Estonian Journal of Engineering, 2010, **16**, 1, 39–55 doi: 10.3176/eng.2010.1.06

Canonical representations of high-level decision diagrams

Anton Karputkin[a], Raimund Ubar[a], Jaan Raik[a] and Mati Tombak[b]

[a] Department of Computer Engineering, Tallinn University of Technology, Raja 15, 12618 Tallinn, Estonia; {kanton, raiub, jaan}@pld.ttu.ee

[b] Department of Informatics, Tallinn University of Technology, Raja 15, 12618 Tallinn, Estonia; mati@ut.ee

Received 28 October 2009, in revised form 26 January 2010

Abstract. In this paper we give a short overview of the decision diagrams, and define a special class of high-level decision diagrams (HLDD) for formal representation of digital systems. We show how the HLDDs can be used for high-level verification of digital systems. For this purpose, HLDDs are represented by characteristic polynomials as a canonical form of HLDDs. The polynomials can be used for proving the equivalence between two HLDDs, which have the same functionalities but may have different internal structures. Some possibilities are shown how to cope with the complexity of the verification problem.

Key words: hardware verification, high-level decision diagrams, characteristic polynomials.

1. INTRODUCTION

As the complexity of digital systems continues to increase, the traditional gate level modelling of systems, especially for verification and test generation purposes, has become obsolete. Economical and practical reasons have pushed the designers to apply automatic test pattern generation at higher abstraction levels to implement functional or hierarchical test strategies, or to approach design validation with the goal to early identify and remove design errors at higher functional levels, saving time and money. Thus many functional automated test pattern generators (ATPG) have been proposed in the literature to generate effective test sequences at the higher [1–3], behaviour [4,5] or functional register-transfer levels [6,7]. A special case of functional techniques are ATPGs, based on perturbation of statements in the digital circuit model using VHDL or RTL [8,9]. Hierarchical approach [10,11], compared to pure functional approach, lies in the possibility of constructing test plans on higher levels and modelling faults on more detailed lower levels, which results in better test quality.

The efficiency of high-level test generation for complex digital systems depends essentially on selecting the diagnostic model of the system and the way how to represent and handle the faults of the system. High-level functional ATPGs can be divided into two main categories: random-based and deterministic. The first set adopts simulation-based strategies, guided by genetic algorithms or other probabilistic techniques [2,3]. They rely on functional fault models and simulation of HDL descriptions (e.g. SystemC, VHDL, Verilog, etc.) of the design. These ATPGs are fast, but they cannot guarantee high fault coverage and they tend to generate very long test sequences. Deterministic ATPGs are based on mathematical strategies to allow a complete exploration of the system's state space [4,6,7,10,11], thus covering corner cases, but they require a larger amount of timing and memory resources.

In search for efficient high-level models, recently a number of papers has been published on implementing assignment decision diagram (ADD) models [12] combined with SAT methods to address register-transfer level (RTL) test pattern generation [6,7]. A promising approach is to use high-level decision diagrams (HLDD) [13] that, unlike ADDs, can be viewed as a generalization of binary decision diagrams (BDDs). HLDDs allow modelling of different abstraction levels from RTL to behavioural while ADDs are limited to RTL only. HLDDs have proven to be an efficient model for simulation [14], fault modelling [15] and test generation [10,11] as they provide for fast evaluation by graph traversal and for easy identification of cause–effect relationships. Further improvements in using HLDDs for high-level functional test generation were reached by combining HLDDs with extended finite state machines (EFSM) [16,17].

Within the last two decades binary decision diagrams (BDD) have become the state-of-the-art data structure in VLSI CAD for representation and manipulation of Boolean functions [18–25]. They were introduced first in 1959 by C. Y. Lee in the form of binary decision programs for representing digital circuits [18]. In 1976 the same model, called alternative graphs [19,20], was introduced for test generation purposes. Independently the same model was introduced into the field of test generation by Akers [21] under the name of BDDs. Today the theory of BDDs is developing quickly with the main purpose of efficient manipulation of logic expressions. The subset of BBDs, called structurally synthesized binary decision diagrams (SSBDDs), represent directly in the graph the structural features of circuits [19,20,26,27]. While traditionally BDDs are generated by Shannon's expansions, which allow to extract only the Boolean function of the logic circuit, the SSBDDs are generated by a superposition procedure that extracts both, function and data about structural paths of the circuit. This feature makes them preferable for diagnosis related tasks like fault modelling, simulation, test generation and fault location.

HLDDs represent a generalization of SSBDDs for modelling the functions and structure of digital systems on higher behaviour, functional or RTL abstraction levels. HLDDs are an excellent way to represent cause–effect and effect–cause relationships at higher levels of system abstraction as a basis for fault diagnosis in technical systems. They allow to skip dedicated low-level technical problems, related to semiconductor technology and to concentrate only on the high-level logic abstractions to carry out diagnostic reasoning. This is the only way how to handle today's complex systems. Moreover, the test related procedures, developed for SSBDDs, can be easily generalized for HLDDs to handle digital systems, represented at higher levels. In [28] two methods for synthesis of HLDDs were proposed. The first method is based on symbolic execution of procedural descriptions, which corresponds to functional representation of the system, e.g. on the behavioural level. The second one is based on iterative superposition of HLDDs, and the created model corresponds to the high-level structural representation of the system. The second method can be regarded as the generalization of the superposition of SSBDDs [29].

