. 234-240.
23. *Peder A., Nestra H., Raik J., Tombak M., Ubar R.* Linear algorithms for testing and parsing superpositional graphs // Facta Universitatis (Nis) Ser.: Elec. Energ. 2011. V. 24, No. 3. P. 325-339.
24. *To K.* Fault Folding for Irredundant and Redundant Combinational Circuits // IEEE Trans, on Computers. 1973. No. 11. P. 1008-1015.
25. *Ubar R.* Overview about Low-Level and High-Level Decision Diagrams for Diagnostic Modeling of Digital Systems // Facta Universitatis (Nis) Ser.: Elec. Energ. 2011. V. 24, No. 3. P. 303-324.
26. *Ubar R., Devadze S., Raik J., AJutman.* Fast Fault Simulation in Digital Circuits with Scan Path // IEEE Proc. of 13th Asia and South Pacific Design Automation Conference - ASP-DAC. 2008. P. 667-672.
27. *Ubar R., Devadze S., Raik J., AJutman.* Parallel X-Fault Simulation with Critical Path Tracing Technique // IEEE Conf. Design, Automation & Test in Europe – DATE. 2010. P. 1-6.
28. *AJutman, Ubar R., Peng Z.* Algorithms for Speeding-Up Timing Simulation of Digital Circuits // DATE. 2001. P. 460-465.

29. *Ubar R.* Multi-Valued Simulation of Digital Circuits with Structurally Synthesized Binary Decision Diagrams. OPA, N.V. Gordon and Breach Publishers, Multiple Valued Logic. 1998. V. 4. P. 141-157.
30. *Cox H., Rajski J.* A Method of Fault Analysis for Test Generation and Fault Diagnosis // IEEE Trans. on CAD. 1988. V. 7. No. 7. P. 813-833.
31. *Kajihara S., Nishigaya R., Sumioka T., Kinoshita K.* Efficient Techniques for Multiple Fault Test Generation // 3rd ATS. 1994. P. 52-56.
32. *Agrawal A., Saldanha A., Lavagno L.* Compact and Complete Test Set Generation for Multiple Stuck-at Faults // ICCAD'93. 1996. P. 212-219.
33. *Ubar R., Kostin S., Raik J.* Multiple Stuck-at-Fault Detection Theorem // The 15th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems – DDECS. Tallinn, Estonia, April 18-20. 2012.
34. *Ubar R., Mironov D., Raik J., AJutman.* Structural Fault Collapsing by Superposition of BDDs for Test Generation // ISQED. 2010. P. 250-257.
35. *Ubar R., Raik J., Karputkin A., Tombak A.* Synthesis of High-Level Decision Diagrams for Functional Test Pattern Generation // 16th Int. Conference MIXDES. 2009. P. 519-524.
36. *Raik J., Ubar R.* Fast Test Pattern Generation for Sequential Circuits Using Decision Diagram Representations. Journal of Electronic Testing: Theory and Applications. Kluwer Academic Publishers. 2000. V. 16, No. 3. P. 213-226.
37. *Thatte S.M., Abraham J.A.* Test Generation for Microprocessors // IEEE Trans. On Сотр. 1980. V. C-29, No. 6. P. 429-441.
38. *Gupta A.K., Armstrong J.R.* Functional Fault modelling and Simulation for VLSI Devices // 22nd Design Automation Conference. 1985. P. 720-726.
39. *Ubar R., Raik J., AJutman, Instenberg M., Wuttke H.-D.* Modeling Microprocessor Faults on High-Level Decision Diagrams. International Conference on Dependable Systems & Networks: Anchorage, USA, Alaska, June 24-27. 2008. P. 17-22.
40. *Jervan G., Ubar R., Peng Z., Eles P.* Test Generation: A Hierarchical Approach // System-level Test and Validation of HW/SW Systems. by M.Sonza Reorda, Z.Peng, M.Violante. Springer Series in Advanced Microelectronics. 2005. V. 17. P. 63-77.
41. *Xeveugle R., Ubar R.* Modeling VHDL Clock-Driven Multi-Processes by Decision Diagrams // J. of Electron Technology. 1999. V. 32, No. 3. P. 282-287.
42. *Ubar R., Morawiec A., Raik J.* Cycle-Based Simulation Algorithms for Digital Systems Using High-Level Decision Diagrams // DATE. 2000. P. 743.
43. *Raik J., Ubar R.* Fast Test Pattern Generation for Sequential Circuits Using Decision Diagram Representations. Journal of Electronic Testing: Theory and Applications. Kluwer Academic Publishers. 2000. V. 16, No. 3. P. 213-226.
44. *Ubar R., Devadze S., Jenihhin M., Raik J., Jervan G., Ellervee P.* Hierarchical Calculation of Malicious Faults for Evaluating the Fault-Tolerance // 4th IEEE DELTA. 2008. P. 222-227.
45. *Jenihhin M., Raik J., Fujiwara H., Ubar R., Viilukas T.* An Approach for Verification Assertions Reuse in RTL Test Pattern Generation // Proc. of the IEEE 11th Workshop on RTL and High Level Testing -WRTLT'10. 2010. P. l-6.
46. *Raik J., Repinski U., Ubar R., Jenihhin M., Chepurov A.* High-Level Design Error Diagnosis Using Backtrace on Decision Diagrams // The 28th IEEE NORCHIP Conference, Tampere. Nov. 15-16. 2010.
47. *Raik J., Repinski U., Ubar R., Jenihhin M., Chepurov A.* High-Level Design Error Diagnosis Using Backtrace on Decision Diagrams // The 28th IEEE NORCHIP Conference, Tampere. Nov. 15-16. 2010.
48. *Guarnieri V., DiGuglielmo G., Bombieri N., Pravadelli G., Fummi F., Hantson H., Raik J., Jenihhin M., Ubar R.* On the Reuse of TLM Mutation Analysis at RTL // J. Electron Test, DOI 10.1007/sl0836-012-5303-6. 2012.
49. *Karputkin A., Ubar R., Tombak M., Raik J.* Automated Correction of Design Errors by Edge Redirection on High-Level Decision Diagrams // IEEE International Symposium on Quality Electronic Design - ISQED, Santa Clara, CA USA, 2012.

