Searching for Optimal Synchronizing Sequences for Testing Logic Circuits

Cheremisinova L.D.
The laboratory of logical design
The United Institute of Informatics Problems of National Academy of Sciences of Belarus
Minsk, Belarus
e-mail: cld@newman.bas-net.by

***Abstract* — The problem under consideration is to find a synchronizing sequence for a logic network with memory. A novel method is proposed that is based on formulation of the task as Boolean satisfiability problem solved with any standard SAT solver. The developed method allows creating a Boolean equation presenting the problem in conjunctive normal form.**

Keywords: design automation; verification; testing

I. Introduction

With the increasing complexity of integrated circuits the problem of ensuring that no faults exist which can cause the device to malfunction is on the rise. The necessity to improve the process of testing is urgent today.

The traditional method for testability verification is based, as a rule, on gate-level simulation. A subset of the functional test patterns (sequences of input vectors) is applied to integrated circuits on the manufacturing testers. The sequential circuit having memory can have internal states that make sequential circuit testing more complex than that of the combinational logic. That is because in typical case the state of internal memory is not known at the beginning of the simulation to test. In order to begin execution of any test sequence it is necessary to bring a sequential circuit under test to some identified state from which it is known how to proceed. Test sequence (the sequence of input vectors) that brings a sequential circuit to some known state regardless of its initial state is the synchronizing sequence. After applying a synchronizing sequence, the final state of the circuit is known without observing the outputs.

In the past and existing literature the problem of homing synchronizing (and homing) sequences generation is usually considered for the case of finite state machines (FSM) [1]. In most applications the underlying FSM is an automaton with abstract state, whose functionality is described by a transition and an output tables (or by state transition graph). Several approaches are used to obtain a synchronizing (and homing) sequences for a FSM. A survey of the main methods can be found in [2]. It should note that finding a shortest synchronizing sequence (with minimum length) is an NP-hard problem.

Unlike that the case considered here concerns to synchronous logical circuits having flip-flop primitives of D type as memory elements. For such a case a method of finding a shortest synchronizing sequence is proposed. The method allows to translate the problem of looking-for synchronizing sequence into Boolean problem expressed by Boolean equation in conjunctive normal form (CNF) and solvable by existing SAT solvers.

II. The problem statement

The proposed method works on a synchronous circuit is imagined in the form of two blocks: combinational logic and flip-flops. The combinational block has two types of inputs: external inputs known as primary inputs and internal inputs, that are supplied by the flip-flops. Similarly, the block has two types of outputs: externally observable and known as primary outputs, and internal outputs, they feed a set of flip-flops and present their excitation functions.

The combinational block is modeled as an interconnect of primitive gates such as AND, OR, NOT, NAND, NOR, XOR. The second block consists of register of the frequently used data flip-flops – D flip-flops.

The task is to be initialize memory elements to some reset states, that is, to find out the synchronizing sequence. Informally, a synchronizing sequence is a sequence of input sets that, when fed the sequential circuit, is guaranteed to bring it to some specified final state.

Let we have a circuit with n primary inputs and m D flip-flops. An input sequence, $\boldsymbol{X} = (\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^k)$ (where $\boldsymbol{x}^i = (x_1^{\,i}, x_2^{\,i}, \ldots, x_n^{\,i})$ is a vector of input signals that is fed the circuit in the time moment i) is said to be a synchronizing sequence of a sequential circuit, if the final circuit internal state after feeding it on the sequence can be determined uniquely regardless of the circuit initial state. We classify a synchronizing sequence $\boldsymbol{X}$ for a circuit as optimal if it is the shortest for all synchronizing sequences accepted by the circuit, that is $\boldsymbol{X}$ is of the shortest length k. It can be not. The task is to find one of the shortest synchronizing sequences.

The behavior of the D flip-flop is described as the behavior of Moore's automaton with two states: 0 and 1. The symbol at its output coincides with the symbol of the state, in which the flip-flop is at the current time instant. The search for known D flip-flop states is reduced to search for predefined values of excitation functions of the flip-flops. So further we are allowed to consider only combinational block having $n + k$ primary inputs $x_1, x_2, \ldots, x_n, x_{n+1}, x_{n+2}, \ldots, x_{n+k}$ corresponding n primary inputs and k flip-flop outputs, and k primary outputs $y_1, y_2, \ldots, y_k$ corresponding to flip-flop inputs defining their excitation functions.