An example of a structurally synthesized HLDD, which represents a RTL data path of a digital system shown in Fig. 1 with functions of the components in Table 1, is depicted in Fig. 2. Variables R_1 and R_2 represent registers, *IN* represents the input bus, 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 $R_1 \cdot R_2$ represent adder and multiplier, respectively. Each node in the HLDD 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 behaviour of the input logic of the register R_2. The bold path in Fig. 2 shows the active mode of the system in the case of input control vector $(y_1, y_2, y_3, y_4) = (1, 0, 3, 2)$, which means that during this clock cycle the system calculates the multiplication $R_2 = R_1 \cdot R_2$. The structural relationships between the HLDD and the original system are highlighted by dotted lines in Fig. 2. The area in Fig. 2 denoted by R_2 corresponds to the DD for the subcircuit R_2, consisting of register R_2 with its input logic in Fig. 1; the area denoted by $R_2 + M_3$ corresponds to the composite DD for the subcircuit, consisting of components M_2, multiplier, M_3 and R_2, highlighted by dark colour in Fig. 1; the area, denoted by $c(M_1)$, corresponds to the DD for the subcircuit consisting of the components M_1 and adder; finally, the area denoted by $d(M_2)$ corresponds to the DDs for the subcircuit, consisting of the components M_2 and multiplier.

Table 1. Functions of the blocks of system in Fig. 1

y_1	M_1/a	y_2	M_2/b	y_3	M_3/e	y_4	R_2
0	R_1	0	R_1	0	$M_1 + R_2$	0	0
1	*IN*	1	*IN*	1	*IN*	1	R_2
				2	R_1	2	M_3
				3	$M_2 \cdot R_2$		

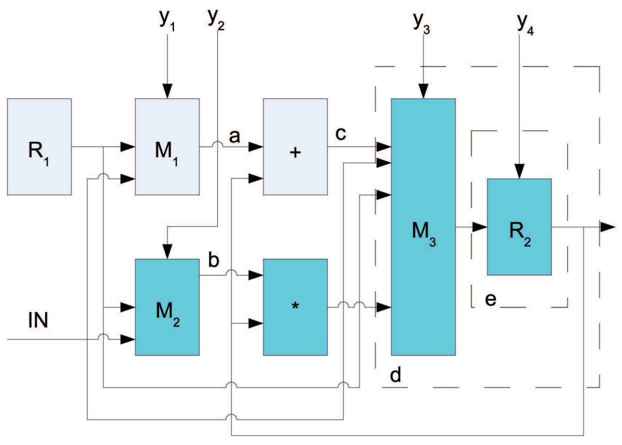

Fig. 1. RTL data path of a digital system.

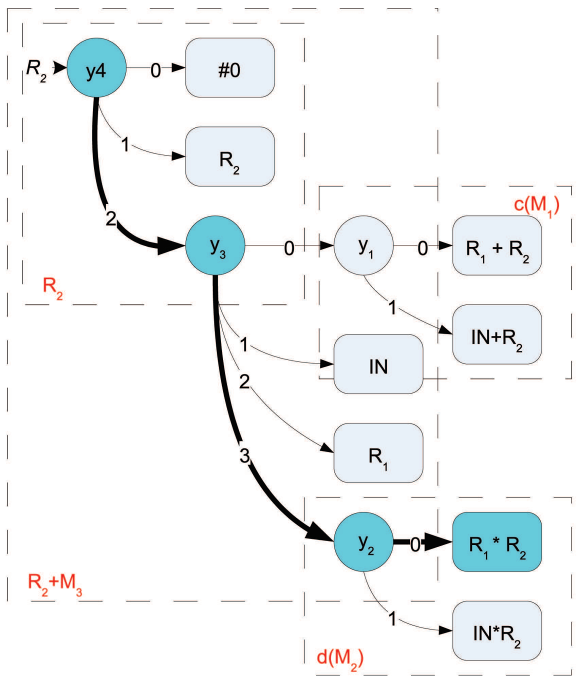

Fig. 2. HLDD for the system in Fig. 1.

In the HLDDs the internal nodes represent the control part of a system and the terminal nodes represent the data manipulation part of the latter. In the general case a system is described by a set of HLDDs, where each HLDD represents a controlled input logic of a register.

In this paper we will show how the HLDDs can be used for high-level verification purposes (more precisely, for probabilistic equivalence checking), for example in the cases when we have got two HLDD representations of a system: (1) high level specification, and (2) high level implementation. We will describe the diagrams by sets of characteristic polynomials and show how to use them in practice. The idea to represent digital systems as polynomials is not new, one can find similar ideas adapted for gate level models in papers [30,31]. An attempt to move up to the higher levels was made in [32]. However, at these levels we do not have such convenient and well-known formal representation of the system as boolean expressions are for the gate level. Before strarting to compute a polynomial we need an initial definition of the function it describes. The multivalued decision diagrams proposed for this purpose in [32] is not the best choice: computing them is the problem itself. This work combined with our previous paper [28] gives a way to get the canonical form of the digital system at higher levels directly from its description.

The paper is organized as follows. In Section 2 we give the formal definition of HLDDs, in Section 3 we show the possibility of representing the HLDDs by characteristic polynomials, which can be used as a canonical form of HLDDs. In Section 4 we show how the characteristic polynomials can be used for proving the equivalence between two HLDDs, which may have different internal structures. Section 5 concludes the paper.

2. DEFINITION OF HLDD

In this section we define the objects being studied in the current article: HLDDs and functions represented by them, further called HLDD functions.

Consider a digital system (Z, F) as a network of subsystems or components where Z is the set of variables (Boolean, Boolean vectors or integers), which represent connections between components, inputs and outputs of the network. Let $Z = X \cup Y$, where X is the set of function arguments and Y is the set of function values where $Q = X \cap Y$ is the set of state variables. $D(z)$ denotes the finite set of all possible values for $z \in Z$ and $D(U)$ is the set of all possible vectors in $U \subseteq Z$. Obviously, if $U = \{u_1, \ldots, u_n\}$ then $D(U) = D(u_1) \times \ldots \times D(u_n)$. Let F be a set of digital functions: $z_k = f_k(Z_k)$ where $z_k \in Z, f_k \in F$, and $Z_k \subset Z$ (k iterates over all elements in F).

