SAT-based verification of implementations of incompletely specified Boolean functions

Liudmila Cheremisinova
United Institute of Informatics
Problems of NAS of Belarus
Minsk, Belarus
e-mail: cld@newman.bas-net.by

ABSTRACT
The problem under discussion is to check whether a given system of incompletely specified Boolean functions is implemented by a logical description with functional indeterminacy that is represented by a system of connected blocks each of which is specified by a system of completely or incompletely specified Boolean functions. SAT-based verification methods are considered which formulate the verification problem as checking satisfiability of a conjunctive normal form. The results of investigation of SAT-based verification methods are given.

Keywords
Computer-aided design, formal verification, satisfiability of a conjunctive normal form

1. 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 [1, 2]. Unfortunately, the capabilities of verification tools has noticeably yielded to the capabilities of design systems, to say nothing of the technological achievements of the semiconductor industry. The problem of verification is to prove the behavioral equivalence of two descriptions of the same device representing different design solutions obtained in a microelectronic device design process.

At present, the approach widely used in industry for the verification of the correctness of integrated circuits is the logical simulation [3] because of its scalability and execution time’s predictability. Since in the general case the simulation cannot provide complete verification (because the upper estimate of the number of test sets is equal to $2^n$, where $n$ is the number of arguments of the verified description), it is insufficient to use only simulation for the solution of the considered problem [4].

In recent decades, formal verification methods that aim to prove the functional identity of projects in a formal way have been developed rapidly as an alternative to simulation-based verification. The methods are based on reduction of the verification problem to a problem of the satisfiability checking of a conjunctive normal form (CNF) [1–3], making it possible to ensure the completeness of the verification, in contrast to simulation methods. The development of these methods was facilitated by the significant progress in solving CNF satisfiability problem observed in the recent decades:

state-of-the-art SAT-solvers [5–7] make it possible to work with CNF that include thousands of disjunctions and variables.

In a typical verification scenario, there are two 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 the desired functionality of the system under design is incompletely specified.

We consider the verification problem for the case, when the desired incompletely specified functionality is given in the form of a system of incompletely specified Boolean functions (ISFs) and the compared functional description represents a multi-block structure with blocks specified by systems of completely or incompletely specified functions. A special case of such a multi-block structure is a combinational network or an ISF system. There is one-to-one correspondence between the arguments and functions of both the compared descriptions. ISFs are specified on intervals (cubes) of values of Boolean input variables, and the intervals are large enough.

2. BASIC DEFINITIONS
An ISF system $F(x) = \{f_1(x), f_2(x), \ldots, f_m(x)\}$ (where $x = (x_1, x_2, \ldots, x_n)$ is a vector) is represented as a mapping of $n$-dimensional Boolean space $B^n$ of vectors $x_i$ into $m$-dimensional space $\{0,1,\ldots\}^m$ of vectors $f_i$, where the symbol “$\ldots$” denotes don’t-care condition. A completely specified Boolean function $f(x)$ implements (covers) 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 $B^n$. An ISF is specified by off-set $U^0$, on-set $U^1$ and dc-set $U^m$ as subsets of $B^n (U^0 \cup U^1 \cup U^m = B^n)$.

Let us specify a system $F(x)$ as a set $I_B$ of multiple-output cubes $(u,t)$ each of which is a pair of ternary vectors $u$ and $t$ of sizes $a$ and $m$. Here the input part of the cube $u$ is a cube in $B^a$ or a set of minterms $b_i$ (elements of $B^a$), and $u$ can be represented by 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 $t$ is 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 $U^1_j$ or in the off-set $U^0_j$ correspondingly; otherwise $t^j$ is don’t-care.

A CNF represents a completely specified Boolean function as a 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 satisfiable 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.

A system \( F(x) \) of ISFs given by the set \( I_F \) 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 cardinals. For example, ISF system \( F(x) \) specified by \( I_F = \{(x_1x_2x_3, f_1), (x_2x_3x_4, f_2), (x_3x_4x_5, f_3)\} \) is shown in Fig. 1a, b).

