Computing

SAT-BASED METHOD OF VERIFICATION USING LOGARITHMIC ENCODING

Liudmila Cheremisinova, Dmitry Novikov

Abstract: The problem under discussion is to check whether a given combinational network realizes a system of incompletely specified Boolean functions. SAT-based procedure is discussed that formulates the overall problem as conventional conjunctive normal form (CNF) on the basis of encoding of multiple-output cubes the Boolean functions are specified on and checking whether the combinational network realizes them using a SAT solver. The novel method is proposed that speeds up the SAT-based procedure due to suggested efficient procedure of logarithmic encoding multiple-output cubes that allows reducing the number of variables to be additionally introduced into the CNF under construction.

Keywords: design automation, verification, simulation, CNF satisfiability.

ACM Classification Keywords: B.6.2 [Logic Design]: Reliability and Testing; G.4 [Mathematical Software]: Verification; B.6.2 [Reliability and Testing]: Error-checking.

Conference: The paper is selected from XVth International Conference "Knowledge-Dialogue-Solution" KDS-2 2009, Kyiv, Ukraine, October, 2009.

Introduction

The role of combinational verification becomes more and more important with the rapid increase of the complexity of designs synthesized by modern CAD (computer-aided design) tools. Today, verification is a bottleneck in the overall VLSI design cycle as it consumes up to 70% of design effort [Drechsler, 2000], [Kuehlmann, 2002]. The objective of verification is to ensure that implemented and specified behaviors are the same. In a typical scenario, there are two structurally similar circuit implementations of the same design, and the problem is to prove their functional equivalence. In contrast to that, in the paper the verification task is examined for the case, when desired functionality of the system under design is incompletely specified. Such a case usually occurs on early stages of designing when assignments to primary inputs of designed device exist which will never arise during a normal mode of the device usage. Thus when hardware implementing this device, its outputs in response of these inputs may be arbitrary specified. In this case verification methodologies must consider only possible input scenarios to the design under verification and verify that every possible output signal of the implemented behavior has its intended (described in initial specification) value.

We consider the verification problem for the case, when 1) desired incompletely specified functionality is given in the form of a system of incompletely specified Boolean functions (ISF system); 2) functions of the system are specified on intervals (cubes) of values of Boolean input variables and these intervals are large enough; 3) the ISF system is implemented in the form of a combinational circuit. Efficient methods for equivalence checking were proposed [Drechsler, 2000], [Kuehlmann, 2002], [Kropf, 1999] most of them can be organized into two major categories: simulation and SAT based equivalence checking. Both these techniques have in our case some peculiarities following from incompletely specified functionality of one of the compared descriptions.

At present, logic simulation is the most widely used technique for ensuring the correctness of digital integrated circuits in industry because of its scalability and predictable run-time behavior. This technique is based on verifying a digital system by stimulating inputs of the circuit with binary signal values that propagate in the circuit leading to a corresponding activation of its outputs, whose values must be consistent with the expected ones. In our case a special type of simulation could be applied: guided simulation, when inputs are assigned based on certain information, provided by the design specification. This could reduce in some cases the search space of the simulator. However though the specification of the designed circuit with n inputs would be specified with a small number of multi-output cubes, the overall size of Boolean space covered by them can contain up to 2^n combinations of n input variables. That is why simulation is infeasible for state-of-the-art designs.

In a modern combinational equivalence checking flow based on formal verification approach, both circuits to be verified are transformed into a single circuit called a miter derived by combining the pairs of inputs with the same names and feeding the pairs of outputs with the same names into EXOR gates, which are ORed to produce the single output of the miter. The miter is a combinational circuit with the same inputs as the original circuit and there is constant 0 on its output if and only if the two original circuits produce identical output values under all possible input assignments. To check whether it takes place usually SAT solvers are used requiring their input to be in conjunctive normal form (CNF). This type of solvers can be applied to circuits by converting them into CNF form. Majority of SAT applications derived from circuit representation rely on some version of Tseitin transformation [Tseitin, 1983] for producing CNF of the circuit called as conventional CNF. A circuit-to-CNF conversion has a linear complexity and introduces as many variables as there are primary inputs and gates in the circuit.