Definition 1. *The high-level decision diagram representing the function $f_k : D(Z_k) \to D(z_k)$ is a directed acyclic graph $G = (V, E)$ with one root node and a set of terminal nodes, where:*

– *Each non-terminal node is labelled by some input or control variable $x \in X \cup Q$. We shall denote the variable of node v by x_v.*
– *Each terminal node w is labelled by some function $g_w : D(Z_w) \to D(z_k)$ (possibly a constant or a single variable), where $Z_w \subseteq Z_k$.*

– *Each edge $e = (v, u)$ is labelled by some constant $c_e \in D(x_v)$.*
– *Each two edges $e_1 = (v, u_1)$ and $e_2 = (v, u_2)$ going from the same source node are labelled by different constants $c_{e_1} \neq c_{e_2}$.*
– *If the node v is labelled by x_v then the number of edges going from this node is $|D(x_v)|$.*

Remark 1. Each BDD is HLDD as well, with two terminal vertices labelled by constant functions 0 and 1, and $D(x) = \{0, 1\}$ for every variable x.

We shall denote the set of terminal nodes by V^T, the set of non-terminal nodes by V^N and the set of all successors of the node v by $\Gamma(v)$. For non-terminal nodes $v \in V^N$ an onto function exists between the values $c \in D(x_v)$ of labels x_v and the successors $v^c \in \Gamma(v)$ of v. By v^c we denote the successor of v for the value $x_v = c$. The edge (v, v^c), which connects nodes v and v^c, is called *activated* iff there exists an assignment $x_v = c$. Activated edges, which connect v_i and v_j, make up an *activated path* $l(v_i, v_j) \subseteq V$. An activated path $l(v_0, v^T)$ from the root node v_0 to a terminal node v^T is called *full activated path* and v^T itself is *activated terminal node*. Without loss of generality we assume further that each variable has at least two values, i.e. $\forall z \in Z,\ D(z) > 1$.

Let $S = X \cup Q$ and let D_i designate a subset of $D(S)$, such that assignments from it will activate the terminal node v_i^T. Thus $D(S)$ is being split to non-intersecting sets $D_1, ..., D_t$, where $t = |V^T|$. More formally,

$$\bigcup_{i=1}^{t} D_i = D(S) \bigwedge \forall i, j (i \neq j \Rightarrow D_i \cap D_j = \emptyset).$$

Now it is easy to get an algebraic expression for the HLDD function f_k. Let $\alpha \in D(S)$ be some assignment to input and control variables and $\chi_i : D(S) \rightarrow \{0, 1\}$ be the characteristic function of the set D_i, i.e. $\chi_i(\alpha_1, \ldots, \alpha_n) = 1 \Leftrightarrow (\alpha_1, \ldots, \alpha_n) \in D_i$. We will use a formula

$$f_k(Z_k) = \textbf{case}_{i=1}^{t}\ \chi_i(S) \rightarrow g_i(Z_i) \tag{1}$$

as a shorthand for the algorithm:

```
begin
    **if** $\alpha \in D_1$ **then** $f_k(Z_k) = g_1(Z_1)$ **endif**
    ...
    **if** $\alpha \in D_t$ **then** $f_k(Z_k) = g_t(Z_t)$ **endif**
end
```

Let us continue with some examples:

Example 1. An HLDD in Fig. 3 evaluates the next state of the variable C. In this example $Z = \{C', A', B', q, x_A, x_B, x_C\}$. The apostrophe means the previous value of the variable. Depending on the values of q, x_A, x_B and x_C, the next value of C is its previous value C', negation $\overline{C'}$ or $A' + B'$.

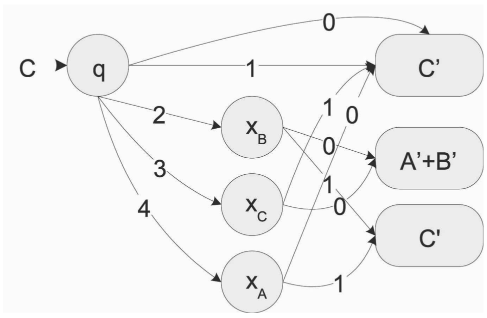

Fig. 3. HLDD that evaluates the next state of the variable C.

3. CHARACTERISTIC POLYNOMIALS

There are two ways to generate an HLDD for some digital system: one based on procedural description and another based on iterative superposition. These methods are described in [28]. Further research on HLDD properties requires us to make the following important assumption: **each path in the diagram does not contain any of the control variables twice**. Diagrams generated by the first algorithm really have such property. The second method can theoretically produce such paths. If we encounter the same variable two times in a path, we can duplicate all variables between the occurrencies of this variable and make an equivalent diagram without such redundancies. Let a path contain nodes, labelled by variables $x_0, x_1, ..., x_k, x_0$. We should produce a new diagram, where this chain would be replaced by two new chains, $x_0, x_1, ..., x_k$ and $x_1, ..., x_k, x_0$. The following theorem will help us.

Theorem 1. *Suppose an HLDD, containing a chain of non-terminal nodes labelled by variables $x_0, x_1, ..., x_k, x_0$, was transformed in the following way:*

- *Remove nodes labelled $x_0, x_1, ..., x_k, x_0$.*
- *Add $2(k+1)$ nodes labelled $x_1, ..., x_k, x_0$ (the first chain) and $x_0, x_1, ..., x_k$ (the second chain).*
- *All connections from and to the nodes of the first chain will remain the same.*
- *All connections from and to x_0 occurence in the second chain will remain the same.*
- *Connections from other nodes to $x_1, ..., x_k$ in the second chain will be removed.*

Then the result is equivalent to the original diagram.

Proof. We shall prove that the second diagram contains all paths from the first one that could be activated and vice versa. $\Rightarrow$:

- Obviously, all paths not containing mentioned nodes remain unchanged.
- All paths containing the first occurrence of the x_0 can be activated in the second chain.

– All paths containing the second occurrence of the x_0 or only intermediate nodes $x_1, ..., x_k$ can be found in the first chain.

$\Leftarrow$: The similar check for all possible paths in our two chains shows that they can be activated in the first diagram. $\square$

Example 2. Figure 4 illustrates the transformation described above. Some vertex and all edge labels are not shown because they are not important. We have a path $x \rightarrow y \rightarrow z \rightarrow x$ that is being split in two chains $x \rightarrow y \rightarrow z$ and $y \rightarrow z \rightarrow x$.

