On Generating Compact Test Sequences for Synchronous Sequential Circuits

Irith Pomeranz and Sudhakar M. Reddy +
Electrical and Computer Engineering Department
University of Iowa
Iowa City, IA 52242
U.S.A.

Abstract

We present a procedure to generate short test sequences for synchronous sequential circuits described at the gate level. Short test sequences are important in reducing test application time and memory requirements. The proposed procedure constructs a test sequence using a combination of fault-independent and fault-oriented criteria. Experimental results are presented to demonstrate its effectiveness.

1. Introduction

We address the problem of generating short (compact) test sequences for synchronous sequential circuits described at the gate level. Short test sequences are important to reduce test application time and memory requirements. The problem of generating compact test sets for combinational circuits was considered, e.g., in [1,2]. It was shown that it is possible to generate compact test sets for combinational circuits in a cost-effective way, achieving significant reductions in test set size by using simple heuristics. With the exception of [3,4], test generation procedures for synchronous sequential circuits generally concentrate on achieving high fault coverage, and either ignore test sequence length, or target it as a secondary objective. For example, overlapping of test sequences for various faults is used to reduce the overall sequence length in most test generation procedures. In [3], an additional feature called "freeze" is used to stop the clocking of flip-flops and freeze the state of the circuit. With the circuit kept at a state S, several primary input combinations are applied to detect faults that require the same values on the state variables but different primary input values. In [4], post-processing of test sequences already generated by a test generation procedure is used to reduce test sequence length.

In this work, we describe a test generation procedure for non-scan synchronous sequential circuits, that directly attacks the problem of generating compact test sequences. The underlying concept of the procedure proposed here is similar to the approach used in [5] for combinational circuits. In [5], a fault-independent procedure is first used to generate tests that detect large numbers of faults. The faults that remain undetected after the first phase are then considered under a fault-oriented test generation procedure that targets the remaining faults one at a time. The test generation procedure proposed here also uses a fault-independent phase and a fault-oriented phase. However, the following differences exist between the proposed procedure, that targets sequential circuits, and [5] that targets combinational circuits. In combinational circuits, a single input combination is required to detect a fault. Consequently, path sensitization by a single input combination is used in [5] as an objective for a fault-independent procedure. In synchronous sequential circuits, sequences of input combinations are required to detect a fault. Therefore, the objectives of the fault-independent phase need to be redefined to accommodate fault activation and fault propagation sequences. The main goal in defining these objectives is to ensure that if a sequence of k input patterns is needed before the next fault is detected, then these input patterns are selected so as to (1) maximize the number of faults detected after k input patterns are applied, and (2) minimize the test sequence length required to detect additional faults following the first fault detection. A fault-independent procedure suitable for synchronous sequential circuits is described in this work. A fault independent procedure for sequential circuits is mentioned in [3], however, its details are omitted and are not available in the literature. Another difference between the procedure proposed here and the procedure of [5] is that the fault-independent and the fault-oriented phases of the proposed procedure are interleaved, in order to ensure that test sequences for the faults that require fault-oriented test generation are overlapped as much as possible with the test sequences of the other faults. The proposed procedure constructs a test sequence by considering the circuit at consecutive time units, selecting an input combination to be applied at time i before considering time $i+1$. The criteria for selecting each input combination and for switching between the fault-independent and the fault-oriented phases are designed so as to achieve small test lengths. Due to the selection of input combinations one at a time, the proposed procedure is similar in structure to test generation procedures that use forward time processing [6,7]. It is also similar to [6] in the use of cost measures to select the input combinations in the test sequence and in considering subsets of faults. The main difference between the proposed procedure and [6] is that [6] is a simulation-based method, whereas the proposed procedure contains a fault-oriented test generation procedure. When the simulation-based approach is not effective in advancing the test sequence towards the detection of new faults, input vectors are selected based on a deterministic test generation procedure. In this way, the test sequence length can be kept low and complete fault coverage can be guaranteed.