In our case when we have a circuit and a system of incompletely specified Boolean functions we cannot construct a miter. The key idea proposed is how to organize testing of all multiple-output cubes whether they are realized by the given circuit. We present a novel SAT based verification method of testing if the given circuit implements all the multiple-output cubes representing the system of incompletely specified Boolean functions based on encoding its multiple-output cubes. The proposed method speeds up the SAT-based procedure due to suggested efficient procedure of logarithmic encoding multiple-output cubes that allows reducing the number of variables to be additionally introduced into the CNF under construction.

Basic definitions

A system $F(X) = \{ f_1(X), f_2(X), \ldots, f_m(X) \}$ (where $X = (x_1, x_2, \ldots, x_n)$) of incompletely specified Boolean functions is represented as mapping of n-dimensional Boolean space E^n into m-dimensional space $\{0,1,–\}^m$, i.e. $F(X)$: $E^n \rightarrow \{0,1,–\}^m$, where the symbol "–" denotes don't-care condition, such a function is not defined over the whole Boolean space E^n. In the case of ISF don't-care points in E^n may differ for different functions. A completely specified Boolean function $f(X)$ realizes an ISF $g(X)$ iff $f(X)$ can be derived from $g(X)$ by assigning either 0 or 1 to each don't-care point of E^n.

An ISF is specified by off-set, on-set and dc-set as subsets of E^n, i.e. sets U_f^0, U_f^1 and U_f^{ds} of cubes ($U_f^1 \cup U_f^0 \cup U_f^{ds} = E^n$). Let us specify a system $F(X)$ of ISFs as a set S_F of multiple-output cubes. A multiple-output cube $(u, t) \in S_F$ is a pair of row ternary vectors u and t (or conjunctions) of dimensions n and m that are called further as its input and output parts. The input part u represents a cube in E^n or a conjunction of some literals (variables $x_i \in X$ or its inversions). The output part t is a ternary vector of values of functions for the cube u or a conjunction of some literals $f_j \in F$. For each $f_j \in F$ the j-th entry t^j of t is 1 or 0 ($t^j = 1, 0$) if all the minterms of the cube u are in the on-set U_{fj}^1 or in the off-set U_{fj}^0 correspondingly; otherwise t^j is don't-care. Representation of a function with multiple-output cubes has the following distinctive feature: cubes u_i and u_j can intersect each other and don't-care value of an element t_i^j of the output part of (u_i, t_i) means that either the function f_j is don't-care on the whole cube u_i or f_j does not take the same definite value on the whole interval u_i, i.e. there exist at least two minterms covered by the cube u_i on which f_j has different definite values.

The system $F(X)$ of ISFs given by the set S_F of multiple-output cubes (u_i, t_i) can be represented in matrix form by a pair of ternary matrices U and T of the same cardinalities. The matrix U contains input parts of multiple-output

cubes from S_F as its rows; similarly matrix T specifies output parts as its rows. For example, ISF system $F(X)$ specified by S_F= {($x_3 x_4 x_5$, f_1), ($x_1 x_2$, $f_1 \bar{f}_2$), ($\bar{x}_2 \bar{x}_3 \bar{x}_4$, $\bar{f}_1 f_2$), ($\bar{x}_2 x_3 \bar{x}_4$, $\bar{f}_2$), ($\bar{x}_2 x_4 \bar{x}_5$, f_2)} is represented as follows:

$$U = \begin{array}{c|ccccc} & x_1 & x_2 & x_3 & x_4 & x_5 \\ & – & – & 1 & 1 & 1 \\ & 1 & 1 & – & – & – \\ – & 0 & 0 & 0 & – & \\ & – & 0 & 1 & 0 & – \\ & – & 0 & – & 1 & 0 \end{array} \quad T = \begin{array}{ccc} f_1 & f_2 & \\ 1 & – & 1 \\ 1 & 0 & 2 \\ 0 & 1 & 3 \\ – & 0 & 4 \\ – & 1 & 5 \end{array} \qquad (1)$$

A CNF represents a completely specified Boolean function as conjunction of one or more clauses, each being in its turn a disjunction of literals. CNF representation is popular among SAT algorithms because each clause must be satisfied (evaluated to 1) for the overall CNF to be satisfied. The SAT problem is concerned with finding a truth assignment of CNF literals, which simultaneously satisfies each of its member clauses. If such an assignment exists the CNF is referred to as satisfiable, and the assignment is known as a satisfying assignment.

Matrix representation of CNF formula C containing k clauses and p distinct variables is a ternary matrix C having k rows and p columns. The entry c_i^j of the matrix in the i-th row and the j-th column is 1, 0 or "–" depending on in what a form (x_j or $\bar{x}_j$) the variable x_j appears or does not appear in the i-th clause of C.