To test whether the circuit output is 1 or 0, the conventional CNF is produced for it. Once the overall problem is formulated in CNF, a SAT solver can be used to solve it [5–7]. A circuit-to-CNF conversion uses as many variables as there are primary inputs and gates in the circuit: for output of each gate its own internal Boolean variable is introduced. And a local CNF is associated with each gate. Then local CNFs are joined in the overall network CNF \( C(S) \) by the conjunction operation (Fig. 2). The derivation of the local CNF for a gate representing a local function \( y = f(z_1, z_2, ..., z_i) \) is based on defining a new Boolean function \( g(y, f) = y \land f \) [5], that is true in the only case when both functions \( y \) and \( f \) assume the same value. Here are the conventional CNF representations of NOT, n-input AND and OR functions:

\[
\begin{align*}
(z_1 \lor y) & (z_1 \land y); \\
(z_1 \lor y)(z_2 \lor y) ... (z_i \lor y) & (z_1 \land z_2 \land ... \land z_i \land y); \\
(z_1 \lor y)(z_2 \lor y) ... (z_i \lor y) & (z_1 \land z_2 \land ... \land z_i \land y).
\end{align*}
\]

Further we are focusing on the case when the first of descriptions to be compared is an ISF system and the second one is represented by some sort of multi-block structure. We consider two cases: 1) the structure has no indeterminacy and each its block is represented by CNF system; 2) the structure has indeterminacy and each its block is represented by ISF system.

The above traditional approach which constructs a comparing circuit cannot be applied, just as it is, for the declared cases because at least one of the given forms of functional representation may be specified incompletely.

To reduce the verification problem to SAT we construct two CNFs \( P(F) \) and \( C(S) \). CNF \( P(F) \) describes all assignments contradictory to the first form (ISF system) and is called prohibitive CNF of the ISF system. CNF \( C(S) \) describes all possible assignments for the second form (logical network or multi-block structure), and it is called traditionally as conventional CNF in the case of the structure without indeterminacy (combinational circuit) or, otherwise, it is called permissible CNF that is some sort of the conventional CNF for a structure with indeterminacy.

**Assertion.** A multi-block structure \( S \) implements an ISF system \( F(x) \) if and only if CNF \( P(F) \land C(S) \) is unsatisfiable [8, 9].

### 4. SAT Based Verification Cases

#### 4.1. Verification case 1

In the case when a multi-block structure has no indeterminacy, it can be easily transformed into a multi-level combinational network which consists of NOT, AND and OR gates. And a problem under discussion is to verify if a given network implements the ISF system. It is true if it takes place for each multiple-output cube. In terms of network CNF, this condition could be reformulated as follows [8]: for every multiple-output cube \( (u_i, t_i) \in I_F \) a partial value assignment satisfying the conjunction \( u_i \land t_i \) should satisfy the network CNF.

In other words, a network implements ISF system \( F(x) \), iff for every multiple-output cube \( (u_i, t_i) \in I_F \) a partial value assignment satisfying the conjunction \( u_i \land t_i \) (i.e. contradicting to \( u_i \land t_i \)) is unsatisfying assignment for the network CNF. If \( u_i = x_1 \land x_2 \land ... \land x_n \) and \( t_i = f_1 \land f_2 \land ... \land f_m \) 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 \( n + m \) clauses:

\[
P_i(x, f) = x_1 \lor x_2 \lor ... \lor x_n (f_1 \land f_2 \land ... \land f_m).
\]
For example, the prohibitive CNF (Fig. 1) for the multiple-output cube \(s_i = (x_1, x_2, f_1, f_2)\) from \(F(x) = (U, T)\) has three clauses: \(P_i(x, f) = (x_1 \lor f_1) \lor (x_2 \lor f_2)\).

Appending clauses of \(P_i\) to the network CNF \(C(S)\) results in CNF \(C(P_i) = C(S) \land P_i\). It is not difficult to prove that CNF \(C(P_i)\) is satisfiable iff the network does not implement the \(i\)-th multiple-output cube. As to the whole ISF system it is not implemented by the network iff at least one of its multiple-output cubes is not implemented by the network, i.e. if the following formula is satisfiable [10]:

\[
C = C(S) \land P(F) = C \land (P_1 \lor P_2 \lor \ldots \lor P_i),
\]

where \(P(F)\) is the ISF system prohibitive CNF.

To apply any SAT-solver to check whether for the CNF \(C\) a satisfying assignment exists it is necessary to convert the formula \(P(F)\) to a CNF form. Theoretically this could be done always, but it is NP-hard problem. We are interested in a method of construction of ISF system prohibitive CNF \(P(F)\) having linear complexity. Next the method is proposed that is based on encoding multiple-output cubes and their prohibitive CNFs using coding variables \(w_i \in w\). After encoding, prohibitive CNFs \(P(x, f)\) are transformed into encoded prohibitive CNFs \(P_i^e(x, f, w)\) and the formula (1) becomes

\[
C^e = C \land (P_1^e \land P_2^e \land \ldots \land P_i^e) \land Q(w),
\]

where \(Q(w)\) provides that the CNF \(C^e\) will be satisfiable iff at least one prohibitive CNF \(P_i \in P(F)\) is satisfiable. From now on \(Q(w)\) is called as alternative CNF.