Experimental results presented in this work demonstrate the effectiveness of the proposed approach in generating short test sequences. We first develop the test generation procedure for circuits that have a fault free reset mechanism. Then, we show how it can be applied to circuits without reset, or when reset faults are considered. We compare the results obtained for circuits with a fault free reset mechanism to those of [8] and to those of VERITAS [9]. Like [8,9], our procedure results in com-

+ Research supported in part by NSF Grant No. MIP-9220549, and in part by NSF Grant No. MIP-9357581

plete fault coverage for all circuits considered. The results obtained when reset does not exist are compared to those of [10], [11] and [12]. We do not compare our results with [3] that uses the additional feature of "freeze" since the fault coverages reported in [3] are not complete.

The paper is organized as follows. In Section 2, we describe a fault-independent procedure to generate tests for a synchronous sequential circuit. The fault-independent procedure cannot guarantee detection of every detectable fault. In Section 3, we consider the incorporation of a fault-oriented test generation procedure into the procedure of Section 2, to guarantee detection of every detectable fault. In Section 4 we present experimental results. Concluding remarks are given in Section 5.

2. A fault-independent procedure

In this section, we first describe a fault-independent procedure for synchronous sequential circuits that have small numbers of primary inputs. We then extend the procedure to circuits with arbitrary numbers of primary inputs.

2.1 Circuits with small numbers of primary inputs

The fault-independent procedure constructs a test sequence T as follows. Initially, T is empty and the fault free and faulty circuits are in their reset states. In step i, the i-th input combination of T, t_i, is determined. The input combination t_i is selected such that the benefit from adding t_i to T is maximal. Consider the following example.

Example: ISCAS-89 benchmark circuit *s* 27 is shown in Figure 1. Circuit lines are identified by integer values. We consider the faults {2 s.a.0, 3 s.a.0, 4 s.a.0, 5 s.a.0, 6 s.a.1, 7 s.a.0} (a reduced fault list is considered in this example for illustration purposes). Let 000 be the reset state of the circuit. Primary output values and next-states are given in Table 1 for input combinations 0000, 0100 and 0111. Each input combination c is applied to the fault free circuit and to the circuit in the presence of every one of the faults above, starting from state 000. Under input combination 0000, the fault 6 s.a.1 is detected (the fault-free/faulty outputs are 1/0). In the presence of every other fault, the faulty circuit remains in state 000, and the fault is not detected. Under input combination 0100, only 6 s.a.1 is detected. In addition, the fault 2 s.a.0 is activated (the fault-free/faulty next state is 001/000). The other four faults are neither detected nor activated, however, the faulty circuits go from state 000 to a new state, 001. In choosing between input combinations 0000 and 0100, 0100 has the advantage that it activates an additional fault and allows exploration of a new state for three other faults. Under input combination 0111, both 2 s.a.0 and 6 s.a.1 are detected. In addition, 3 s.a.0 is activated (the fault-free/faulty states are 000/001). In selecting between 0100 and 0111, we prefer 0111 since it detects an additional fault. □

For circuits with small numbers of primary inputs, such that all primary input combinations can be explicitly considered, three parameters are computed for each input combination c to determine the benefit of adding it to T. To define them, we use the following terminology.

A faulty circuit is said to be brought to an *activation state* by an input sequence T if the state of the faulty circuit is different from the state of the fault free circuit after T is applied starting from the reset state. For example, after applying to *s* 27 the input sequence (0100), comprised of a single input combination 0100, the fault free state is 001 and the faulty state in the presence of 2 s.a.0 is 000 (cf. Table 1). Thus, the faulty machine

