INFORMATICA, 2003, Vol. 14, No. 2, 135–154

© 2003 *Institute of Mathematics and Informatics, Vilnius*

Identifying Legal and Illegal States in Synchronous Sequential Circuits Using Test Generation

Eduardas BAREIŠA, Kęstutis MOTIEJŪNAS, Rimantas ŠEINAUSKAS
Department of Software Engineering, Kaunas University of Technology
Studentų 50–406, LT-3028 Kaunas, Lithuania
e-mail: kestas@soften.ktu.lt

Received: March 2003

Abstract. Identifying legal and illegal states significantly reduces computational complexity of ATPG. A unified framework for identification of the legal and illegal states is presented. Most known methods for identification of the legal and illegal states are interpretable within this framework. New theorems and the resulting procedures for identifying exact collection of legal or illegal states of a circuit are presented. Experimental results demonstrate that exact collection of legal states for some circuits is significantly smaller than collections obtained by backward state search algorithm and by algorithm based on combinational ATPG theorems. The use of the exact collection of legal states allows identifying more undetectable faults. The proposed procedure for identifying of the exact collection of legal states starts from any state of the circuit, builds initially an enlarged collection of legal states and converges rapidly to the exact solution.

Key words: sequential circuits, legal and illegal states, detectability of faults.

1. Introduction

Test development for sequential circuits can be tedious and time-consuming. Automatic test pattern generation (ATPG) tools have attempted to address this problem. ATPG for a given target fault consists of two phases. Fault activation phase establishes a signal value at the fault site opposite to that produced by the fault. Fault propagation phase propagates the fault effect forward by sensitising a path from the fault site to a primary output.

The efficiency of ATPG is largely determined by decision-making and backtracking strategies. Decision-making takes place during justification, the process of assigning values to gate inputs when the logic value of the gate output is known. The process is recursive and ends when the primary inputs are reached. Backtracking takes place as a result of implication, the process by which new values assigned during justification uniquely determine the values on other signals. Because a circuit has reconvergent fan-outs, new signal values during implication can conflict with the values of the signals assigned earlier (Cheng and Krstic, 1999).

One of the complicating factors for ATPG is the existence of illegal states, i.e., states that cannot be reached during the normal operation of a circuit. A sequential test generator may waste a lot of time trying to justify illegal states. This often occurs when the target

fault is undetectable; a situation recognised only when the exhaustive search for a test sequence terminates without finding any. This search process relies on backtracking to recover from dead-end situations. A dead-end is characterised by encountering either an inconsistency (conflicting assignments) or a loop of illegal states.

Determining legal and illegal states is an efficient pre-processing technique of ATPG that significantly reduces computational complexity (Long *et al.*, 2000). For example, knowing that a state is illegal allows for an ATPG algorithm to avoid an expensive and futile state justification process. Also, any fault whose detection requires the circuit to be in an illegal state can be immediately identified as undetectable.

In this work, we describe the forward and backward state search procedures and a procedure for identifying exact collection of the legal states. All procedures are based on applying of the conventional test pattern generator for stuck-at faults of combinational circuits. The combinational test generator is managed by the state sets implemented as PLA's. This feature allows avoiding of circuit representation by BDD. A unified framework for identification of the legal and illegal states is presented. Most known methods for identification of the legal states are interpretable within this framework.

The suggested procedures rely on the combinational part of a sequential circuit only what is very important for efficiency of practical computation. The goal of the experiments is to compare the cardinality of the exact legal states collection with the collection of reachable states identified on the iterative logic array model and to establish what influence it has for identifying of undetectable faults.

2. Preliminaries

A state variable corresponds to the flip-flop of sequential circuit. A state variable has the fixed value 1 or 0 after powering up a circuit. The powering up collection W consists of all possible states after powering up a sequential circuit.

DEFINITION 1. A state variable v is *settable* if there exists an input sequence which sets v to the fixed value starting from any state of powering up collection W.

DEFINITION 2. A state variable is *unsettable* if it is not settable.

DEFINITION 3. A sequential circuit is *synchronizable* if all state variables are settable.

DEFINITION 4. A sequential circuit is *non-synchronizable* if not all state variables are settable.

DEFINITION 5. A sequential circuit is *partially synchronizable* if some state variables are settable.

Operating of a sequential circuit typically begins by applying an initialisation sequence. During this initialisation phase, the output responses are usually ignored. An

initialisation sequence brings a circuit to the initialisation state and determines a predictable behaviour. During the normal operation, the circuit always operates in some finite collections of states that satisfy the following conditions:

1. Any state in a collection is reachable from any other state in that collection.
2. No state in a collection can reach a state outside the collection.

These finite collections of states are typically referred to as the *terminal strongly connected components* (tSCC) of a circuit.

DEFINITION 6. The states of tSCC are *legal* states.

DEFINITION 7. A state is said to be *illegal* if it is not legal state.

DEFINITION 8. A tSCC is *legal* if there exists an initialisation sequence that brings from each initial state of collection W to the state of this tSCC.

A synchronizable circuit has at most one legal tSCC. In sequential circuits undectability and redundancy have different concepts. Several different definitions for sequential undetectability and redundancy can be found in the literature. The definitions of this paper are based on the definitions contained in (Pomeranz and Reddy, 1993; 1994).

DEFINITION 9. Initial states S and S^f are circuit states after powering up the faulty-free and faulty circuit, respectively.

DEFINITION 10. A fault f is said to be detectable if there exists an input sequence I such that for every pair of initial states S and S^f of the fault-free and faulty circuit respectively the response $Z(I,S)$ of the fault-free circuit to the input sequence I is different from the response $Z^f(I,S^f)$ of the faulty circuit at a specific time unit and on a specific primary output.

Note that this definition is based on a single observation time (SOT) approach (Pomeranz and Reddy, 1992).

DEFINITION 11. A fault is said to be undetectable if it is not detectable.

DEFINITION 12. A fault is partially detectable if there exists an initial state S^f of the faulty circuit and an input sequence I such that for every fault-free initial state S the response of the fault-free circuit to I, starting from S, $Z(I,S)$ is different from the response of the faulty circuit starting from S^f, $Z^f(I,S^f)$ at a specific time unit and on a specific primary output.