When transforming the formula (1) into the CNF form (2) each cube-prohibitive CNF \(P_i\) is encoded by a code in the form of a disjunction \(d = w_1^{a_1} \lor w_2^{a_2} \lor \ldots \lor w_{\ell_1}^{a_{\ell_1}}\) \((\sigma_{\ell_1} \in \{0, 1\})\) and \(w_{\ell_1}^{-1} = w_0\) and \(w_{\ell_1}^0 = w_0\) and \(w_{\ell_1} \in w\):

\[
P_i^e(x, f, w) = (x_1 \lor d_1) \ldots (x_n \lor d_n) (f_1^i \lor \ldots \lor f_{\ell_1}^i \lor d)\quad (3)
\]

To formulate the conditions the alternative CNF \(Q(w)\) in (2) must satisfy for the chosen cube-prohibitive CNF encoding, let us denote by \(f_0^i\) and \(f_j^i\) the functions represented by \(Q(w)\) and \(d(w)\) and by \(U_{0}^{i}Q\) and \(U_{0}^{j}d\) – their on-sets.

**Assertion** [10]. Any alternative CNF \(Q(w)\) for a given encoding of cube-prohibitive CNFs must satisfy the following conditions:

1. \((\land f_0^i) \land f_j^i = 0\) or \((\lor M_{0}^i) \lor M_{0}^j = \emptyset\);
2. \((\land f_0^i) \land f_j^i = 0\) or \((\lor M_{0}^i) \lor M_{0}^j = \emptyset\) for all \(i, j\).

The first condition ensures the CNF \(P(x, f, w) = (P_1^e \land P_2^e \land \ldots \land P_i^e) \land Q\) be unsatisfiable when the circuit implements the analyzed ISF system, i.e. when all cube-prohibitive CNFs \(P(x, f)\) are unsatisfiable. The second condition ensures the CNF \(P(x, f, w)\) be satisfiable when the circuit does not implement the analyzed ISF system, i.e. there exists at least one multiple-output cube, for example \(j\)-th one, that is not realized by it. Thus, a variable assignment can be found satisfying the cube-prohibitive CNF \(P(x, f)\) and \(P_i^e(x, f, w)\), too. Fulfillment of the second condition guarantees that there exists at least one assignment of coding variables that ensures satisfiability of \(Q(w)\) and all cube prohibitive CNFs \(P^e_i\) except the \(j\)-th one (that is satisfiable by the assumption).

Two basic methods of encoding multiple-output cubes (satisfying the above Assertion) have been investigated: encoding by codes of unit [8] and logarithmic length [9]. The first method supposes to introduce as many coding variables \(w_i\) as there exist multiple-output cubes in the ISF system specification \(I_{P}\). The second method introduces the minimal number of coding variables that is \(r = \lceil \log_2(|I_P|) \rceil\). The method of encoding by codes of logarithmic length allows to reduce substantially the number of coding variables as compared with the method of encoding by unary codes, but codes (and CNF clauses) are dense enough.

Further we focus upon increasing efficiency of verification process using unary encoding of multiple-output cubes. Using it \(P_i^e(x, f, w)\) (3) changes for the following encoded form:

\[
P_i^e(x, \ldots, w) = (x_1 \lor w_1)(x_2 \lor w_2)(f_1^i \lor \ldots \lor f_{\ell_1}^i \lor w_{\ell_1}),
\]

and the alternative CNF \(Q\) satisfying the above Assertion becomes: \(Q = w_1 \lor w_2 \lor \ldots \lor w_{\ell_1}\). For example, the fragment of the prohibitive CNF for the ISF system in the matrix form is shown in Fig. 1.b. If we combine the circuit conventional CNF (Fig. 2.b) and the prohibitive CNF (identifying \(y_1, y_2\) with \(f_i, f_j\)) and then carry out the satisfiability test of the resulting CNF, we may make sure that it is nonsatisfiable. Therefore, the circuit (Fig. 2.a) implements the ISF system (Fig. 1.a).

### 4.2. Verification case 2

Here we consider the verification problem for the case, when both compared descriptions are incompletely specified. For example, we have a multi-block structure \(S\) with indeterminacy such as one in Fig. 3 and an ISF system \(F(x)\) such as one in Fig. 1.a.