Suppose now that we have two HLDDs, representing the same functionality. They can look very different as the next examples illustrate.

Example 3. Figure 5 shows two HLDDs of the same function. The only difference is how we evaluate this function. The first diagram represents the situation when we first check the value of the variable X, then, if X equals 2, we check the Y value. The second diagram displays the process of evaluation starting with checking Y.

Example 4. Figure 6, left side, shows the diagram with 4 non-terminal nodes labelled by variables X, Y, Z. However, it is easy to see that the variable Y is redundant in this case and the diagram from the right side without this variable is the equivalent one.

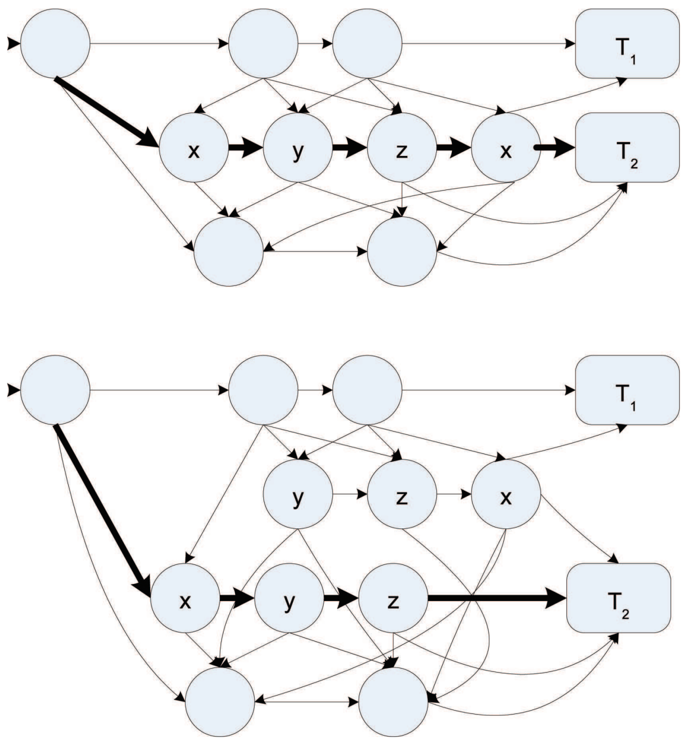

Fig. 4. Removing duplicate variables.

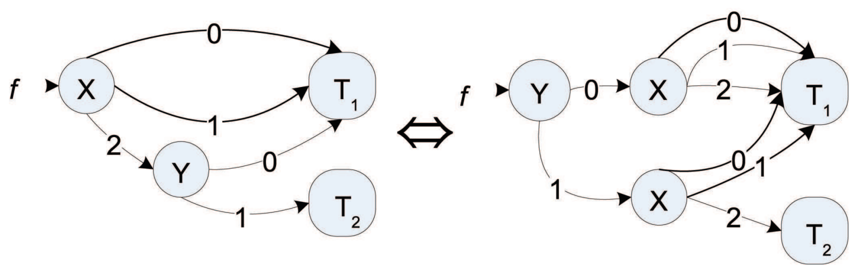

Fig. 5. Variables in different order.

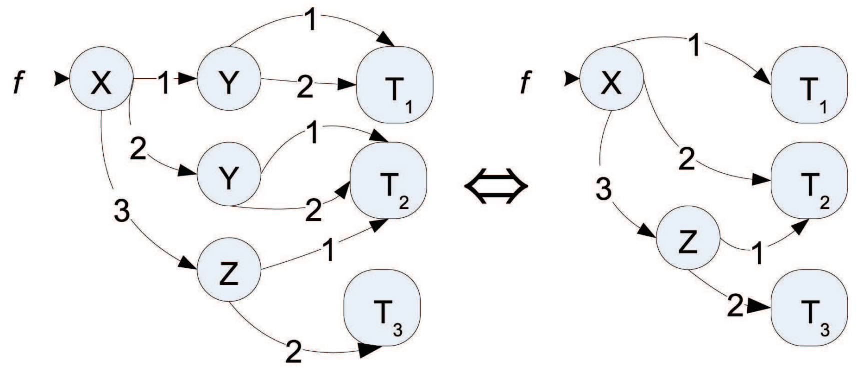

Fig. 6. Redundant node.

Let us suppose we have two diagrams G_1 and G_2, with the same sets of control variables and terminal nodes and wish to prove that they are equivalent. Actually it means that the function (1) for both diagrams is the same. As the functions g_i are the same, we should compare the corresponding characteristic functions. Without loss of generality we may assume that $D(x_i) = \{1, \dots, |D(x_i)|\}$. If not we may define a bijective mapping $h : D(z_i) \to \{1, \dots, |D(x_i)|\}$ and use the values of this function instead of their originals. Then the following theorem takes place.

Theorem 2. *The characteristic function of the set D_i, $\chi_i(S)$, can be represented by a unique polynomial $P_i : \mathbb{Q}^n \to \mathbb{Q}$ of degree at most $\sum_{i=1}^{n} (|D(x_i)| - 1)$ where we have for each vector $\alpha = (\alpha_1, ..., \alpha_n) \in D(S)$*

$$\chi_i(\alpha_1, ..., \alpha_n) = P_i(\alpha_1, ..., \alpha_n).$$

Proof. We have $D(S)$ different vectors in $\mathbb{Q}^n$. The sought-for polynomial should be equal to 1 for vectors from D_i and to 0 for vectors from $D(S)\backslash D_i$. In numerical analysis the Lagrange interpolation polynomial is well-known [33]: consider we have measured the values for a function $f : [a, b] \to \mathbb{R}$ at points $x_0, ..., x_n$, and obtained results are $y_0, ..., y_n$, then we can interpolate them to the whole segment $[a, b]$ by a polynomial of degree at most n:

$$f(x) \approx P(x) = \sum_{i=0}^{n} y_i \frac{(x - x_0)...(x - x_{i-1})(x - x_{i+1})...(x - x_n)}{(x_i - x_0)...(x_i - x_{i-1})(x_i - x_{i+1})...(x_i - x_n)}.$$