Убар Раймонд, д-р техн. наук, профессор. E-mail: raiub@pld.ttu.ee
Таллиннский технический университет (Эстония)

Поступила в редакцию 3 февраля 2014 г.

Убар Раймонд. (Таллиннский технический университет, Эстония)

Влияние неисправностей цифровых систем на топологию решающих диаграмм низкого и высокого уровней.

Ключевые слова*:* решающие диаграммы; генерация тестов; диагностика неисправностей.

Для тестирования сложных цифровых систем необходимо исследовать их иерархические многоуровневые представления. Рассматриваются Structurally Synthesized Binary Decision Diagram (SSBDD) решающие диаграммы, обеспечивающие соответствие между полюсами таких диаграмм и вентилями цифровых систем. Методы синтеза тестов и диагностики неисправностей, разработанные на основе использования SSBDD решающих диаграмм, развиваются для описаний цифровых систем на более высоком уровне абстракции. Речь идет об использовании High Level Decision Diagram (HLDD) решающих диаграмм для автоматической генерации тестов и диагностики неисправностей сложных цифровых систем.

REFERENCES

1. Lee C.Y. Representation of switching circuits by binary decision programs. *The Bell System Technical Journal,* 1959, pp. 985-999. DOI: 10.1002/j.1538-7305.1959.tb01585.x.