SAT-based verification of logical descriptions with functional indeterminacy

The conventional CNF of a combinational network specifies all combinations of signal values of its terminals that can take place when it functions. The procedure of derivation of conventional CNF is known, it associates a CNF formula with each network gate that captures the consistent assignments between gate inputs and its output. All such gate local CNFs are joined then in the overall network conventional CNF by using the conjunction operation. CNF for a gate representing a local function $y = f(z_1, z_2, \ldots, z_k)$ is based on defining and representing in a CNF form a new Boolean function $\varphi(y, f) = y \sim f(z_1, z_2, \ldots, z_k)$ [Kunz, 2002] that is true in the only case when both functions y and $f(z_1, z_2, \ldots, z_k)$ assume the same value.

When we have conventional CNF C of a combinational network and a system $F(X)$ of ISFs, the arguments and the functions of which correspond to primary inputs and outputs of the network, a problem under discussion is to check whether a given network implements the ISF system. It is true if it takes place for each multiple-output cube of the system. In terms of the network CNF this condition could be reformulated as follows. For every multiple-output cube $(u_i, t_i) \in S_F$ a value assignment satisfying the conjunction $u_i \wedge t_i$ must satisfy the network CNF. Or, this condition could be reformulated in the form more suitable for the task of SAT solving: a network realizes an ISF system $F(X)$ iff for every multiple-output cube $(u_i, t_i) \in S_F$ a value assignment contradicting to it and satisfying the conjunction $u_i \wedge \bar{t}_i$ is unsatisfying assignment for a network CNF. If $u_i = x_1^i x_2^i \ldots x_{ni}^i$ and $t_i = f_1^i f_2^i \ldots f_{mi}^i$ then the CNF P_i specifying the contradiction of the multiple-output cube (u_i, t_i), called as the cube-prohibitive CNF, consists of the following $n_i + 1$ clauses regarding the set $X \cup F$ of variables:

$$P_i(X, F) = x_1^i x_2^i \ldots x_{ni}^i (\bar{f}_1^i \vee \bar{f}_2^i \vee \ldots \vee \bar{f}_{mi}^i).$$

For example, the cube-prohibitive CNF for the multiple-output cube $s_2 = (x_1 x_2, f_1 \bar{f}_2)$ of the mentioned ISF system $F(X)$ (1) contains three clauses: $P_2(X, F) = (x_1)(x_2)(\bar{f}_1 \vee f_2)$. And all cube-prohibitive CNFs for the ISF system are shown in the third column of Table 1.

Appending clauses of P_i to the network conventional CNF C results in CNF $C_{Pi} = C \wedge P_i$. It is not difficult to prove that CNF C_{Pi} is satisfiable iff the network does not realize the i-th multiple-output cube. As to the whole ISF system it is not realized by the network iff it does not implement at least one of its multiple-output cubes, i.e. if the following formula is satisfiable:

$$C_P = C \wedge P = C \wedge (P_1 \vee P_2 \vee \ldots \vee P_l), \qquad (2).$$

where P is the ISF system prohibitive formula that, in general, is not in the form of CNF.

Table 1

Encoding cube-prohibitive CNFs $P_i^k(X, F, W)$of the ISF system (1)