If f is a polynomial of such degree then we get the exact result, otherwise there will be some error. Note that for $x = x_0, ..., x_n$ we will always get the exact result: $P(x_i) = y_i = f(x_i)$ and it is a polynomial of lowest degree that gives such result. This is the property we are interested in the current paper. Although in numerical analysis textbooks only the case of one-variable function is usually studied, these results can be easily transferred to the multiple-variable case. So, our sought-for polynomial P_i is the Lagrange polynomial that evaluates to 1 for each vector from D_i and to 0 for each vector from $D(S) \backslash D_i$:

$$P_i(x_1, ..., x_n) = \sum_{(\alpha_1, ..., \alpha_n) \in D_i} \prod_{j=1}^{n} \prod_{\substack{k=1 \\ k \neq \alpha_j}}^{|D(x_j)|} \frac{x_j - k}{\alpha_j - k}. \tag{2}$$

The degree of this polynomial is at most $\sum_{i=1}^{n} (|D(x_i)| - 1)$. Let us prove that this is the only polynomial of such degree.

– *The basis*. Let $n = 1$. Assume we have 2 polynomials, $P(x_1)$ and $Q(x_1)$, $\deg P = \deg Q = |D(x_1)| - 1$ and $\forall (i \in 1..|D(x_1)|)\ P(i) = Q(i)$. Then the polynomial $P - Q$ has $|D(x_1)|$ roots, from 1 to $|D(x_1)|$, which could be only in case $P \equiv Q$.
– *The induction step*. The proof is similar to the basis case one: assume we have 2 polynomials, $P(x_1, ..., x_n)$ and $Q(x_1, ..., x_n)$, $\deg P = \deg Q = \sum_{i=1}^{n} (|D(x_i)| - 1)$. After assigning values to x_1 we get $|D(x_1)|$ pairs of $(n - 1)$-variable polynomials. They are pairwise equal by induction hypothesis. Thus, the polynomial function $P - Q : \mathbb{Q} \to \mathbb{Q}[x_2, ..., x_n]$ of degree at most $|D(x_1)| - 1$ has $|D(x_1)|$ roots in $\mathbb{Q}[x_2, ..., x_n]$. Thus, $P \equiv Q$.

□

Corollary 1. *Two diagrams G_1 and G_2 are equivalent iff they have equal sets of control variables, terminal nodes and the polynomial representations* (2) *of characteristic functions for any two corresponding terminal nodes are the same.*

We shall call the right side of formula 2 the *characteristic polynomial* of the node v_i^T. The Algorithm 1 shows how to get such polynomials for a certain diagram. Here we shall prove that it is correct.

Theorem 3. *The Algorithm* 1 *produces the set of characteristic polynomials for the diagram G.*

Algorithm 1. Evaluation of characteristic polynomials.

```
Input: HLDD $G$
Output: The set of characteristic polynomials for $G$
we shall evaluate polynomials node by node.  A polynomial
for node $v$ will be denoted by $P_v$;
order all nodes in $G$ topologically. Let $T$ be an array of ordered nodes;
$P_{T[0]} = 1$;
for $v \in T$ do
    $P_v = 0$;
end
for $v \in T$ do
    for parentnode $w$ of $v$ do
        $i = c_{(w,v)}$;
        $P_v$ += $P_w \prod_{\substack{j=1 \\ j \neq i}}^{|D(w)|} \frac{x_w - j}{i - j}$;
    end
end
return $\{P_v | v \in V^T\}$
```

Proof. Let W be a set of all paths from the root node to some terminal node v^T. Each path $w \in W$ activated by the assignment $(x_{v_1} = \alpha_1, ..., x_{v_l} = \alpha_l)$ will be represented in the resulting polynomial by the following summand:

$$\prod_{j=1}^{l} \prod_{\substack{k=1 \\ k \neq i_j}}^{|D(x_{v_j})|} \frac{x_{v_j} - k}{\alpha_j - k}. \tag{3}$$

(This can be easily proved by induction). The resulting polynomial will be the sum of these summands over all paths from W. As we have assumed in the beginning of the chapter, all variables in w are different. So, $l \leq n$. The only difference between summands in (2) and (3) is the bound of the first product sign: generally, the path w should not contain all variables; some of them may be missing. This means that one path actually represents $|D(r_1) \times ... \times D(r_{n-l})|$ assignments $\{(x_{v_1} = \alpha_1, ..., x_{v_l} = \alpha_l, r_1 = \alpha_{l+1}, ..., \alpha_{n-l} = i_n) | \alpha_m \in D(r_{m-l}), l < m \leq n\}$ where $\{r_1, ..., r_{n-l}\} = S \backslash \{x_{v_1}, ..., x_{v_l}\}$. But we have

$$\sum_{(\alpha_{l+1},...,\alpha_n) \in D(r_1) \times ... \times D(r_{n-l})} \left(\prod_{j=1}^{n-l} \left(\prod_{\substack{k=1 \\ k \neq \alpha_j}}^{|D(x_j)|} \frac{x_j - k}{\alpha_j - k} \right) \right) = 1. \tag{4}$$

This is the Lagrange polynomial that evaluates to 1 for all possible values from $D(r_1) \times ... \times D(r_{n-l})$. The simplest polynomial with such property is the constant 1, so they should coincide. Multiplying this polynomial with our summand gives the sum of $D(r_1) \times ... \times D(r_{n-l})$ summands, each representing certain assignment for the whole set of variables. Finally, adding them together results in the formula (2) (none of the summands will appear twice, because all paths have the same source node and thus assignments of corresponding paths will differ for at least one variable; otherwise they would never branch off). □