A collection L of legal states and a collection L^n of illegal states can be exact (complete), enlarged and partial. The exact collection L (L^n) has all legal (illegal) states. A partial collection L (L^n) has only some of the legal (illegal) states. The exact collection L (L^n) is complementary to the exact collection L^n (L). The enlarged collection L (L^n) is

<table>
<tr><td colspan="4">All possible states of the circuit</td></tr>
<tr><td colspan="2">Exact collection L of legal states
(Identification of undetectable faults)</td><td colspan="2">Exact collection L^n of illegal states
(Identification of undetectable faults)</td></tr>
<tr><td colspan="3">Enlarged collection L of legal states
(Identification of undetectable faults)</td><td colspan="1">Partial collection L^n of illegal states
(Identification of undetectable faults)</td></tr>
<tr><td colspan="1">Partial collection L of legal states</td><td colspan="3">Enlarged collection L^n of illegal states</td></tr>
</table>

Fig. 1. The relations between exact, partial and enlarged state collections.

complementary to the partial collection L^n (L) and includes all legal (illegal) and some illegal (legal) states. The use in ATPG of any collection L (L^n) can reduce computational complexity of test generation. However, the identification of undetectable faults relies on the use of the exact or enlarged collection L and on the use of the exact or partial collection L^n . The partial collection L and enlarged collection L^n are not applicable for identification of undetectable circuits. Fig. 1 illustrates the relations between exact, partial and enlarged collections and their application for identification of undetectable faults.

3. Identification of Legal and Illegal States

The existing approaches for identification of collections of legal L and illegal L^n states can be divided into four groups:

- Methods of the group M1 for identification of the legal states rely on a forward state search of the circuit starting from a given set of reset states (Lin *et al.*, 1990; Touati *et al.*, 1990; Cho *et al.*, 1993). The forward state search yields the exact collection L of legal states, which can be complemented to get the exact collection L^n of illegal states. The methods require at least one legal state or initialisation sequence to be known in advance. However, some circuits operate without reset.
- Methods of the group M2 for identification of the legal states rely on three-valued logic simulation and on a forward state search of the circuit starting from an unknown state (Liang *et al.*, 1995). The use of the collection of legal states allows identifying of undetectable faults in the circuit. However, faults that are claimed as undetectable using three-valued simulation may, in fact, be testable (Long *et al.*, 2000).
- Methods of the group M3 for identifying of the legal states rely on the iterative logic array of the circuit (Pomeranz and Reddy, 1994). The forward state search starting from the all-possible initial states yields the enlarged collection L of legal states. The approach is based on so called combinational ATPG Theorems (Agrawal and Chakrather, 1993; 1995) and on so called Pixley's outer envelope computation (Qadeer *et al.*, 1996).

- Methods of the group M4 for identifying of the illegal states rely on eliminating of the states without predecessors and ignoring self-loops (Long *et al.*, 2000). The backward state search starting from all possible states as legal yields a partial collection L^n of illegal states. Methods don't require reset or initialisation sequence.

Methods of the first two groups M1 and M2 either are restricted to the synchronizable circuits only or get an incorrect result. Methods of the last two groups M3 and M4 yield the enlarged collection of legal states. We propose a unified framework for identification legal and illegal states and a procedure for identifying legal states that doesn't require reset or initialisation sequence and yields exact collection of legal states for synchronizable, partially synchronizable and non-synchronizable circuits. The procedure generalises and accumulates the best features of the methods from all four groups.

4. Forward State Search Procedure

The iterative logic array model of a synchronous sequential circuit C consists of copies of the combinational logic of the circuit, connected in such a way that the next state variables of one copy drive the present state variables of the copy to its right (Breuer and Friedman, 1976). Our procedures rely only on one copy of the combinational logic of the iterative logic array model.

First at all, consider how to manage search space of conventional test generator for stuck-at faults of a combinational circuit by means of the PLA's. The extra PLA's connected to the some or all inputs or outputs of circuit restrict search space of test generator. The search space depends on the PLA function. Therefore, the modification of PLA function changes the search space of test generator. Let consider the simple circuit given in Fig. 2.

Let suppose all possible output vectors should be established. The conventional test generator applied to the circuit gives input and output vectors given in Fig. 3, which detect all stuck-at faults of the circuit.

The test generator defines three output vectors 100, 000, 011. Further, these defined output vectors must be forbidden in the output search space by means of PLA in order to get more output vectors by test generator. The modified circuit with PLA is given in Fig. 4.

The PLA function generates on the PLA output the value 1 for the forbidden vectors and generates value 0 in all other cases. Therefore, conventional test generator applied to

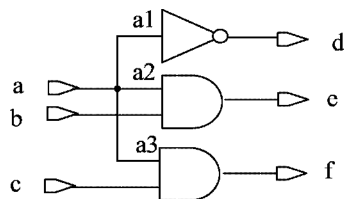

Fig. 2. Circuit *test1*.

Input vectors			Output vectors			Detected faults
a	b	c	d	e	f	
0	1	1	1	0	0	$a_1^1, a_2^1, a_3^1, d^0, e^1, f^1$
1	0	0	0	0	0	$a_1^0, b^1, c^1, d^1, e^1, f^1$
1	1	1	0	1	1	$a_1^0, a_2^0, a_3^0, b^0, c^0, d^1, e^0, f^0$

Fig. 3. Test patterns of circuit *test1*.

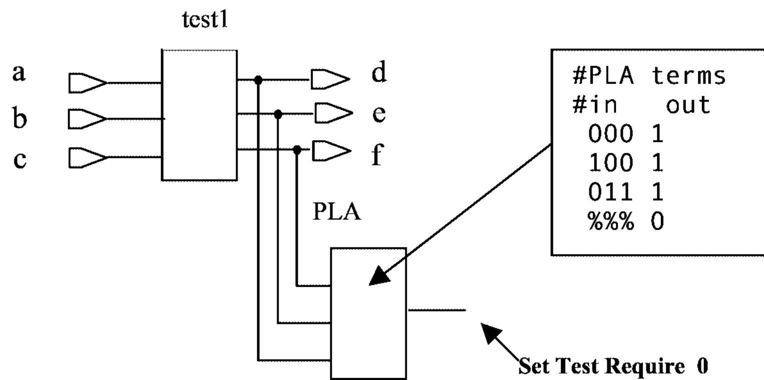

Fig. 4. Circuit *test1* with PLA.