№	$P_i(X, F)$	cube-prohibitive CNFs $P_i(X, F)$										unary codes					logarithmic codes			logarithmic codes		
		x_1	x_2	x_3	x_4	x_5	z_1	z_2	z_3	f_1	f_2	w_1	w_2	w_3	w_4	w_5	w_1	w_2	w_3	w_1	w_2	w_3
1.	P_1	–	–	1	–	–	–	–	–	–	–	0	–	–	–	–	0	0	0	0	0	0
2.		–	–	–	1	–	–	–	–	–	–	0	–	–	–	–	0	0	0	0	0	0
3.		–	–	–	–	1	–	–	–	–	–	0	–	–	–	–	0	0	0	0	0	0
4.		–	–	–	–	–	–	–	–	0	–	0	–	–	–	–	0	0	0	0	0	0
5.	P_2	1	–	–	–	–	–	–	–	–	–	–	0	–	–	–	0	0	1	–	0	1
6.		–	1	–	–	–	–	–	–	–	–	–	0	–	–	–	0	0	1	–	0	1
7.		–	–	–	–	–	–	–	–	0	1	–	0	–	–	–	0	0	1	–	0	1
8.	P_3	–	0	–	–	–	–	–	–	–	–	–	–	0	–	–	0	1	0	–	1	0
9.		–	–	0	–	–	–	–	–	–	–	–	–	0	–	–	0	1	0	–	1	0
10.		–	–	–	0	–	–	–	–	–	–	–	–	0	–	–	0	1	0	–	1	0
11.		–	–	–	–	–	–	–	–	1	0	–	–	0	–	–	0	1	0	–	1	0
12.	P_4	–	0	–	–	–	–	–	–	–	–	–	–	–	0	–	0	1	1	–	1	1
13.		–	–	1	–	–	–	–	–	–	–	–	–	–	0	–	0	1	1	–	1	1
14.		–	–	–	0	–	–	–	–	–	–	–	–	–	0	–	0	1	1	–	1	1
15.		–	–	–	–	–	–	–	–	–	1	–	–	–	0	–	0	1	1	–	1	1
16.	P_5	–	1	–	–	–	–	–	–	–	–	–	–	–	–	0	1	0	0	1	–	–
17.		–	–	–	1	–	–	–	–	–	–	–	–	–	–	0	1	0	0	1	–	–
18.		–	–	–	–	1	–	–	–	–	–	–	–	–	–	0	1	0	0	1	–	–
19.		–	–	–	–	–	–	–	–	–	0	–	–	–	–	0	1	0	0	1	–	–
20.	$Q(W)$											1	1	1	1	1	1	1	–			
21.																	1	–	1			

Encoding of cube-prohibitive CNFs

Converting the formula $P = P_1 \vee P_2 \vee \ldots \vee P_l$ (from 2) into a CNF form could be done always, but that can be hard problem. We propose the method of construction of ISF system prohibitive CNF P that has linear complexity. It is based on encoding multiple-output cubes and their prohibitive CNFs using some coding variables $w_i \in W$. After encoding, cube-prohibitive CNFs $P_i(X, F)$ are transformed into encoded cube-prohibitive CNFs $P_i^k(X, F, W)$ and the formula (2) into

$$C^k{}_P = C \wedge (P_1{}^k \wedge P_2{}^k \wedge \ldots \wedge P_l{}^k) \wedge Q(W), \tag{3}$$

where $Q(W)$ provides that the CNF $C^k{}_P$ will be satisfiable iff at least one CNF $C_{Pi} = C \wedge P_i$ is satisfiable. From now on $Q(W)$ is called as alternative CNF.

When transforming the formula (2) to the CNF form (3) each cube-prohibitive CNF P_i is encoded by a code in the form of a clause $d_i = w_{i1}{}^{\sigma_{i1}} \vee w_{i2}{}^{\sigma_{i2}} \vee \ldots \vee w_{ir}{}^{\sigma_{ir}}$ ($\sigma_{ir} \in \{0,1\}$ and $w_{ir}{}^1 = w_{ir}$, $w_{ir}{}^0 = \bar{w}_{ir}$) where $w_{ij} \in W$. As $P_i(X, F) = x_1{}^i x_2{}^i \ldots x^i_{ni} (\bar{f}_1{}^i \vee \bar{f}_2{}^i \vee \ldots \vee \bar{f}_{mi}{}^i)$ then

$$P_i^k(X, F, W) = (x_1{}^i \vee d_i) \ldots (x^i_{ni} \vee d_i)\,(f_1{}^i \vee \ldots \vee f_{mi}{}^i \vee d_i).$$

All used codes should differ from each other. To formulate the conditions the alternative CNF $Q(W)$ in (3) must satisfy for the chosen encoding of cube-prohibitive CNFs, let us denote by f_Q and f_{di} functions defined by $Q(W)$ and $d_i(W)$ and by $U_Q{}^1$ and $U_{di}{}^1$ – their on-sets that are subsets of the Boolean space E^r of all different minterms of Boolean variables w_i.

Assertion 1. Codes $d_1(W)$, $d_2(W)$, …, $d_l(W)$ of cube-prohibitive CNFs and the appropriate alternative CNF $Q(W)$ must satisfy the following conditions:

1) $(\bigwedge_i f_{di}) \wedge f_Q = 0$ or $(\bigcap_i M_{di}^1) \cap M_Q^1 = \varnothing$;

2) $(\bigwedge_{i \neq j} f_{di}) \wedge f_Q \neq 0$ or $(\bigcap_{i \neq j} M_{di}^1) \cap M_Q^1 \neq \varnothing$ for all $j \in \{1,2,\ldots,r\}$.