2. Ubar R. Test generation for digital circuits with alternative graphs. *Proceedings of Tallinn Technical University,* 1976, no. 409, pp.75-81. DOI: 10.1007/3-540-58426-9_129.
3. Akers S.B. Functional testing with binary decision diagrams. *J. of Design Automation and Fault-Tolerant Computing,* 1978, vol. 2, pp. 311-331. DOI: 10.1109/TC.1986.1676774.
4. Bryant R.E. Graph-based algorithms for Boolean function manipulation. *IEEE Trans on Comp*, 1986, vol. C-35, no. 8, pp. 667-690. DOI: 10.1109/TC.1986.1676819.
5. Sasao T., Fujita M. (eds.). *Representations of discrete functions*. Kluwer Academic Publ., 1996. 331 p.
6. Drechsler R., Becker B. *Binary decision diagrams*. Kluwer Academic Publ., 1998. 200 p.
7. Minato S., Ishiura N. Shared binary decision diagrams with attributed edges for efficient Boolean function manipulation. *Proc. 27th IEEE/ACM ICCAD*, 1990, pp. 52-57. DOI: 10.1145/123186.123225.
8. Srinivasan A., Kam T., Malik S., Bryant R.E. Algorithms for discrete function manipulation. *Proc. of Informations Conference on CAD -ICCAD*, 1990, pp. 92-95. DOI: 10.1109/ICCAD.1990.129849.
9. Kebschull U., Schubert E., Rosenstiel W. Multilevel logic synthesis based on functional decision diagrams. *IEEE EDAC*, 1992.
10. Minato S. Zero-suppressed BDDs for set manipulation in combinational problems. *Proc. 30th DAC,* 1995, pp. 272-277. DOI: 10.1145/157485.164890.
11. Bahar R., Frohm E., Gaona C., Hachtel G., Macii E., Pardo A., Somenzi F. Algebraic decision diagrams and their applications. *Int. Conf. on CAD,* 1993, pp. 188-191. DOI: 10.1109/ICCAD.1993.580054.
12. Drechsler R., Sarabi A., Theobald M., Becker B., Perkowski M.A. Efficient representation and manipulation of switching functions based on Ordered Kronecker Functional Decision Diagrams. *Proc. DAC*, 1994. DOI: 10.1145/196244.196444.
13. Bryant R.E., Chen Y.-A. Verification of arithmetic functions with binary moment diagrams. *32nd ACM/IEEE DAC*, 1995. DOI: 10.1145/217474.217583.
14. Bern J., Meinel C., Slobodova A. Efficient OBDD-based manipulation in CAD beyond current limits. *32-nd DAC*, 1995, pp. 408-413. DOI: 10.1145/217474.217563.
15. Clarke E., Fujita N., Zhao X. *Multi-terminal binary decision diagrams and hybrid decision diagrams*. In: Sasao T., Fujita M. (eds.) *Representations of discrete functions*. Kluwer Academic Publishers, 1996, pp. 93-108.
16. Stankovic R., Astola J., Stankovic M., Egiazarjan K. Circuit synthesis from Fibonacci decision diagrams. *VLSI Design, Special Issue on Spectral Techniques and Decision Diagrams*, 2002, vol.14, pp. 23-34. DOI: 10.1080/10655140290009783.
17. Karpovsky M.G., Stankovic R.S., Astola J.T. *Spectral logic and its applications for the design of digital devices*. Wiley-Interscience, 2008. 500 p.
18. Ubar R. Test synthesis with alternative graphs. *IEEE Design&Test of Computers*, Spring, 1996, pp. 48-57. DOI: 10.1109/54.485782.
19. Ubar R., Raik J., Jutman A., Jenihhin M. *Diagnostic Modeling of Digital Systems with Multi-Level DDs*. In: Ubar R., Raik J., Vierhaus H.Th. (eds.) *Design and test technology for dependable SoC*. 2011, pp. 92-118.
20. Brace K.S., Rudell R.L., Bryant R.E. Efficient implementation of a BDD Package. *Proc. of the 27th DAC,* June 1990, pp. 40-45. DOI: 10.1145/123186.123222.
21. Giinther W., Drechsler R. Minimization of Free BDDs. *Proc. of Asia and South Pacific Design Automation Conf, Hong Kong*, Jan 1999, pp. 323-326. DOI: 10.1016/S0167-9260(02)00041-X.
22. Ubar R., Vassiljeva T., Raik J., Jutman A., Tombak M., Peder A. Optimization of structurally synthesized BDDs. *IASTED Conf. on Modelling, Simulation and Optimization*, Kauai, Hawaii, USA, August 17-19, 2004, pp. 234-240.
23. Peder A., Nestra H., Raik J., Tombak M., Ubar R. Linear algorithms for testing and parsing superpositional graphs. *Facta Universitatis (Nis) Ser.: Elec. Energ,* 2011, vol. 24, no. 3, pp. 325-339. DOI: 10.2298/FUEE1103325P.
24. To K. Fault folding for irredundant and redundant combinational circuits. *IEEE Transactions on Computers*, 1973, no.11, pp. 1008-1015. DOI: 10.1109/T-C.1973.223637.
25. Ubar R. Overview about low-level and high-level decision diagrams for diagnostic modeling of digital systems. *Facta Universitatis (Nis) Ser.: Elec. Energ.,* 2011, vol. 24, no. 3, pp. 303-324. DOI: 10.2298/FUEE1103303U.
26. Ubar R., Devadze S., Raik J., Jutman A. Fast fault simulation in digital circuits with scan path. *IEEE Proc. of 13th Asia and South Pacific Design Automation Conference - ASP-DAC*, 2008, pp. 667-672.
27. 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, pp. 1-6. DOI: 10.1109/DATE.2010.5456929.
28. Jutman A., Ubar R., Peng Z. Algorithms for speeding-up timing simulation of digital circuits. *DATE*, 2001, pp. 460-465.
29. Ubar R. Multi-valued simulation of digital circuits with structurally synthesized binary decision diagrams. *OPA, N.V. Gordon and Breach Publishers, Multiple Valued Logic*, 1998, vol. 4, pp. 141-157.
30. Cox H., Rajski J. A method of fault analysis for test generation and fault diagnosis. *IEEE Transactions on CAD*, 1988, vol. 7, no. 7, pp. 813-833. DOI: 10.1109/43.3952.
31. Kajihara S., Nishigaya R., Sumioka T., Kinoshita K. Efficient techniques for multiple fault test generation. *3rd ATS*, 1994, pp. 52-56. DOI: 10.1109/ATS.1994.367254.
32. Agrawal A., Saldanha A., Lavagno L. Compact and complete test set generation for multiple stuck-at faults. *ICCAD'93*, 1996, pp. 212-219.
33. Ubar R., Kostin S., Raik J. Multiple stuck-at-fault detection theorem. *The 15th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems - DDECS*, Tallinn, Estonia, April 18-20, 2012. DOI: 10.1109/DDECS.2012.6219064