Input vectors			Output vectors			Detected faults
a	b	c	d	e	f	
1	0	1	0	0	1	$a_1^0, b^1, c^0, d^1, e^1, f^0$
1	1	0	0	1	0	$a_1^0, a_2^0, b^0, c^1, d^1, e^0, f^1$

Fig. 5. Test patterns of circuit with PLA.

the circuit with PLA in Fig. 4 and required 0 on the PLA output will generate test vectors, which output vectors are different from vectors forbidden by PLA. Note that in this case the test generator considers all stuck-at faults again. The input and output vectors and detected stuck-at faults of conventional test generator applied to the circuit of Fig. 4 are given in Fig. 5.

Test generator defines two new output vectors 001, 010. Note that in the case of search space restriction using PLA not all stuck-at faults can be detected. Further, new defined output vectors must be forbidden in the output search space including new terms into PLA in order to get new output vectors by test generator. Test generator for the circuit in Fig. 4 with five terms 100, 000, 011, 001, 010 of PLA gives no new vectors. We will prove later that all possible output vectors are defined in this case.

The search space of input vectors can be restricted by PLA in similar way. The PLA with two terms connected to the circuit inputs b and c is given in Fig. 6.

In this case the test generator will consider only such test vectors that have on inputs b and c signal values 10 or 11, i.e., only vectors 010, 110, 011,111 are permitted on the circuit inputs. The test generator would consider only such test vectors that don't have on inputs b and c signal values 10 or 11, i.e., vectors 010, 110, 011, 111 are forbidden on the circuit inputs, if the signal value 0 were required on the output of PLA.

A model for identifying of the states that are reachable (that can be reached) from a given initial state for synchronous sequential circuits is shown in Fig. 7. All the next state variables are assumed to be observable. All the primary inputs and present state variables are assumed to be controllable. A combinational test generation procedure for

single stuck-at fault model is used to generate a test for the single fault injected into combinational logic of the circuit.

A test generation procedure takes into account collection P of present and collection N of next state variables. The states in collections P and N are permitted for the present and next state variables during a search by test generation. The forward state search procedure starts from one or few initial states. At the beginning, the present state search space P consists of initial states and the next state search space N consists of all possible states. The test generation procedure produces test vectors for single faults of combinational logic of the copy. The next state variables of the output test vectors identify a collection R of states reached from the initial states. Then the collection R of reached states reduces the collection N ($N := N \backslash R$) and supplements the collection F of front states ($F := F \cup R$), which is empty at the beginning of iteration ($F := \varnothing$). The states in the collection F reflect the front by expansion of states. A repeated test generation for all faults of combinational logic produces new test vectors and a new collection R of

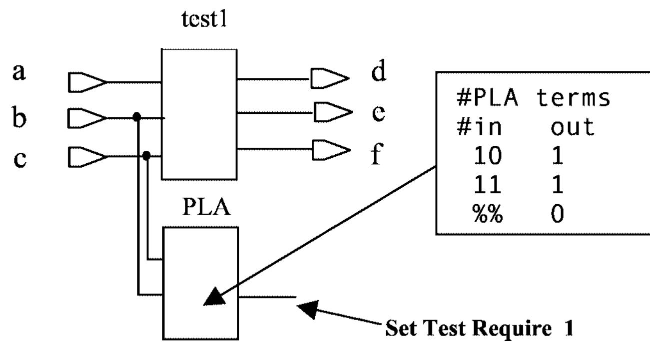

Fig. 6. Circuit *test1* with PLA connected to the inputs b and c.

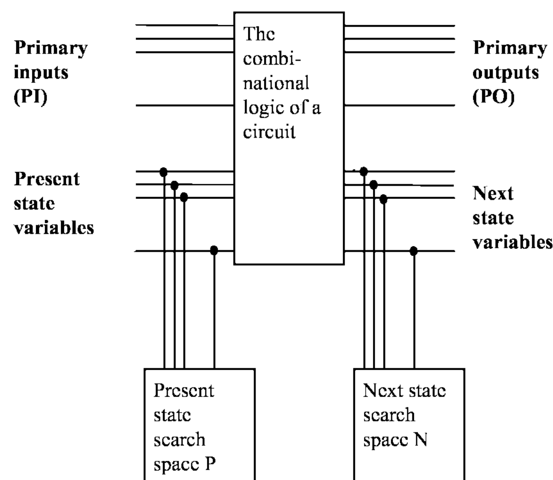

Fig. 7. A model for state search of a circuit.

the reached states, which reduces again the collection N and supplements the collection F. An iteration ends if test generation procedure returns no test vectors. This means that all reachable states are reached from the collection of states P and indicates the end of state search front. New iteration (front) of the forward state search procedure starts with collection $P := F$. The iterations of the procedure terminate if test generation procedure starting from the last front collection F returns no test vectors. This means that no state can be reached from the last front collection F, and a collection complementary to N is the collection of reachable states of the circuit C. An illustration of the first and second fronts for states expansion from the initial state is given in Fig. 8.

A search space of the combinational test generator for stuck-at faults is limited by state collections P and N that are implemented using PLA's. The PLA's manage a conventional test generator for stuck-at faults of a combinational circuit. Let the collection P contains states {11011, 11010, 01011, 01001}. It corresponds to the cubes {1101x, 010x1}. Then the PLA description for SYNOPSYS is the following:

```
#PLA terms
#in              out
1  1  0  1  -    1
0  1  0  -  1    1
%  %  %  %  %    0
```

The synthesised PLA has one output. All inputs of the PLA are connected to the present state variables. The test generator is managed by requesting permanent value "true" on the output of the PLA. This request restricts the search space of test generator by states in the collection P. The requesting of the permanent value "false" on the output of the PLA restricts the search space of test generator by states complementary to the states in the collection P.

The collection of the reached states R can directly supplement the collection P ($P := P \cup R$). In this case the number of state search fronts can't be fixed. The Forward State Search procedure (FSS) for obtaining reachable states of the circuit without fixing the number of the expansion fronts is given in Fig. 9.

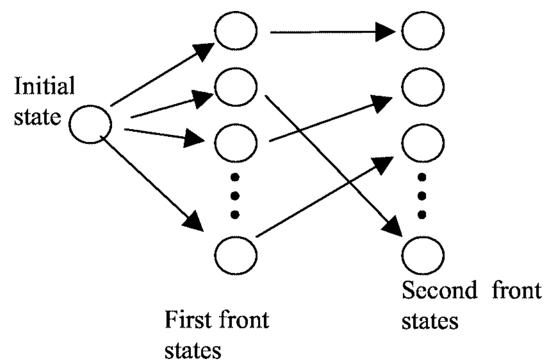

Fig. 8. The fronts of initial state expansion.