Example 5. Let us now find the characteristic polynomials for HLDD from Example 1. First of all we change the labels of edges labelled by 0 to 5 for the one going from node q and to 2 for others. We have 4 paths to the first terminal node: $(q = 5)$, $(q = 1)$, $(q = 3, x_C = 1)$, $(q = 4, x_A = 2)$. Thus,

$$\begin{aligned}
P_1(q, x_A, x_B, x_C) &= \frac{(q-1)(q-2)(q-3)(q-4)}{(5-1)(5-2)(5-3)(5-4)} \\
&\quad + \frac{(q-5)(q-2)(q-3)(q-4)}{(1-5)(1-2)(1-3)(1-4)} \\
&\quad + \frac{(q-5)(q-1)(q-2)(q-4)(x_C-2)}{(3-5)(3-1)(3-2)(3-4)(1-2)} \\
&\quad + \frac{(q-5)(q-1)(q-2)(q-3)(x_A-1)}{(4-5)(4-1)(4-2)(4-3)(2-1)} \\
&= -\frac{1}{6}q^4x_A - \frac{1}{4}q^4x_C + \frac{3}{4}q^4 + \frac{11}{6}q^3x_A + 3q^3x_C - \frac{53}{6}q^3 \\
&\quad - \frac{41}{6}q^2x_A - \frac{49}{4}q^2x_C + \frac{143}{4}q^2 + \frac{61}{6}qx_A \\
&\quad + \frac{39}{2}qx_C - \frac{173}{3}q - 5x_A - 10x_C + 31.
\end{aligned}$$

For the second node we have two paths $(q = 2, x_B = 2)$ and $(q = 3, x_C = 2)$, so the second polynomial will be

$$\begin{aligned}
P_2(q, x_A, x_B, x_C) &= \frac{(q-5)(q-1)(q-3)(q-4)(x_B-1)}{(2-5)(2-1)(2-3)(2-4)(2-1)} \\
&\quad + \frac{(q-5)(q-1)(q-2)(q-4)(x_C-1)}{(3-5)(3-1)(3-2)(3-4)(2-1)} \\
&= -\frac{1}{6}q^4x_B + \frac{1}{4}q^4x_C - \frac{1}{12}q^4 + \frac{13}{6}q^3x_B - 3q^3x_C + \frac{5}{6}q^3 \\
&\quad - \frac{59}{6}q^2x_B + \frac{49}{4}q^2x_C - \frac{29}{12}q^2 + \frac{107}{6}qx_B - \frac{39}{2}qx_C \\
&\quad + \frac{5}{3}q - 10x_B + 10x_C.
\end{aligned}$$

Finally, paths $(q = 2, x_B = 2)$ and $(q = 4, x_A = 1)$ give us the third polynomial:

$$\begin{aligned}
P_3(q, x_A, x_B, x_C) &= \frac{(q-5)(q-1)(q-3)(q-4)(x_B-2)}{(2-5)(2-1)(2-3)(2-4)(1-2)} \\
&\quad + \frac{(q-5)(q-1)(q-2)(q-3)(x_A-2)}{(4-5)(4-1)(4-2)(4-3)(1-2)} \\
&= \frac{1}{6}q^4x_A + \frac{1}{6}q^4x_B - \frac{2}{3}q^4 - \frac{11}{6}q^3x_A - \frac{13}{6}q^3x_B + 8q^3 \\
&\quad + \frac{41}{6}q^2x_A + \frac{59}{6}q^2x_B - \frac{100}{3}q^2 - \frac{61}{6}qx_A - \frac{107}{6}qx_B \\
&\quad + 56q + 5x_A + 10x_B - 30.
\end{aligned}$$

4. PRACTICAL USAGE OF CHARACTERISTIC POLYNOMIALS

A skeptically minded reader, after looking at Eq. (2), may notice that the evaluation of such polynomial for a large modern digital system would take huge amount of time. This is true. However, we can get a lot of useful information about the function without evaluating it directly in analytical form. Here we are giving an algorithm that can be used in HLDD verification and is residing in complexity class P. It evaluates the coefficients of lower degrees. Before we provide it we should agree on mapping from $D(z)$ to $\mathbb{Z}$. Generally, each mapping is good except the ones containing zeroes. Multiplication by zero will lose information about some paths. Thus, for example in case of $D(z) = 0, 1, ..., |D(z)| - 1$ it is better to use mapping $h(z) = z + 1$. Indeed, assume $0 \in D(z)$ and we need to evaluate the constant term. In this case all assingnments with $z \neq 0$ will produce summands

Algorithm 2. Evaluating polynomial coefficients of degrees $\leq k$.

```
Input: HLDD $G$, maximal degree $k$
Output: Set of polynomials, one per each terminal node, that contain members of the
        characteristic polynomials with degree $\leq k$
order all nodes in $G$ topologically. Let $T$ be an array of ordered nodes;
$P_{T[0]} = 1$;
for $v \in T$ do
    $P_v = 0$;
end
for $v \in T$ do
    for parentnode $w$ of $v$ do
        $i = c_{(w,v)}$;
        $P_v$ += $P_w \prod_{\substack{j=1 \\ j \neq i}}^{|D(w)|} \frac{x_w - j}{i - j}$;
        if $P_v$ contains coefficiens of degree $> k$ delete them;
    end
end
return $\{P_v | v \in V^T\}$
```

containing factor z in Eq. (2). The constant terms of these summands are equal to 0, so they are not taken into account when we evaluate the last coefficient of the whole polynomial.

Once we have chosen the proper mapping we can use Algorithm 2 to calculate all coefficients of degrees $\leq n$. This method cannot allow us to be 100% confident that our two diagrams are equivalent but if they are not then it can be found very quickly. We continue with an example.