34. Ubar R., Mironov D., Raik J., Jutman A. Structural fault collapsing by superposition of BDDs for test generation. *ISQED*, 2010, pp. 250-257. DOI: 10.1109/ISQED.2010.5450451
35. Ubar R., Raik J., Karputkin A., Tombak A. Synthesis of high-level decision diagrams for functional test pattern generation. *16th Int. Conference MIXDES*, 2009, pp. 519-524.
36. Raik J., Ubar R. Fast test pattern generation for sequential circuits using decision diagram representations. *Journal of Electronic Testing: Theory and Applications,* 2000, vol. 16, no. 3, pp. 213-226. DOI: 10.1023/A:1008335130158.
37. Thatte S.M., Abraham J.A. Test generation for microprocessors. *IEEE Transactions On Comp.*, 1980, vol. C-29, no. 6, pp. 429-441. DOI: 10.1109/TC.1980.1675602.
38. Gupta A.K., Armstrong J.R. Functional fault modelling and simulation for VLSI Devices. *22nd Design Automation Conference*, 1985. pp.720-726. DOI: 10.1109/DAC.1985.1586022
39. Ubar R., Raik J., Jutman A., Instenberg M., Wuttke H.-D. Modeling microprocessor faults on high-level decision diagrams. *International Conference on Dependable Systems & Networks: Anchorage*, USA, Alaska, June 24-27, 2008, pp. 17-22.
40. Jervan G., Ubar R., Peng Z., Eles P. *Test generation: A hierarchical approach*. In: Sonza Reorda M., Peng Z., Violante M. (eds.) *System-level test and validation of HW/SW systems*. Springer Series in Advanced Microelectronics, 2005, vol. 17, pp. 63-77. DOI: 10.1007/1-84628-145-8_5.
41. Xeveugle R., Ubar R. Modeling VHDL clock-driven multi-processes by decision diagrams. *Journal of Electron Technology*, 1999, vol. 32, no. 3, pp. 282-287.
42. Ubar R., Morawiec A., Raik J. Cycle-based simulation algorithms for digital systems using high-level decision diagrams. *DATE*, 2000, pp. 743.
43. Raik J., Ubar R. Fast test pattern generation for sequential circuits using decision diagram representations. *Journal of Electronic Testing: Theory and Applications,* 2000, vol. 16, no. 3, pp. 213-226. DOI: 10.1023/A:1008335130158.
44. Ubar R., Devadze S., Jenihhin M., Raik J., Jervan G., Ellervee P. Hierarchical calculation of malicious faults for evaluating the fault-tolerance. *4th IEEE DELTA,* 2008, pp. 222-227. DOI: 10.1109/DELTA.2008.60.
45. Jenihhin M., Raik J., Fujiwara H., Ubar R., Viilukas T. An approach for verification assertions Reuse in RTL test pattern generation. *Proc. of the IEEE 11th Workshop on RTL and High Level Testing -WRTLT'10*, 2010, pp. l-6.
46. Raik J., Repinski U., Ubar R., Jenihhin M., Chepurov A. High-level design error diagnosis using backtrace on decision diagrams. *The 28th IEEE NORCHIP Conference*, Tampere. November 15-16, 2010. DOI: 10.1109/NORCHIP.2010.5669486.
47. Raik J., Repinski U., Ubar R., Jenihhin M., Chepurov A. High-level design error diagnosis using backtrace on decision diagrams. *The 28th IEEE NORCHIP Conference*, Tampere. November 15-16. 2010. DOI: 10.1109/NORCHIP.2010.5669486.
48. Guarnieri V., DiGuglielmo G., Bombieri N., Pravadelli G., Fummi F., Hantson H., Raik J., Jenihhin M., Ubar R. On the Reuse of TLM mutation analysis at RTL. *Journal of Electronic Testing, 2012.* DOI 10.1007/sl0836-012-5303-6.
49. Karputkin A., Ubar R., Tombak M., Raik J. Automated correction of design errors by edge redirection on high-level decision diagrams. *IEEE International Symposium on Quality Electronic Design - ISQED*, Santa Clara, CA USA, 2012. DOI: 10.1109/ISQED.2012.6187566.