(1)	Build the model for state search of synchronous sequential circuit C as presented in Fig. 7.
(2)	$P := \{\text{initial states}\}$, $N := \{\text{all states}\}$.
(3)	Apply to the model test generation procedure for all single stuck-at faults of combinational logic. All the next state variables are assumed to be observable. All the primary inputs and present state variables are assumed to be controllable.
(4)	The next state variables of the test vectors identify some collection R of states reached from the states in collection P.
(5)	If the collection of reachable states R is empty, then collection P consists of reachable states of the circuit C. Otherwise set $P := P \cup R$ ($P := P \backslash \{\text{initial states}\}$ after expansion of the first front), $N := N \backslash R$ and return to Step (3).

Fig. 9. Procedure FSS to obtain states reachable from a given initial state.

Theorem 1. *The procedure FSS identifies all states reachable from a given initial state.*

Proof. Let a circuit C have a reachable state S', which is not in the collection P after procedure FSS has finished. This means that there exists a state $S \in P$, which is a predecessor of S', and exists a test vector that brings C from the state S to S'. Each input vector detects at least one fault of the combinational logic, therefore, this test vector detects at least one fault of the combinational logic of the circuit. However, it is a contradiction to the fact that a test generator did not find any test vector for the faults of combinational logic in case when the input vectors are restricted to have states from the collection P only.

Theorem 2. *The reachable states identified by procedure FSS compose a tSCC if the initial state is a legal one.*

Proof. The states of legal tSCC must satisfy the following conditions:

1. Any state in a collection is reachable from any other state in that collection.
2. No state in a collection can reach a state outside the collection.

A collection of reachable states satisfies second condition according to the Theorem 2. Suppose that a state $S' \in P$ can't be reached from some other legal state $S \in P$. The states S' and S are reachable from the initial state. The initial state can be reached from the legal state S. The state S' can be reached from the initial state also and, therefore, S' can be reached from S, what is a contradiction to the assumption that state S' is not reachable from S.

For example, consider the ITC'99 gate-level benchmark circuit b02 given in Fig. E1. The state diagram of the circuit b02 is shown in Fig. E2, where filled circles represent legal states. The combinational logic of the example circuit is given in Fig. E3. The input names with first letters "psv" represent present state variables and the output names with first letters "nsv" represent next state variables.

The realisation of the proposed model for circuit b02 is presented in Fig. E4, where block b02c represents combinational logic of the circuit, blocks N and P are PLA's and the content of PLA's for the third iteration is given in the box. The results of experiments

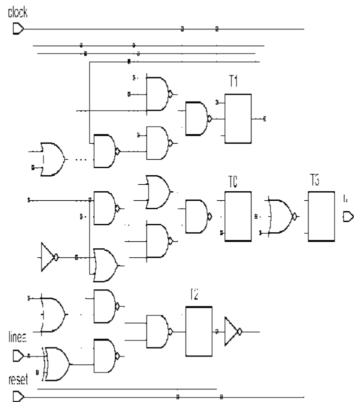

Fig. E1. Circuit b02.

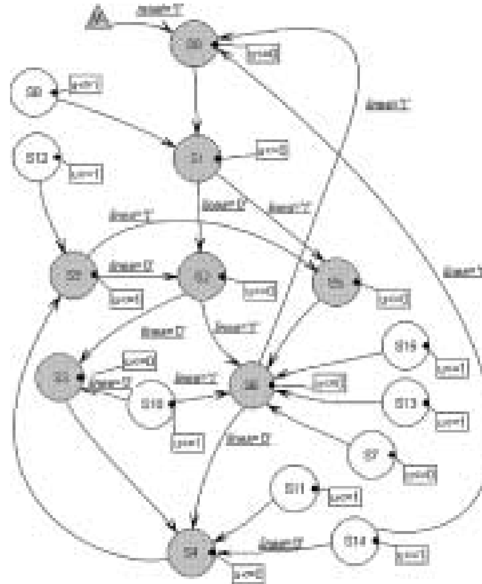

Fig. E2. All state diagram of circuit b02.

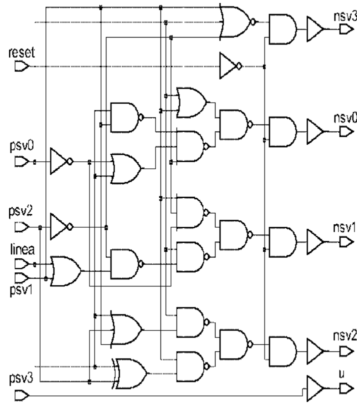

Fig. E3. The combinational logic of circuit b02.

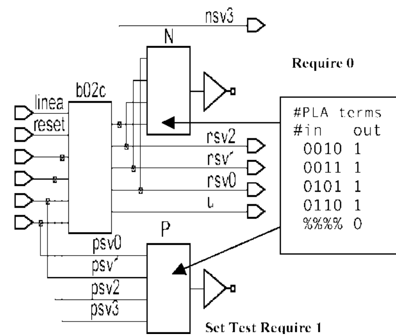

Fig. E4. Realisation of the model for circuit b02.

with Forward State Search Procedure on example circuit are given in Tables E1 and E2, where "Itn." – number of iteration, flip-flop order in state vector – T3, T2, T1, T0, the encoding style of states – binary. The collection P in the third iteration contains states $\{0010, 0011, 0101, 0110\}$. These states in the collection P restrict the input vector search for test generator. The states in the collection N are complementary to the states in the collection P and restrict the output vector search for test generator as well. The conventional test generator finds a test vector for each state transition from a state in collection P to the state in collection N if any fault of combinational part of the circuit is a target.

Table E1
The results of experiments with Forward State Search procedure on example circuit. S1 $\in$ tSCC

Itn.	**1**	**2**	**3**	**4**	**5**
P	{S1}	{S2, S5}	{S2, S3, S5, S6}	{S0, S2, S3, S4, S5, S6}	{S0, S1, S2, S3, S4, S5, S6, S9}
R	{S2, S5}	{S3, S6}	{S0, S4}	{S1, S9}	$\varnothing$

Table E2
The results of experiments with Forward State Search procedure on example circuit. S15 $\notin$ tSCC

Itn.	**1**	**2**	**3**	**4**	**5**	**6**
P	{S15}	{S6}	{S0, S4, S6}	{S0, S1, S4, S6, S9}	{S0, S1, S2, S4, S5, S6, S9}	{S0, S1, S2, S3, S4, S5, S6, S9}
R	{S6}	{S0, S4}	{S1, S9}	{S2, S5}	{S3}	$\varnothing$

5. Backward State Search Procedure

By analogy with FSS procedure the Backward State Search procedure BSS to obtain the states from which initial state can be reached was developed. This procedure is given in Fig. 10. The procedure identifies all states from which a given initial state is reachable.