Example 6. Let us introduce an error to the diagram in Fig. 3. For instance, let the edge $(x_B, A' + B')$ now point to the first terminal node, C', as it is shown in Fig. 7. The characteristic polynomial of the third terminal node remains the same, while 2 others will change:

$$\begin{aligned}
P_1^{err}(q, x_A, x_B, x_C) &= \frac{(q-1)(q-2)(q-3)(q-4)}{(5-1)(5-2)(5-3)(5-4)} \\
&\quad + \frac{(q-5)(q-2)(q-3)(q-4)}{(1-5)(1-2)(1-3)(1-4)} \\
&\quad + \frac{(q-5)(q-1)(q-2)(q-4)(x_C-2)}{(3-5)(3-1)(3-2)(3-4)(1-2)} \\
&\quad + \frac{(q-5)(q-1)(q-2)(q-3)(x_A-1)}{(4-5)(4-1)(4-2)(4-3)(2-1)} \\
&\quad + \frac{(q-5)(q-1)(q-3)(q-4)(x_B-1)}{(2-5)(2-1)(2-3)(2-4)(2-1)} \\
&= -\frac{1}{6}q^4 x_A - \frac{1}{6}q^4 x_B - \frac{1}{4}q^4 x_C + \frac{11}{12}q^4 + \frac{11}{6}q^3 x_A \\
&\quad + \frac{13}{6}q^3 x_B + 3q^3 x_C - 11q^3 - \frac{41}{6}q^2 x_A - \frac{59}{6}q^2 x_B \\
&\quad - \frac{49}{4}q^2 x_C + \frac{547}{12}q^2 + \frac{61}{6}q x_A + \frac{107}{6}q x_B \\
&\quad + \frac{39}{2}q x_C - \frac{151}{2}q - 5x_A - 10x_B - 10x_C + 41,
\end{aligned}$$

$$\begin{aligned}
P_2^{err}(q, x_A, x_B, x_C) &= \frac{(q-5)(q-1)(q-2)(q-4)(x_C-1)}{(3-5)(3-1)(3-2)(3-4)(2-1)} \\
&= \frac{1}{4}q^4 x_C - \frac{1}{4}q^4 - 3q^3 x_C + 3q^3 + \frac{49}{4}q^2 x_C \\
&\quad - \frac{49}{4}q^2 - \frac{39}{2}q x_C + \frac{39}{2}q + 10x_C - 10.
\end{aligned}$$

As we see, even only the check for constant terms would detect the error.

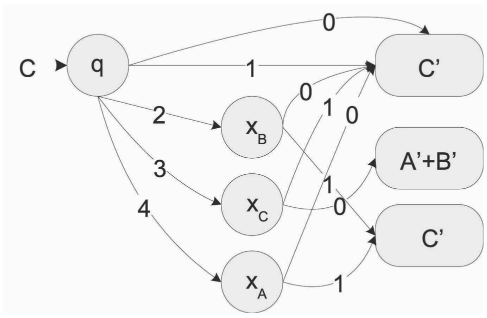

Fig. 7. HLDD with design error.

5. CONCLUSIONS

A novel method for probabilistic equivalence checking of digital systems was proposed. It is based on representing the high-level dedcision diagrams as the model of digital systems by the sets of characteristic polynomials. It was shown that this representation is canonical, i.e. the sets of polynomials for equivalent diagrams are the same up to the names of the variables. Computing the full set of polynomials is unfeasible for large diagrams as it demands checking all assignments to the control variables. To cope with this problem we have developed a polynomial algorithm for probabilistic checking.

The algorithm calculates coeficients of low-degree summands up to the given fixed degree k. If the coeficients do not coincide, then the HLDDs are definitely different; if the coefficients coincide, then the HLDDs are with high probability equivalent whereas the probability depends on the chosen degree of k. To prove that the HLDDs are not equivalent is possible also by only comparing constant terms of the polynomial. For instance, in Example 6, we can see, that even a small erroneous change in the diagram is detectable by comparing constant terms.

The technique itself does not have limitations. However, for some classes of digital systems, optimization techniques may be needed to create efficient HLDD models, but this topic does not belong to the scope of the paper. Also, the equivalence checking of the terminal node functions was left uncovered. The general idea is that those functions are usually simple ones and can be verified using gate level methods. Nevertheless we plan to look at those functions more intently in our future research.

ACKNOWLEDGEMENTS

The work has been supported by the European Commission Framework Program FP7 CREDES, by the European Union through the European Regional Development Fund, by the Estonian Science Foundation (grants Nos. 7068 and 7483), and Enterprise Estonia funded ELIKO Development Centre.