The first condition ensures the CNF $P(X, F, W) = (P_1^k \wedge P_2^k \wedge \ldots \wedge P_l^k) \wedge Q$ (the part of the CNF $C^k{}_P$) be unsatisfiable when the analyzed ISF system is realized by the network, i.e. when all cube-prohibitive CNFs $P_i(X, F)$ (not coded) are unsatisfiable with respect to the network CNF C. If the first condition takes place then there exists no value assignment of variables of the set W that can ensure all P_i^k and $Q(W)$ be true. The second condition ensures the CNF $P(X, F, W)$ be satisfiable when the analyzed ISF system is not realized by the circuit, i.e. there exists at least one multiple-output cube, for example the j-th one, that is not realized by it. Thus, a variable value assignment can be found satisfying the j-th cube-prohibitive CNF $P_j(X, F)$ (and thus $P_j^k(X, F, W)$ too). The fulfillment of the second condition guaranties that there exists at least one assignment of coding variables that ensures satisfiability of $Q(W)$ and all cube prohibitive CNFs P_i^k except the j-th one (that is satisfiable by the assumption).

Encoding by codes of unit length

The method of unary encoding has been proposed in [Cheremisinova, 2008], that introduces as many coding variables w_i as there exist multiple-output cubes in the ISF system specification. According to the method the following encoded cube-prohibitive CNF is generated for each multiple-output cube $(u_i, t_i) = (x_1^i x_2^i \ldots x_{ni}^i, f_1^i f_2^i \ldots f_{mi}^i)$:

$$P_i^k(X, F, W) = (x_1^i \vee w_i)(x_2^i \vee w_i) \ldots (x_{ni}^i \vee w_i)(\bar{f}_1^i \vee \ldots \vee \bar{f}_{mi}^i \vee w_i),$$

and the following alternative CNF Q satisfying the Assertion 1 is (see the forth column of Table 1):

$$Q(W) = \bar{w}_1 \vee \bar{w}_2 \vee \ldots \vee \bar{w}_l.$$

Such an alternative CNF Q satisfies the conditions of the Assertion 1 proposed above:

$$f_{di} = w_i,\ f_Q = \bar{w}_1 \vee \bar{w}_2 \vee \cdots \vee \bar{w}_l,\ \bigwedge_i f_{di} = w_1 w_2 \ldots w_l, \text{ so } (\bigwedge_i f_{di}) \wedge f_Q = 0;$$

$$\bigwedge_{i \neq j} f_{di} = w_1 w_2 \ldots w_{j-1} w_{j+1} \ldots w_l, \text{ so } (\bigwedge_{i \neq j} f_{di}) \wedge f_Q = w_1 w_2 \ldots w_{j-1} \bar{w}_j w_{j+1} \ldots w_l \neq 0.$$

In [Cheremisinova, 2008] it has been proven that an ISF system is realized by the network with functional description specified by CNF C iff the following CNF is unsatisfiable:

$$C \wedge (P_1^\kappa \wedge P_2^\kappa \wedge \ldots \wedge P_l^\kappa) \wedge (w_1 \vee w_2 \vee \ldots \vee w_l).$$

Encoding by codes of logarithmic length

The unary encoding of cube-prohibitive CNFs introduces too many additional variables into the CNF that is the input for a SAT solver. The considerable reduction of the number of coding variables and accordingly the reduction of the complexity of resulting CNF can be achieved when to encode cube-prohibitive CNFs with codes of logarithmic length. In that case the minimal number of coding variables could be introduced, it is $r = \lceil log_2 l \rceil$, where l is the number of multiple-output cubes. In this method cube-prohibitive CNFs are encoded with different elementary dizjunctions $d_i = w_{i1}^{\sigma_{i1}} \vee w_{i2}^{\sigma_{i2}} \vee \ldots \vee w_{ir}^{\sigma_{ir}}$ of size r. Accordingly, the following encoded cube-prohibitive CNF is generated for each multiple-output cube $(u_i, t_i) = (x_1^i x_2^i \ldots x_{ni}^i, f_1^i f_2^i \ldots f_{mi}^i)$:

$$P_i^k = (x_1^i \vee d_i)(x_2^i \vee d_i) \ldots (x_{ni}^i \vee d_i)(\bar{f}_1^i \vee \ldots \vee \bar{f}_{mi}^i \vee d_i),$$

An alternative CNF $Q(W)$ for the given encoding of cube-prohibitive CNFs could be found on the grounds of the following assertion, that follows from the Assertion 1.