Theorem 3. *The procedure BSS identifies all states from which a given initial state can be reached.*

Proof. Suppose a circuit C has a state S' from which an initial state can be reached, but this state is not in the collection N upon termination of the procedure BSS. This means that there exists a state $S \in N$, which can be reached from S', and also exists a test vector that brings the state S' to S. This test vector detects at least one fault of the combinational logic. However, it is a contradiction to the fact that procedure BSS did not detect any fault of combinational logic and did not return any test vector for the collection of states N.

(1) Build the model of synchronous sequential circuit C as presented in Fig. 7.
(2) $N := \{\text{initial states}\}$; $P := \{\text{all states}\}$;
(3) Apply to the model test generation procedure for all single stuck-at faults of combinational logic. All the next state variables are assumed to be observable. All the primary inputs and present state variables are assumed to be controllable.
(4) The present state variables of the test vectors identify some collection R of predecessor states, from which the states in the collection N are reachable.
(5) If the collection of predecessor states R is empty, then collection N consists of states from which an initial state of the circuit C is reachable. Otherwise set $N := N \cup R$ ($N := N \backslash \{\text{initial states}\}$ after expansion of the first front), $P := P \backslash R$ and return to Step (3).

Fig. 10. Procedure BSS to obtain states from which an initial state is reachable.

Table E3
The results of experiments with Backward State Search procedure on example circuit. S1 ∈ tSCC

Itn.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
N	{S1}	{S0, S8}	{S0, S8, S14}	{S0, S6, S8, S14}	{S0, S2, S5, S6, S8, S14}	{S0, S1, S2, S5, S6, S8, S9, S10, S13, S14}	{S0, S1, S2, S4, S5, S6, S7, S8, S9, S10, S13, S14}	{S0, S1, S2, S3, S4, S5, S6, S7, S8, S9, S10, S11, S12, S13, S14, S15}
R	{S0, S8}	{S14}	{S6}	{S2, S5}	{S1, S9, S10, S13}	{S4, S7}	{S3, S11, S12, S15}	∅

Table E4
The results of experiments with Backward State Search procedure on example circuit. S15 ∉ tSCC

Itn.	**1**	**2**
N	{S15}	∅
R	∅	∅

The results of experiments with Backward State Search Procedure on example circuit are given in Tables E3 and E4. Note, we always get $N := \{\text{all states}\}$ (see Table E3) if procedure starts from a legal state.

6. Identification of the Exact Collection of Legal States

The reset or initialization sequence identifies at least one legal state of synchronizable circuits. However, no initialisation sequence exists for non-synchronizable and partially synchronizable circuits. Recently Qadeer *et al.* (1996) have proposed an exact algorithm for computing tSCC. A tSCC is obtained by successively pruning the state space and performing forward and backward reachability computations from a state picked at random from the not reduced state space. If the state picked randomly belongs to a tSCC, then the forwardly reachable collection will be a tSCC and the picked state will belong to the backwardly reachable collection; otherwise, the whole backwardly reachable collection can be pruned away. In the worst case the forward or backward reachability computations may require the number of iterations exponential to the number of state variables. We suggest a new procedure for identification of the exact collection of tSCC for synchronizable, non-synchronizable and partially synchronizable circuits if no legal state is known in advance. The procedure we will describe initially gives an enlarged collection of legal states and converges rapidly to the exact solution.

Each state of the circuit has at least one successor. However, some states of the circuit don’t have predecessors. A state is called a deadlock state if from this state no other state can be reached.

DEFINITION 13. The circuit is a securely operating circuit (SO) if an operational behaviour of the circuit doesn't depend on the initial states after powering up.

A circuit is always securely operating circuit if all its states are legal. A circuit with deadlock states can't operate securely. A circuit can occur in the deadlock state after powering up.

Let the procedure FSS identify a collection R^F of states reachable from the initial state and the procedure BSS identify a collection R^B of states from which the initial state is reachable.

Theorem 4. *The* tSCC $= R^F$ *if* $R^F \subseteq R^B$.

Proof. The states of tSCC must satisfy the following conditions:

1. Any state in a collection is reachable from any other state in that collection.
2. No state in a collection can reach a state outside the collection.

A collection R^F of states satisfies the second condition according to Theorem 2. Suppose a state $S' \in R^F$ can't be reached from other state $S \in R^F$. The initial state can be reached from $S \in R^F$, because S also belongs to R^B. A state S' can be reached from the initial state and, therefore, from the state S, what is a contradiction to the assumption that state S' is not reachable from S.

Theorem 5. *The states of the collection* R^B *do not belong to* tSCC *if* $R^F \cap R^B = \varnothing$.

Proof. Suppose a state $S \in R^B$ belongs to tSCC. The states of the tSCC must satisfy the following conditions:

1. Any state in a collection is reachable from any other state in that collection.
2. No state in a collection can reach a state outside the collection.

The states of collection R^F must belong to the same tSCC, because these states are reachable from the state $S \in R^B$. However, the state S is not reachable from the states of collection R^F, because $R^F \cap R^B = \varnothing$. It is a contradiction to the condition 1 for tSCC, and the assumption that the state $S \in R^B$ belongs to a tSCC is wrong. The initial state doesn't belong to the tSCC also, because the collection R^B doesn't include R^F.

Theorem 6. *The collection* R^F *includes all the states of* tSCC *for securely operating circuits.*

Proof. Suppose a state S belongs to a tSCC, but $S \notin R^F$. In this case the state S of tSCC isn't reachable from any state of collection R^F. The procedure FSS terminates if the circuit includes deadlock states or states of tSCC only. Therefore, the collection R^F must include deadlock states if the state $S \notin R^F$. However, it is in contradiction to the condition of Theorem 6 that only securely operating circuits are considered.

Theorem 7. *A securely operating circuit has at most one* tSCC.