![Three-block structure with indeterminacy](Image 325x212 to 512x394)

**Figure 3.** Three-block structure with indeterminacy

Just as in the previous section we formulate the verification problem as verifying whether CNF \(C = C(S) \land P(F)\) is satisfiable. Here \(P(F)\) is the ISF system \(F(x)\) prohibitive CNF, \(C(S)\) is some sort of the conventional CNF but only for a multi-block structure with indeterminacy or for an ISF system (that can be considered instead of one-block structure). To distinguish it from the conventional CNF let’s call it further as the **permissible** CNF. The permissible CNF describes the set of admissible combinations of signals on all the nodes of structure blocks, i.e., each set of values satisfying the CNF is admissible for this structure. The permissible CNF \(C(S)\) is the conjunction of permissible CNFs \(C(B)\) of its blocks or permissible CNFs \(C(F)\) of their ISF systems.
Three methods of construction of a permissible CNF for an ISF system are proposed. The first one is based on the paraphrased representation of ISFs [11]. And two methods are based on the application of implicative conditions: implication [12] and implication with coding of conditions [13] methods. Here we dwell on the implication method.

**Assertion** [12]. The permissible CNF $C(G)$ of an ISF system $G(x)$ defined by a set of its multiple-output cubes $s_i = (u_i, t_i)$ $(i = 1, 2, ..., r)$ is generated by the formula:

$$C_i = (u_i \rightarrow t_i) \land (u_i \rightarrow t_i) \land \ldots \land (u_i \rightarrow t_i).$$

The permissible CNF for a multiple-output cube $s_i^G = (u_i, t_i)$ with $u_i = x_1^i x_2^i \ldots x_n^i$ and $t_i = y_1^i y_2^i \ldots y_m^i$ consists of as many clauses as the size of the term $t_i$ is.

$$C_i = (u_i \rightarrow t_i) = \lnot u_i \lor x_1^i \lor \ldots \lor \lnot x_m^i \lor y_1^i \lor \ldots \lor y_m^i.$$

For example, the permissible CNF for the first block of the structure (Fig. 3) consists of five clauses: $(x_1 \lor \lnot x_2 \lor z) \land (x_1 \lor x_2 \lor z) \land (x_1 \lor x_2 \lor \lnot z) \land (x_1 \lor \lnot x_2 \lor \lnot z)$.

The key idea of the proposed method of checking whether an ISF system $P(x)$ is implemented by a multi-block structure $S$ with indeterminacy is as follows. We obtain the prohibitive CNF $P_F$ and the permissible CNF $C(S)$.

**Assertion.** A multi-block structure $S$ with indeterminacy implements an ISF system $P(x)$ iff the CNF $P_F \land C(S)$ is unsatisfiable.

One can make sure after constructing $P_F \land C(S)$ for the ISF system in Fig. 1a and three-block structure in Fig. 3 that there is no satisfying assignment for the CNF. Thus, the structure implements the ISF system.

### 4.3. Organization of SAT problem solving for verification

Three verification methods are proposed [10] that are based on successive, simultaneous and group testing multiple-output cubes from $P_F$. The first method formulates as many SAT problems as the number of cubes are there in an ISF system tested. The second method formulates verification task as the only SAT problem (using coding the cubes as shown above). The third method divides the overall set $P_F$ of multiple-output cubes into groups and formulates as many SAT problems as the number of groups are there.

The investigations of the methods [10] have shown that the group method is more effective because it allows 1) to achieve trade-offs between expenses on forming data for SAT-solver and SAT-solver performance; and thereby 2) to reduce the overall verification time.

## 5. EXPERIMENTAL RESULTS

All the mentioned verification methods have been implemented on C++ programming language. Then the programs were investigated on the sets of pseudo-random pairs of descriptions: ISF system and multi-block structure implementing it (with or without indeterminacy). MiniSat solver [7] has been used in the experiments. The goal of experiments was 1) to compare the competitive methods solving the same task on the same set of examples; 2) to investigate experimentally domains of preferable usage of the proposed methods; 3) to find out the most effective value of group size for group testing verification methods. The experiments have shown that:

1) the group size about 200 gives good enough results: group methods gain stably in efficiency compared with the methods of successive and simultaneous testing of multiple-output cubes, the win gain is about 35% over the method of simultaneous testing;
2) substantial reduction of variables when using logarithmic encoding of multiple-output cubes did not bring about substantial speedup of the solution of verification problem;
3) despite the fact that the implication method is simpler than that of implication with condition coding and gives shorter CNFs, it has smaller speed.

Simulation based verification methods have 60 times greater speed on average than SAT based methods solving the same task.

### REFERENCES