Assertion 2. If cube-prohibitive CNFs are encoded with codes $d_1(W)$, $d_2(W)$, …, $d_l(W)$ then the appropriate alternative CNF $Q(W)$ is specified by the function $f_Q = \overline{\bigwedge_i f_{di}}$, where f_{di} is the function defined by $d_i(W)$.

Indeed, two conditions mentioned in the Assertion 1 are satisfied:

1) $(\bigwedge_i f_{di}) \wedge (\overline{\bigwedge_i f_{di}}) = 0$;

2) $(\bigwedge_{i \neq j} f_{di}) \wedge \overline{\bigwedge_i f_{di}} = (f_{d1} \dots f_{dj-1} f_{dj+1} \dots f_{dl}))\ (\bar{f}_{d1} \vee \dots \vee \bar{f}_{dj-1} \vee \bar{f}_{dj} \vee \bar{f}_{dj+1} \vee \dots \vee \bar{f}_{dl}) = f_{d1} \dots f_{dj-1}\ \bar{f}_{dj}\ f_{dj+1} \dots f_{dl} \neq 0.$

In conformity with the Assertion 2 the alternative CNF $Q(W)$ will be:

$$\begin{aligned} Q(W) &= \neg((w_{11}^{\sigma 11} \vee w_{12}^{\sigma 12} \vee \dots \vee w_{1r}^{\sigma 1r}) \wedge \dots \wedge (w_{l1}^{\sigma l1} \vee w_{l2}^{\sigma l2} \vee \dots \vee w_{lr}^{\sigma lr})) = \\ &= \neg(w_{11}^{\sigma 11} \vee w_{12}^{\sigma 12} \vee \dots \vee w_{1r}^{\sigma 1r}) \vee \dots \vee \neg(w_{l1}^{\sigma l1} \vee w_{l2}^{\sigma l2} \vee \dots \vee w_{lr}^{\sigma lr}) = \\ &= \bar{w}_{11}^{\sigma 11}\ \bar{w}_{12}^{\sigma 12} \dots\ \bar{w}_{1r}^{\sigma 1r} \vee \dots \vee\ \bar{w}_{l1}^{\sigma l1}\ \bar{w}_{l2}^{\sigma l2} \vee \dots \vee\ \bar{w}_{lr}^{\sigma lr}. \end{aligned} \quad (4)$$

Thus alternative CNF $Q(W)$ will consists of $2^l - l$ clauses that correspond to those combinations of variable w_i values that are not used for encoding of cube-prohibitive CNFs. For example, when using codes

$$\{(\bar{w}_1 \vee \bar{w}_2 \vee \bar{w}_3), (\bar{w}_1 \vee \bar{w}_2 \vee w_3), (\bar{w}_1 \vee w_2 \vee \bar{w}_3), (\bar{w}_1 \vee w_2 \vee w_3), (w_1 \vee \bar{w}_2 \vee \bar{w}_3)\}$$

for encoding five cube-prohibitive CNFs there are three free codes

$$\{(w_1 \vee \bar{w}_2 \vee w_3), (w_1 \vee w_2 \vee \bar{w}_3), (w_1 \vee w_2 \vee w_3)\}$$

that generate alternative CNF $Q(W) = (w_1 \vee w_2)(w_1 \vee w_3)$ (the fifth column of Table 1).

To simplify the alternative CNF $Q(W)$, it to have less clauses and literals, the task arises to choose such $2^l - l$ clauses that would be free from encoding of cube-prohibitive CNFs and could generate simpler CNF $Q(W)$ (if to apply the identical transformations to them).

The method of encoding by codes of logarithmic length

The idea of the proposed method of logarithmic encoding of cube-prohibitive CNFs is to take constant 1 as an alternative CNF $Q(W)$. So we should to select such codes for cube-prohibitive CNFs that 1) they satisfy the conditions of the Assertion 1 and 2) the on-sets of the functions f_{di} corresponding to them cover the whole Boolean space E^r:

1) $\bigwedge_i f_{di} = 0$ or $\bigcap_i M_{di}^1 = \varnothing$;

2) $\bigwedge_{i \neq j} f_{di} \neq 0$ or $\bigcap_{i \neq j} M_{di}^1 \neq \varnothing$ for all $j \in \{1,2,\dots,r\}$;

3) $\bigvee_i f_{di} = 1$ or $\bigcup_i M_{di}^1 = E^r$.