Proof. Suppose such a circuit has two terminal strongly connected components tSCC_1 and tSCC_2. A tSCC satisfies the condition that no state in a tSCC can reach a state outside the tSCC. The initial state after powering up can be a state of tSCC_1, and from this state no state in tSCC_2 can be reached. Therefore, the states of tSCC_2 become not legal. It is a contradiction to the fact that the circuit is securely operating one and an operational behaviour doesn't depend on the initial states after powering up.

Theorems 4, 5 and 6 create a basis for procedure Identifying STates (IST) of tSCC, which is shown in Fig. 11.

The procedure IST is applicable for synchronizable, partially synchronizable and non-synchronizable circuits. Note that almost always the last state included into collection R^F by procedure BSS is a legal one. The use of this property creates a possibility mostly successfully to select legal state as an initial state and to avoid using more than two iterations in the procedure. The suggested procedure differs from the algorithm of Qadeer *et al.* (1996) in that it selects the initial state from the smaller collection T, prevents backward search among all illegal states, and early identifies the initial state as illegal. These differences allow increasing the speed of convergence to the exact solution significantly.

The results of experiments with Procedure IST on example circuit are given in Tables E5 and E6.

(1) Build the model for state search of synchronous sequential circuit C as presented in Fig. 7.
(2) $P :=$ {all states}; $N :=$ {all states};
(3) Apply test generation procedure to the model for all single stuck-at faults of combinational logic. All the next state variables are assumed to be observable. All the primary inputs and present state variables are assumed to be controllable.
(4) The present state variables of the test vectors identify some collection T necessary for detecting faults of the combinational logic. Select at random an initial state S from the collection T.
(5) Identify using procedure FSS the collection R^F of states reachable from the initial state S.
(6) If $S \in R^F$, then identify using procedure BSS (among the states of collection R^F) the collection R^B of states from which initial state S is reachable. Otherwise, select other initial state S, which was last included into collection R^F by procedure FSS and return to Step (5).
(7) If $R^F = R^B$, then tSCC:= R^F. Otherwise set $R^F := R^F \backslash R^B$, select other initial state S, which was last included into collection R^F by procedure FSS and return to Step (5).

Fig. 11. Procedure IST to obtain the exact collection of legal states.

Table E5
The results of experiments with procedure IST on example circuit Start from the legal state

Itn.	**1**	**2**
Initial state	S=S4 (legal state)	tSCC= R^F
R^F	{S0, S1, S2, S3, S4, S5, S6, S9}	–
R^B	{S0, S1, S2, S3, S4, S5, S6, S9}	–

Table E6
The results of experiments with procedure IST on example circuit. Start from the illegal state

Itn.	**1**	**2**	**3**
Initial state	S=S8 (illegal state)	S=S3, S3$\in R^F$	tSCC$= R^F$
R^F	{S0, S1, S2, S3, S4, S5, S6, S9}	{S0, S1, S2, S3, S4, S5, S6, S9}	–
R^B	–	{S0, S1, S2, S3, S4, S5, S6, S9}	–

7. Identification of Undetectable Faults

A model for identifying legal states can be applied for identifying undetectable faults. The procedure UNF for identifying UNdetectable Faults based on the states of tSCC is given in Fig. 12.

Lemma 1. *The response $Z(I, S)$ of the fault-free circuit to the input sequence I and the response $Z^f(I, S^f)$ of the faulty circuit are different for at least on one legal state, reached by input sequence I, if fault f is detectable by the sequence I.*

Proof. Suppose responses are different on an illegal state only. In this case the responses depend on the initial circuit state. However, it is a contradiction to the statement of Lemma 1 that fault f is detectable and responses are different for every pair of initial states S and S^f.

Theorem 8. *The faults identified as undetectable by procedure UNF are undetectable in circuit C.*

Proof. Suppose that a fault f identified as undetectable by procedure UNF is detectable in circuit C. According to Definition 11 a fault f is detectable if there exists an input sequence I such that for every pair of initial states S and S^f of the fault-free and faulty circuit, respectively, the response $Z(I, S)$ of the fault-free circuit to the input sequence I is different from the response $Z^f(I, S^f)$ of the faulty circuit at a specific time unit and on a specific primary output. An input sequence brings the circuit from the fully unspecified initial state to the legal one. The fault f can be detectable if there exists a state for which the responses of fault-free and faulty circuits are different on a specific circuit output or on the next state. All legal states are evaluated by procedure UNF. It remains, that an input

(1) Build the model of synchronous sequential circuit C as presented in Fig. 7.
(2) $P :=$ {states of tSCC}; $N :=$ {all states};
(3) Apply to the model test generation procedure for all single stuck-at faults of combinational logic. All the next state variables are assumed to be observable. All the primary inputs and present state variables are assumed to be controllable.
(4) Test generation procedure identifies undetectable faults of combinational logic.

Fig. 12. Procedure UNF to obtain undetectable faults.

sequence has different responses on the illegal state, but it is a contradiction to Lemma 1, which states that an input sequence response are different on at least one legal state.

The results of identifying undetectable faults depend on the cardinality of the state collection considered. The state collection used in the procedure UNF includes only legal states. The proof of the theorems (Pomeranz and Reddy, 1994; Reddy *et al.*, 1999) for the procedures for identifying of undetectable faults using an iterative logic array model is based on the possibility to reach a fault activation state from a state on a cycle. The reachable states on the output of next state variables of an iterative logic array of unlimited length are only the states on a cycle. These states on a cycle include all legal states and may include some illegal states on a cycle. Therefore, the proof of theorem in (Pomeranz and Reddy, 1994) is not valid for the legal states only. For example, experimental results of this paper show that for circuit s713 the collection of reachable states on the output of next state variables of an iterative logic array of unlimited length contains 6461 states on a cycle. This collection of states allows identifying of 38 undetectable faults. The collection tSCC of legal states for this circuit contains 1544 states and allows identifying of 101 undetectable faults.

8. Identification of the Legal and Illegal States from all Possible States

The methods of the groups M3 and M4 apply forward and backward state search from all possible input and output states. The state collection shrinks by each iteration starting from the collection of all possible input (output) states. The procedure terminates if each state of the collection has at least one other state as predecessor in the collection. An implementation (procedure FBS) of both methods is given in Fig. 13.

The results of experiments with Procedure FBS on example circuit are given in Table E7.