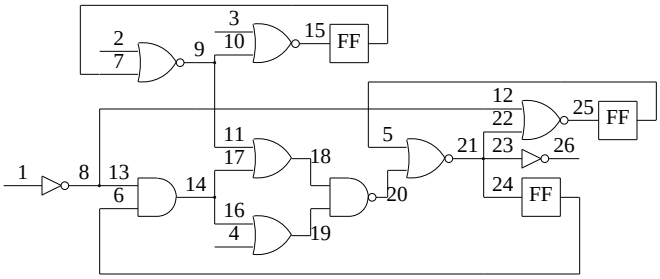

Figure 1: ISCAS-89 benchmark circuit *s* 27

Table 1: Responses and next states of *s* 27 **(**$i = 0$**)**

	0000		0100		0111	
fault	next-state	primary outputs	next-state	primary outputs	next-state	primary outputs
fault-free	000	1	001	1	000	1
2 s.a.0	000	1	000	1	010	0
3 s.a.0	000	1	001	1	001	1
4 s.a.0	000	1	001	1	000	1
5 s.a.0	000	1	001	1	000	1
6 s.a.1	010	0	011	0	010	0
7 s.a.0	000	1	001	1	000	1

in the presence of 2 s.a.0 is brought to an activation state.

A faulty circuit is said to be brought to a *new state* by an input sequence T if the composite fault-free/faulty state at the end of T is not reached by the fault-free/faulty circuits before the end of T. For example, considering *s* 27 and the fault 3 s.a.0, the state 001/001 reached by the fault-free/faulty circuits after applying the input sequence (0100) is a new state. If, in addition, the final state under T is an activation state, then the faulty circuit is said to be brought to a *new fault activation state*.

Using this terminology, the following parameters are defined with respect to a test sequence T.

$N_{\det}(c)$ is the number of yet-undetected faults which are detected by the sequence $T{\cdot}c$ ($T{\cdot}c$ is the sequence obtained by adding c at the end of T).

$N_{new\_activ}(c)$ is the number of yet-undetected faults which are brought to a new activation state by the sequence $T{\cdot}c$.

$N_{new}(c)$ is the number of yet-undetected faults which are brought to a new state by the sequence $T{\cdot}c$.

When selecting an input combination c to be added to T, we require that $N_{\det}(c)$, $N_{new\_activ}(c)$ and $N_{new}(c)$ be maximized, in this order. The requirement to maximize $N_{\det}(c)$ results from our goal to detect as many faults as possible by the fault-independent procedure. The reason for maximizing $N_{new\_activ}(c)$ is that by exploring as many activation states for various faults as possible, we potentially drive as many of the faults as possible closer to states where they can be detected. The reason for maximizing $N_{new}(c)$ is that by exploring as many different states for various faults as possible, we potentially drive as many of the faults as possible closer to fault activation states, and eventually to fault detection. The fault-independent procedure is summarized next.

Procedure 1: Fault-independent test generation for a circuit with a small number of primary inputs

(1) Set $i = 0$. Set $T = \phi$. Set F to contain every target fault.

(2) For every input combination c, compute $N_{\det}(c)$, $N_{new\_activ}(c)$ and $N_{new}(c)$.

(3) Set C to contain every primary input combination.
Remove from C every input combination c' such that $N_{\det}(c') < \max\{N_{\det}(c): c \in C\}$.
Remove from C every input combination c' such that $N_{new\_activ}(c') < \max\{N_{new\_activ}(c): c \in C\}$.
Remove from C every input combination c' such that $N_{new}(c') < \max\{N_{new}(c): c \in C\}$.

(4) Randomly select an input combination $c_0 \in C$. Add c_0 to T (i.e., set $T = T \cdot c_0$) and drop from F every fault detected by the new sequence T.

(5) Set $i = i+1$. If $F \neq \phi$ and i does not exceed a predetermined bound, go to Step 2.

2.2 Circuits with large numbers of primary inputs

In Section 2.1 we assumed that every primary input combination can be considered explicitly, and we established criteria for selecting the next input combination to be included in the test sequence T. For circuits with large numbers of primary inputs, one of the following approaches can be taken.

It is possible to simulate a large number of randomly determined primary input combinations and select the best one. Alternatively, the methods proposed in [6] can be used.

It is also possible to generate test sequences for several yet-undetected faults, starting from the final state reached under the partially constructed test sequence T, and use the first input combinations of these sequences as candidates for inclusion in T. This method has a high complexity, since it requires fault-oriented test generation for several faults, however, it has the advantage of ensuring that every primary input combination selected advances at least one fault towards detection.