Let us call a clause containing the literals corresponding to all variables from W as a complete one, its size equals $r = |W|$. The function $f_{di}(W)$ specified by a complete clause $d_i(W) = w_{i1}^{\sigma i1} \vee w_{i2}^{\sigma i2} \vee \dots \vee w_{ir}^{\sigma ir}$ takes the value 0 on the only minterm that is $w_{i1}^{\bar{\sigma} i1} w_{i2}^{\bar{\sigma} i2} \dots w_{ir}^{\bar{\sigma} ir}$. And the function $f_{di}(W)$ specified by an arbitrary clause $d_i(W) = w_{i1}^{\sigma i1} \vee w_{i2}^{\sigma i2} \vee \dots \vee w_{ip}^{\sigma ip}$, $p < r$, takes value 0 on 2^{r-p} minterms. Let us denote the set of these minterms as $I(d_i)$.

Assertion 3. Clauses $d_1(W)$, $d_2(W)$, …, $d_l(W)$ may be chosen as codes of cube-prohibitive CNFs, allowing an alternative CNF $Q(W) = 1$, if they satisfy the following conditions:

1) $\bigcup_i I(d_i) = E^r$;

2) each $I(d_i)$ contains at least one minterm (concerned with the clause d_i) which enters into no other set $I(d_j)$.

These conditions follow from three ones sited above. From the Assertion 3 follows that any *i*-th minterm concerned with one of *I* codes of cube-prohibitive CNFs belongs to the only set $I(d_i)$ but each of $2^r - I$ free minterms belongs to one or more sets $I(d_i)$.

The following method of encoding *I* cube-prohibitive CNFs by codes of logarithmic length results from the Assertion 3.

1. Find $r = \lceil log_2 I \rceil$.

2. Take *I* different minterms from E^r as ones concerned with codes, they are accepted as initial codes.

3. Form the set *M* of free minterms of E^r that are concerned with no code.

4. For each initial code $d_i(W)$ having initially the size *r* find the maximal interval in the set $I(d_i) \cup M$. That could be done using Quine–McCluskey method [Zakrevskij, 2008]. The clause corresponding to the found maximal interval will be the code of the *i*-th cube-prohibitive CNF satisfying the second condition of the Assertion 3.

Let us consider the above example of ISF system consisting of five multiple-output cubes. Here three variants of encoding cube-prohibitive CNFs are shown (minterms concerned with codes are on the left from them, free minterms are below the set of concerned minterms):

	w_1	w_2	w_3	w_1	w_2	w_3	w_1	w_2	w_3	w_1	w_2	w_3	w_1	w_2	w_3	w_1	w_2	w_3
P_1	0	0	0	0	0	0	0	1	0	0	–	0	0	0	0	0	0	–
P_2	0	0	1	–	0	1	0	1	1	0	–	1	0	1	0	0	1	–
P_3	0	1	0	–	1	0	1	0	0	–	0	0	1	0	0	1	0	–
P_4	0	1	1	–	1	1	1	0	1	–	0	1	1	1	0	1	1	0
P_5	1	0	0	1	–	–	1	1	0	1	1	–	1	1	1	–	–	1
	1	0	1				0	0	0				0	0	1			
	1	1	0				0	0	1				0	1	1			
	1	1	1				1	1	1				1	0	1			

The first encoding variant is shown in the sixth column of Table 1 and the functions $f_{di}(W)$ realized by the clauses $d_i(W)$ corresponding to the codes of this variant are specified by the following truth table:

w_1	w_2	w_3	f_{d1}	f_{d2}	f_{d3}	f_{d4}	f_{d5}
0	0	0	1	1	1	0	0
0	0	1	1	1	0	1	0
0	1	0	1	0	1	1	0
0	1	1	1	1	1	1	0
1	0	0	1	1	1	0	1
1	0	1	1	1	0	1	1
1	1	0	1	0	1	1	1
1	1	1	0	1	1	1	1

One could compare cods shown in the fifth and the sixth columns of Table 1 that are received by simple logarithmic encoding and by the proposed improved method. It should note too that it is better to encode cube-prohibitive CNFs with more clauses by codes with less definite components.