Algorithm FILL (Long *et al.*, 2000) eliminates self-loops of states by identifying illegal states from all possible states. The model for eliminating self-loops for unified framework is shown in Fig. 14. Forbidding the same states on the input and output of combi-

(1) Build the model for identifying legal states of synchronizable circuit C as presented in Fig. 7.
(2) $P := \{\text{all possible states}\}$; $N := \{\text{all possible states}\}$; $F := \varnothing$;
(3) Apply test generation procedure to the model for all single stuck-at faults of combinational logic. All the next state variables are assumed to be observable. All the primary inputs and present state variables are assumed to be controllable.
(4) The next state variables of the obtained test vectors identify some collection R of states reached from the states in the collection P.
(5) If the collection of reachable states R is not empty, then set $F := F \cup R$, $N := N \setminus R$ and return to Step (3).
(6) If $F = P$, then collection F consists of states reachable from all possible input and output states. Otherwise set $P := F$; $N := F$; and return to Step (3).

Fig. 13. Procedure FBS to obtain a collection of states with at least one predecessor in the collection.

Table E7
The results of experiments with procedure FBS on example circuit

Itn.	**1.1**	**1.2**	**2.1**	**2.2**
P	ALL	ALL	{S0, S1, S2, S3, S4, S5, S6, S9}	{S0, S1, S2, S3, S4, S5, S6, S9}
N	ALL	ALL$\backslash R$	{S0, S1, S2, S3, S4, S5, S6, S9}	∅
R	{S0, S1, S2, S3, S4, S5, S6, S9}	∅	{S0, S1, S2, S3, S4, S5, S6, S9}	∅
F	{S0, S1, S2, S3, S4, S5, S6, S9}	{S0, S1, S2, S3, S4, S5, S6, S9}	{S0, S1, S2, S3, S4, S5, S6, S9}	$F = P$

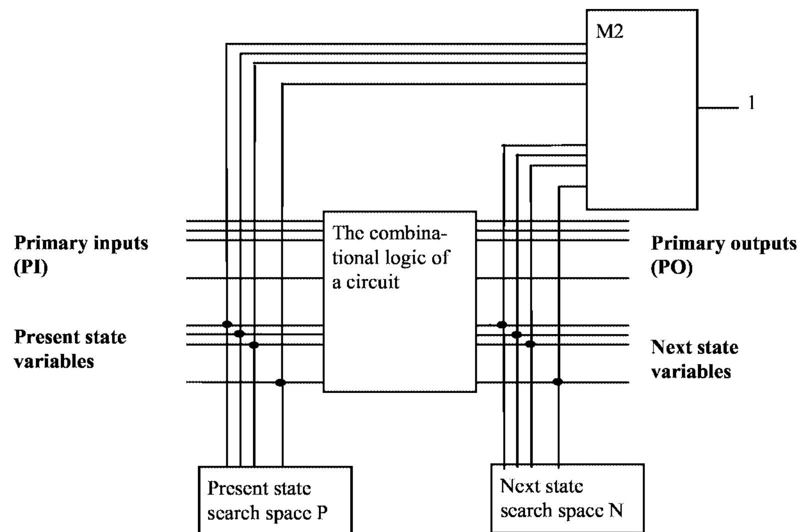

Fig. 14. A model for eliminating self-loops.

national logic eliminates self-loops of states. This is achieved by requesting permanent value "true" on the output of the gate M2 (modulus 2).

9. Experimental Results

The collection of reachable states identified on the iterative logic array model should be theoretically larger than the exact collection of tSCC. The goal of experiments is to compare the cardinality of the exact collection tSCC and of the collection of reachable states identified on the iterative logic array model and to establish what influence it has for identifying of undetectable faults.

The present and next state search space constraints of the model in Fig. 7 are implemented as PLA's. We have used conventional SYNOPSYS test generator for combi-

national circuits. The value of PLA output corresponds to legal states, the complementary value – to the illegal states. In general, some states can be compacted into a single cube. Saving and processing a state cube is better than individually processing the state minterms represented by that cube. Since a cube is a compact representation of a set of values, the number of lines that need to be saved and processed is smaller.

Note that modification of the circuit by routing a global reset signal to all FFs significantly changes the quantity of tSCC states. For example, the original circuit S344 has 1458 legal states, whereas the circuit with global reset signal has 2625 states. Therefore, the cardinality of legal states exact sets given in (Long *et al.*, 2000) is not correct for some circuits.

We used as examples the ISCAS89 Benchmark circuits and b02 and b03 circuits from ITC'99 Benchmarks. As an indication of how accurate are different methods, Table 1 compares the number of legal states and undetectable faults found by procedure IST with those obtained by procedure FBS based on combinational ATPG theorems (Liang *et al.*, 1995; Agrawal and Chakrather, 1993) and by procedure FBS with eliminating self-loops what corresponds to the FILL algorithm (Long *et al.*, 2000). Eliminating of self-loops shrinks a collection of states obtained by backward state search procedure for circuits s344and s349. Table 1 comprises the results of procedure FSS starting from two initial states (010101....) and (101010....), respectively. In most cases the procedure FSS gives almost exact collection of legal states except for the circuits s641 and s713 for the state (101010....).

The columns under heading "IST-tSCC" give the exact number of legal states and the number of undetectable faults. In all cases IST gives almost exact collection of legal states already after the first iteration. The exact collection of legal states is significantly smaller as compared to collections obtained by backward state search algorithm and by algorithm based on combinational ATPG theorems (for circuits s344, s349, s641, s713,

Table 1

Experimental results

Circ.	FBS		FBS–loops		FSS–010101..		FSS–101010..		IST-tSCC		#FF	%
	States	Und	States	Und	States	Und	States	Und	States	Und		
s298	218	35	218	35	254	18	251	30	218	35	14	1.33
s344	5342	7	1487	7	1490	5	1487	7	1487	7	15	4.54
s349	5342	9	1487	9	1490	7	1487	9	1487	9	15	4.54
s386	13	70	13	70	13	70	13	70	13	70	6	20.31
s510	57	0	47	0	47	0	47	0	47	0	6	20.31
s641	6461	0	6461	0	1548	58	6461	0	1544	59	19	0.29
s713	6461	38	6461	38	1548	100	6461	38	1544	101	19	0.29
s820	25	35	25	35	25	35	25	35	25	35	5	78.13
s832	25	51	25	51	25	51	25	51	25	51	5	78.13
s1196	2615	0	2615	0	2615	0	2615	0	2615	0	18	1.00
s1238	2615	68	2615	68	2615	68	2615	68	2615	68	18	1.00
s1488	48	40	48	40	48	40	48	40	48	40	6	75.00
s1494	48	51	48	51	48	51	48	51	48	51	6	75.00
b02	8	0	8	0	8	0	8	0	8	0	4	50.00
b03	*	5	*	5	2058	67	>2500	22	2058	85	30	$1.92e^{-4}$

b03). It conditions less number of undetectable faults for circuits s641, s713, b03. The last two columns show the number of flip-flops and the ratio of the number of legal states to the number of all-possible states in percents for the circuits considered.