III. SAT-based approach

A CNF represents a Boolean function as conjunction of one or more clauses, each being in its turn a disjunction of literals (Boolean variables or their inversions). Matrix representation of CNF formula is a ternary matrix having a row for each clause and a column for each variable. The SAT problem is concerned with finding a truth assignment of literals which simultaneously satisfies each of CNF clauses. If such an assignment exists the CNF is referred to as satisfiable, and the assignment is called as a satisfying one.

Majority of SAT applications derived from circuit representation produce so called conventional CNF describing all combinations of signal values on all circuit terminals. The procedure of derivation of conventional CNF associates a CNF formula [3] with each circuit gate that captures the consistent assignments between gate primary inputs and outputs. Here are the conventional CNF representations of 2EXOR, nAND, nOR functions:

$$y = z_1 \oplus z_2 \rightarrow (z_1 \vee z_2 \vee \bar{y})(\bar{z}_1 \vee \bar{z}_2 \vee \bar{y})(z_1 \vee \bar{z}_2 \vee y)(\bar{z}_1 \vee z_2 \vee y);$$

$$y = z_1 \wedge z_2 \wedge \ldots \wedge z_n \rightarrow (z_1 \vee \bar{y})(z_2 \vee \bar{y}) \ldots (z_n \vee \bar{y})(\bar{z}_1 \vee \bar{z}_2 \vee \ldots \vee \bar{z}_n \vee y);$$

Библиотека БГУИР

$$y = z_1 \vee z_2 \vee \ldots \vee z_n \rightarrow (\bar{z}_1 \vee y)(\bar{z}_2 \vee y)\ldots(\bar{z}_n \vee y)(z_1 \vee z_2 \vee \ldots \vee z_n \vee \bar{y});$$

The obtained gate local CNFs are joined then in the overall circuit CNF by using the conjunction operation.

Given a conventional CNF formula, the SAT problem may be restated as the problem of finding a variable value assignment that satisfies every clause.

IV. The Method of Search for Synchronizing Sequence

During the search for SAT solution of the synchronizing sequence problem there is no cost mechanisms to favor one solution over another. Thus formulating SAT problem of searching the shortest synchronizing sequence, we are able only to get the answer whether some solution (of the predefined length) of our problem exists. That is why the problem of optimal synchronizing sequence finding is solved regarding a priori assigned synchronizing sequence length.

Thus, we are forced to reformate continuously the folding problem with increasing values of synchronizing sequence length until a satisfiable problem formulation arises. Such a reformulation of the problem based on enumeration of sequence length values seems cumbersome for logic circuits of great size, but below it will be shown that the process of alternate CNF building for increasing synchronizing sequence length is iterative.

At the beginning we search for a synchronizing sequence $\boldsymbol{X}^1 = (\boldsymbol{x}^1)$ of the length 1 and form conventional CNF C^1 for the combinational circuit under test assuming that its primary inputs corresponding to primary inputs of ancestor sequential circuit are $(x_1^{\,1}, x_2^{\,1}, \ldots, x_n^{\,1})$ and primary inputs corresponding to flip-flop outputs of ancestor sequential circuit are $(x_{n+1}^{\,1}, x_{n+2}^{\,1}, \ldots, x_{n+k}^{\,1})$. The values of the last variables are accepted to be don't-care: $x_{n+1}^{\,1}$ = "–", $x_{n+2}^{\,1}$ = "–", …, $x_{n+k}^{\,1}$ = "–" because we don't know their initial values.