The optimality criterion when searching for the encoding of cube-prohibitive CNFs is the total number of literals of all codes. It should be noted that from the point of view of the complexity of the encoding searched for, in some cases it is not indifferent which of the minterms will be concerned with some code and which are free. That can influence on the encoding complexity. The matter is each minterm is adjacent to as many minterms as its size is, and it can be merged only with each of them. Thus the upper bound of the number of don't cares in codes is equal to the sum of sizes of free minterms minus the number of pairs of adjacent free minterms. So it is desirable to choose $2^r - I$ free minterms to minimize the number of adjacent among them.

From the procedure of constructing codes of cube-prohibitive CNFs and the Assertion 3 follows that the proposed method can ensure finding optimal codes of logarithmic length. The more is the number of free minterms the less is the size of constructed codes. In the worth case when we have 2^r multiple-output cubes (and correspondingly 2^r cube-prohibitive CNFs) the number of free minterms equals to 0 and the codes are of size *r*.

Results of computer experiments

Two encoding methods have been implemented on C++ programming language: one based on unary encoding and the other based on the proposed here logarithmic encoding. Two similar programs of verification using these methods of encoding cube-prohibitive CNFs were compared on the same set of pseudo-random pairs of descriptions: ISF system and combinational network implementing it. MiniSat solver [MiniSat] has been used to check whether the CNF $C^{\kappa}_{P} = C \wedge (P_1^k \wedge P_2^k \wedge \ldots \wedge P_l^k)$ is satisfiable.

The experiments have shown that the considerable reduction of variables (when using logarithmic encoding) did not bring about substantial speedup of the solution of SAT problem. This fact could take place because of two antagonistic impacts on the efficiency of SAT-solver. On one hand, the reduction of variables should to speed up SAT-solver. For example, if verified ISF system consists of l = 32000 multiple-output cubes then it is necessary to introduce 15 coding variables using logarithmic encoding compared with 32000 using unary encoding. But on the other hand, the increment of the sizes of clauses in the case of logarithmic encoding (in comparison with the unary encoding) complicates SAT-solving.

Acknowledgements

The paper is published with financial support by the project ITHEA XXI of the Institute of Information Theories and Applications FOI ITHEA Bulgaria www.ithea.org and the Association of Developers and Users of Intelligent Systems ADUIS Ukraine www.aduis.com.ua.

Bibliography

[Cheremisinova, 2008] L. Cheremisinova, D. Novikov. SAT-Based Approach to Verification of Logical Descriptions with Functional Indeterminacy. In: Proc of 8th International Workshop on Boolean Problems, Freiberg: September 18–19, 2008, pp. 59–66.

[Drechsler, 2000] R. Drechsler. Formal Verication of Circuits. Kluwer Academic Publishers, 2000.

[Kropf, 1999] T. Kropf. Introduction to Formal Hardware Verification. Springer, 1999.

[Kuehlmann, 2002] A. Kuehlmann, A.J. van Eijk Cornelis: Combinational and Sequential Equivalence Checking. In: Logic synthesis and Verification. Ed. S.Hassoun, T.Sasao and R.K.Brayton. Kluwer Academic Publishers, 2002, pp. 343–372.

[Kunz, 2002] W. Kunz, J. Marques-Silva, S. Malik. SAT and ATPG: Algorithms for Boolean Decision Problems. In: Logic synthesis and Verification. Ed. S.Hassoun, T.Sasao and R.K.Brayton. Kluwer Academic Publishers, 2002, pp. 309–341.

[MiniSat] The MiniSat Page / http://minisat.se/MiniSat.html.

[Tseitin, 1983] G.C. Tseitin. On the Complexity of Derivation in Propositional Calculus. In: Studies in Constructive Mathematics and Mathematical Logic, part 2, 1968, pp. 115–125. Reprinted in J.Siekmann and G.Wrightson, eds., Automation of Reasoning, vol. 2, 1983, Springer-Verlag, pp. 466–483.

[Zakrevskij, 2008] A. Zakrevskij, Yu. Pottosin, L. Cheremisinova. Optimization in Boolean space. TUT Press, 2008, 241 p

Authors' Information

Liudmila Cheremisinova – Principal Researcher, The United Institute of Informatics Problems of National Academy of Sciences of Belarus, Surganov str., 6, Minsk, 220012, Belarus, e-mail: cld@newman.bas-net.by

Dmitry Novikov – Post graduate student, The United Institute of Informatics Problems of National Academy of Sciences of Belarus, Surganov str., 6, Minsk, 220012, Belarus, e-mail: yakov_nov@tut.by