However, the sequential static learning approach (Lin *et al.*, 1998) gives almost the same number of undetectable faults as that obtained using exact collection tSCC. It allows to conclude that practical procedures based on the iterative logic array model with sequential static learning give a solution very close to the solution produced using the exact collection of tSCC for the circuits considered.

10. Concluding Remarks

We have presented a unified framework for identification legal states and undetectable faults in synchronous sequential circuits. Most known methods for identification legal states and undetectable faults are interpretable within this framework. The framework uses the combinational logic of sequential circuit and PLA's for limiting of search space. We have presented new theorems and the resulting procedures for identifying exact collection of legal states and for identifying undetectable faults. We proved that a securely operating circuit has at most one terminal of strongly connected components (tSCC). The forward and backward search procedures FSS and BSS give all possible states reachable from a given initial state and all possible states from which an initial state is reachable. The theorems create a basis for procedure IST for obtaining the exact collection of legal states from any initial state for synchronizable, non-synchronizable and partially synchronizable circuits. A unified framework allows implementing of algorithm based on combinational ATPG theorems, implementing algorithms for backward state search starting from all possible states and eliminating self-loops. Experimental results were presented to demonstrate that the exact collection of legal states is significantly smaller than the collections obtained by backward state search algorithm and by algorithm based on the combinational ATPG theorems. The proposed procedure IST gives almost exact collection of legal states after the first iteration already.

References

Agrawal, V.D., and S.T. Chakrather (1993). Combinational ATPG theorems for identifying untestable faults in sequential circuits. In *Europ. Test Conf.*, pp. 249–253.

Agrawal, V.D., and S.T. Chakrather (1995). Combinational ATPG theorems for identifying untestable faults in sequential circuits. *IEEE Trans. on Computer-Aided Design*, 1155–1160.

Breuer, M.A., and A.D. Friedman (1976). *Diagnosis & Reliable Design of Digital Systems*, Computer Science Press.

Cheng, K.T., and A. Krstic (1999). Current directions in automatic test-pattern generation. *Computer*, 58–64.

Cho, H., G.D. Hachtel and F. Somenzi (1993). Redundancy identification/removal and test generation for sequential circuits using implicit state enumeration. *IEEE Trans. on CAD*, **12**(7), 935–945.

Liang, H.-C., C.L. Lee and J.E. Chen (1995). Identifying untestable faults in sequential circuits. *IEEE Design and Test of Computers*, 14–23.

Lin, B., H.J. Touati and A.R. Newton (1990). Don't care minimization of multi-level sequential logic networks. In *Proc. Int. Conf. on Computer-Aided Design*. pp. 414–417.

Lin, X., I. Pomeranz and S.M. Reddy (1998). On finding undetectable and redundant faults in synchronous sequential circuit. In *Proc. Intl. Conf. on Computer Design*.
Long, D.E., M.A. Iyer and M. Abramovici (2000). FILL & FUNI: algorithms to identify illegal states and sequentially untestable faults. *ACM Transactions on Design Automation of Electronics System*, **5**(3).
Pomeranz, I., and S.M. Reddy (1992). The multiple observation time test strategy. *IEEE Trans. on Computers*, **41**(5), 627–637.
Pomeranz, I., and S.M. Reddy (1993). Classification of faults in synchronous sequential circuits. *IEEE Transactions on Computers*, 1066–1077.
Pomeranz, I., and S.M. Reddy (1994). On identifying untestable and redundant faults in synchronous sequential circuits. In *12th IEEE VLSI Test Symposium*. pp. 8–14.
Reddy, S.M., I. Pomeranz, X. Lin and N. Z. Basturkmen (1999). New procedures for identifying undetectable and redundant faults in synchronous sequential circuits. In *Proc. 17th VLSI Test Symposium*. pp. 275–281.
Qadeer, S., R.K. Brayton, V. Singhal and C. Pixley (1996). Latch redundancy removal without global reset. In *Proc. Int. Conf. on Computer Design*.
Touati, H., H. Savoj, B. Lin, R.K. Brayton and A. Sangiovanni-Vincentelli (1990). Implicit state enumeration of finite state machines using BDD's. In *Proc. Int. Conf. on Computer-Aided Design*. pp. 130–133.

E. Bareiša graduated from Kaunas Polytechnic Institute in 1987. Currently he is in position of assoc. professor at Software Engineering Department, Kaunas University of Technology, Lithuania. His research interests include high-level synthesis and VLSI test generation.

K. Motiejūnas graduated from Kaunas Polytechnic Institute in 1981. Currently he is in position of assoc. professor at Software Engineering Department, Kaunas University of Technology, Lithuania. His research interest is test generation for digital circuits.

R. Šeinauskas graduated from Kaunas Polytechnic Institute in 1972. He got his Doctor habilitus from Leningrad Energetics Institute in 1982. Currently he is in position of professor at Software Engineering Department, Kaunas University of Technology, Lithuania. His research interest is VLSI test generation.

Legalių ir nelegalių būsenų nustatymas sinchroninėse nuosekliose schemose remiantis testų sudarymu

Eduardas BAREIŠA, Kęstutis MOTIEJŪNAS, Rimantas ŠEINAUSKAS

Legalių ir nelegalių būsenų nustatymas reikšmingai sumažina testų sudarymo skaičiavimų apimtį. Straipsnyje pateiktos legalių ir nelegalių būsenų nustatymo sinchroninėse nuosekliose schemose procedūros. Įrodyta, kad darbe pasiūlytos procedūros leidžia apskaičiuoti tikslias legalių ir nelegalių būsenų aibes. Eksperimentiniai rezultatai demonstruoja, kad tikslių legalių ir nelegalių būsenų aibių panaudojimas testų generavime įgalina nustatyti daugiau netikrinamų gedimų nei anksčiau žinomi metodai.