If such a manner formed CNF C^1 is satisfiable we will obtain synchronizing sequence $\boldsymbol{X}^1 = (x_1^{\,1}, x_2^{\,1}, \ldots, x_n^{\,1})$ of unit length and the corresponding values of the circuit primary outputs $y_1^{\,1}, y_2^{\,1}, \ldots, y_k^{\,1}$ that define values of excitation functions. Otherwise we should augment the experiment length and form the conventional CNF C^2 to search for a two cycle synchronizing sequence $\boldsymbol{X}^2 = (\boldsymbol{x}^1, \boldsymbol{x}^2)$. So two block combinational circuit will be considered: the first block is the circuit considered on the first step and the second one is a circuit identical to the first one but having primary inputs $(x_1^{\,2}, x_2^{\,2}, \ldots, x_n^{\,2})$ and $x_{n+1}^{\,2} = y_1^{\,1}$, $x_{n+2}^{\,2} = y_2^{\,1}$, …, $x_{n+k}^{\,2} = y_k^{\,1}$. Its k primary inputs $x_{n+1}^{\,2}, x_{n+2}^{\,2}, \ldots, x_{n+k}^{\,2}$ are connected with primary outputs $y_1^{\,1}, y_2^{\,1}, \ldots, y_k^{\,1}$ of the first block circuit, identifying the variables $y_1^{\,1}, y_2^{\,1}, \ldots, y_k^{\,1}$ and $x_{n+1}^{\,2}, x_{n+2}^{\,2}, \ldots, x_{n+k}^{\,2}$, so we use the first ones (have been introduced earlier) instead of the second ones. Then we again verify whether such a formed CNF C^2 is satisfiable to test whether there exists synchronizing sequence $\boldsymbol{X}^2 = (\boldsymbol{x}^1, \boldsymbol{x}^2)$ of the length 2 and so on as long as we will obtain satisfiable CNF or the number of iterations exceeds the limit of iterations predefined in advance. In the last case our search fails and we don't find out any synchronizing sequence.

In Figure 1 the process of augmentation of a circuit copies is shown. For the case of three input two output logic circuit three copies are connected in series to search for synchronizing sequence $\boldsymbol{X}^3 = (\boldsymbol{x}^1, \boldsymbol{x}^2, \boldsymbol{x}^3)$ of the length 3.

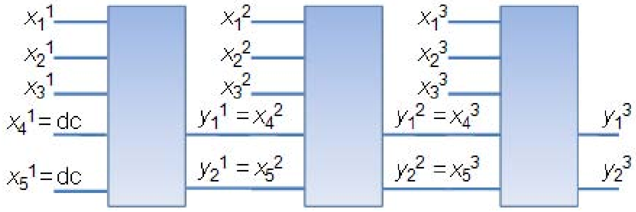

Fig. 1. The process of augmentation of a circuit copies

V. Some Peculiarities of the Method

Here one should draw attention that there are some peculiarities of searching for synchronizing sequence via Boolean satisfiability. They result from existence of don't-care signals in tested circuit that can cause local fragments of conventional CNF to be unsatisfiable though the tested circuit has a synchronizing sequence. That is because don't-care signal run through this circuit fragment from input to output. The characteristic example of such a case is primitive gate, such as XOR that always has don't-care on its output when it has don't-care signal at least in an input. When forming conventional CNF for the tested circuit we may delete fragments associated with such gates taking the values of their output variables to be don't-care.

Moreover there can exist circuit fragment implementing a function like XOR. To do not miss a synchronizing sequence for a circuit having such fragments, when testing CNF satisfiability it should use SAT solvers (for instance, SAT solver PicoSAT [4]) that permit to give proof traces from which it is possible to extract a reason why the tested CNF is erroneous. In that case we can substitute the unsatisfiable fragment with assigning don't-care value to the appropriate fragment output signal, just as we have substituted CNF fragments concerned with XOR gate.

VI. Conclusion

The problem of search for synchronizing sequence for logic circuits with memory elements is considered. A novel reformulation of the problem as the Boolean satisfiability problem solved with any existing SAT-solver was developed.

[1] Z. Kohavi, Switching and Finite Automata Theory, 2nd ed., The McGraw-Hill College, 1978.

[2] D. Lee and M. Yannakakis, "Principles and methods of testing finite state machine – a survey", Proc. of the IEEE, 84(8), Aug. 1996, pp. 1090–1123.

[3] 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.

[4] A. Biere, "PicoSAT Essentials", in Journal on Satisfiability, Boolean Modeling and Computation, vol. 4, 2008, pp. 75–97.