REFERENCES

1. Kapuer, R. High Level ATPG is important and is on its way! In *Proc. IEEE International Test Conference*. Atlantic City, NJ, 1999, 1115–1116.
2. Corno, F., Cumani, G., Sonza Reorda, M. and Squillero, G. Effective techniques for high-level ATPG. In *Proc. IEEE Asian Test Symposium*. Kyoto, Japan, 2001, 225–230.
3. Fin, A. and Fummi, F. Genetic algorithms: the philosopher's stone or an effective solution for high-level TPG? In *Proc. IEEE High-Level Design Validation and Test Workshop*. San Francisco, CA, 2003, 163–168.
4. Ferrandi, F., Fummi, F. and Sciuto, D. Implicit test generation for behavioral VHDL models. In *Proc. IEEE International Test Conference*. Washington, D.C., 1998, 436–441.
5. Xin, F., Ciesielski, M. and Harris, I. Design validation of behavioral VHDL descriptions for arbitrary fault models. In *Proc. IEEE European Test Symposium*. Tallinn, Estonia, 2005, 156–161.
6. Ghosh, I. and Fujita, M. Automatic test pattern generation for functional register-transfer level circuits using assignment decision diagrams. *IEEE Trans. Comput.-Aided Des. Integrated Circuits Systems*, 2001, **20**, 402–415.
7. Zhang, L., Ghosh, I. and Hsiao, M. Efficient sequential ATPG for functional RTL circuits. In *Proc. International Test Conference*. Charlotte, NC, 2003, 290–298.
8. Armstrong, J. A., Lam, F. S. and Ward, P. C. Test generation and fault simulation for behavioural models. In *Performance and Fault Modelling with VHDL* (Shoen, J. M., ed.). Prentice Hall, Englewood Cliffs, NY, 1992, 240–303.
9. Cho, H. Ch. and Armstrong, J. A. B-algorithm – a behavioural test generation algorithm. In *Proc. International Test Conference*. Washington, D.C., 1994, 968–979.
10. Raik, J. and Ubar, R. Fast test pattern generation for sequential circuits using decision diagram representations. *J. Electronic Testing: Theory and Applications*, 2000, **16**, 213–226.
11. Jervan, G., Ubar, R., Peng, Z. and Eles, P. Test generation: a hierarchical approach. In *System-level Test and Validation of Hardware/Software Systems* (Sonza Reorda, M., Peng, Z. and Violante, M., eds.), Springer, London, 2005, 63–77.
12. Chayakul, V., Gajski, D. D. and Ramachandran, L. High-level transformations for minimizing syntactic variances. In *Proc. ACM/IEEE Design Automation Conference*. Dallas, TX, 1993, 413–418.
13. Ubar, R. Test synthesis with alternative graphs. *IEEE Des. Test Comput.*, 1996, **13**, 48–57.
14. Ubar, R., Morawiec, A. and Raik, J. Back-tracing and event-driven techniques in high-level simulation with decision diagrams. In *Proc. IEEE International Symposium on Circuits and Systems '00*. Geneva, 2000, 208–211.
15. Ubar, R., Raik, J., Jutman, A., Instenberg, M. and Wuttke, H.-D. Modeling microprocessor faults on high-level decision diagrams. In *Proc. International Conference on Dependable Systems & Networks*. Anchorage, AL, 2008, C-17–C-22.
16. Di Guglielmo, G., Fummi, F., Marconcini, C. and Pravadelli, G. Improving gate-level ATPG by traversing concurrent EFSMs. In *Proc. IEEE VLSI Test Symposium*. Berkeley, CA, 2006, 172–179.
17. Guglielmo, G., Fummi, F., Jenihhin, M., Pravadelli, G., Raik, J. and Ubar, R. On the combined use of HLDDs and EFSMs for functional ATPG. In *Proc. 5th IEEE East-West Design and Test International Symposium*. Jerevan, Armenia, 2007, 503–508.
18. Lee, C. Y. Representation of switching circuits by binary decision programs. *Bell Syst. Techn. J.*, 1959, **38**, 985–999.
19. Ubar, R. Test generation for digital circuits with alternative graphs. *Proc. Tallinn Technical University*, 1976, No. 409, 75-81 (in Russian).

20. Plakk, M. and Ubar, R. Digital circuit test design using the alternative graph model. In *Automation and Remote Control*. Plenum Publishing Corporation, New York, 1980, **41**, 714–722.
21. Akers, S. B. Functional testing with binary decision diagrams. *J. Design Automat. Fault-Tolerant Comput.*, 1978, **2**, 311–331.
22. Bryant, R. E. Graph-based algorithms for Boolean function manipulation. *IEEE Trans. Computers*, 1986, **C-35,** 677–691.
23. Minato, S. *Binary Decision Diagrams and Applications for VLSI CAD*. Kluwer, Norwell, MA, 1996.
24. Sasao, T. and Fujita, M. (eds.). *Representations of Discrete Functions*. Kluwer, Norwell, MA, 1996.
25. Drechsler, R. and Becker, B. *Binary Decision Diagrams*. Kluwer, Boston, Dordrecht, London, 1998.
26. Ubar, R. Multi-valued simulation of digital circuits with structurally synthesized binary decision diagrams. In *Multiple Valued Logic*. Gordon and Breach Publishers, 1998, vol. 4, 141–157.
27. Jutman, A., Raik, J. and Ubar, R. SSBDDs: Advantageous model and efficient algorithms for digital circuit modeling, simulation & test. In *Proc. 5th International Workshop on Boolean Problems*. Freiberg, Germany, 2002, 157–166.
28. Ubar, R., Raik, J., Karputkin, A. and Tombak, M. Synthesis of high-level decision diagrams for functional test pattern generation. In *Proc. 16th International Conference on Mixed Design of Integrated Circuits and Systems*. Lodz, Poland, 2009, 519–524.
29. Ubar, R., Vassiljeva, T., Raik, J., Jutman, A., Tombak, M. and Peder, A. Optimization of structurally synthesized BDDs. In *Proc. 4th IASTED International Conference on Modelling, Simulation and Optimization*. Kauai, HI, 2004, 234–240.
30. Agrawal, V. D. and Lee, D. Characteristic polynomial method for verification and test of combinational circuits. In *Proc. 9th International Conference on VLSI Design: VLSI in Mobile Communication*. Bangalore, India, 1996, 341–342.
31. Jain, J., Bitner, J., Fussell, D. S. and Abraham, J. A. Probabilistic design verification. ICCAD-91. In *Digest of Technical Papers., 1991 IEEE International Conference on Computer-Aided Design*. Santa Clara, CA, 1991, 468–471.
32. Dubrova, E. and Sack, H. Probabilistic verification of multiple-valued functions. In *Proc. 30th IEEE International Symposium on Multiple-Valued Logic*. Portland, OR, 2000, 460–466.
33. Atkinson, K. A. *An Introduction to Numerical Analysis*, 2nd ed. J. Wiley, New York, 1989.

Kõrgtaseme otsustusdiagrammide kanooniline esitus

Anton Karputkin, Raimund Ubar, Jaan Raik ja Mati Tombak

On antud lühike ülevaade otsustusdiagrammidest ja defineeritud spetsiaalne kõrgtaseme otsustusdiagrammide (KTOD) mudel digitaalsüsteemide formaalseks esituseks. On näidatud, kuidas on võimalik rakendada KTOD-mudelit digitaalsüsteemide verifitseerimiseks kõrgtasandil. Selleks defineeritakse kõigepealt KTOD-mudeli kanooniline esitus karakteristlike polünoomide süsteemina. Edasi on näidatud, kuidas saab polünoomide abil tõestada kahe KTOD-mudeli ekvivalentsust, kus mudeli funktsionaalsused on samad, kuid struktuurid võivad olla erinevad. On näidatud võimalusi, kuidas verifitseerimise probleemi keerukusega toime tulla.