Alternatively, it is possible to extend the approach taken in [1] to sequential circuits. The test generation procedure of [1] proceeds as follows. A target fault f, called the primary target fault, is selected from the fault list. A test pattern t is generated for f, leaving as many primary inputs as possible unspecified. Additional primary inputs are then unspecified using the maximal compaction heuristic of [1]. Additional target faults, called secondary target faults, are then selected. When a secondary target fault f' is considered, an attempt is made to specify unspecified primary inputs in t such that the resulting test would be a test for f'. The process continues until all secondary target faults are considered. For synchronous sequential circuits, the test t for the primary target fault may be a sequence of primary input combinations, and its extension for a secondary target fault may involve setting additional primary input values in one or more time units. The fault-independent procedure is replaced in this case by a test generation process targeting secondary faults. Alternatively, it is possible to consider one time unit at a time, as in Procedure 1.

In our experiments we select each input pattern of the test sequence from 100 randomly generated input patterns.

3. Ensuring complete fault coverage

The fault-independent procedure of Section 2 (Procedure 1) does not guarantee complete fault coverage. It also suffers from the following limitation. Consider a fault f under a partially constructed test sequence $T = (t_0, t_1, \cdots, t_{k-1})$. Let the fault free circuit and the faulty circuit in the presence of f go through the following transitions under T.

Fault free: $S_0 \xrightarrow[t_0]{} S_1 \xrightarrow[t_1]{} S_2 \xrightarrow[t_2]{} \cdots \xrightarrow[t_{k-1}]{} S_k$.

Faulty: $Q_0 \rightarrow Q_1 \rightarrow Q_2 \rightarrow \cdots \rightarrow Q_k$.

Suppose that f can be detected by a primary input combination t if the fault-free/faulty circuits are brought to states S_i/Q_i for some $i < k$. Let $t_i \neq t$. An input combination $t_i \neq t$ may be selected by Procedure 1, e.g., if t_i is more effective than t in detecting other faults after $(t_0, \cdots, t_{i-1})$ is applied to the circuit (i.e., $N_{\det}(t_i) > N_{\det}(t)$ when $T = (t_0, t_1, \cdots, t_{i-1})$). Thus, although the state S_i/Q_i is reached after applying the subsequence $(t_0, \cdots, t_{i-1})$, f is not detected since $t_i \neq t$ is selected next. In this case, the test sequence may have to bring the fault-free/faulty circuits back to states S_i/Q_i before the fault f can be detected. However, the measures $N_{\det}(c)$, $N_{new\_activ}(c)$ and $N_{new}(c)$ may all be zero for every c until state S_i/Q_i is reached again. Thus, they may not help in guiding Procedure 1 back to this state. A related problem occurs when $N_{\det}(c) = 0$ and $N_{new\_activ}(c)+N_{new}(c) \leq 1$ for every primary input combination c. In this case, any input combination c that may be selected is potentially effective only for one fault. In fact, even if $N_{\det}(c) = 0$, $N_{new\_activ}(c) \leq 1$ and $N_{new\_activ}(c)+N_{new}(c) > 1$, the benefit of selecting c as the next input combination may be low. It is better in this case to select the next input combination by using a fault-oriented test generation procedure for a yet-undetected fault f.

To resolve the problems discussed above and to ensure that complete fault coverage can be achieved by the proposed test generation procedure, we take the following approach. Whenever $N_{\det}(c) = 0$ and $N_{new\_activ}(c) \leq 1$ for every primary input combination c, we select the first yet-undetected fault f in the fault list. We attempt to generate a test sequence T' for f starting from the last pair of fault-free/faulty states reached under T. If a test sequence cannot be found, then f is marked and dropped from the fault list. Otherwise, if a test sequence T' is generated for f, we use the first input combination of T' as the next input combination of T. If the measures $N_{\det}(c)$ or $N_{new\_activ}(c)$ increase, selection of primary input combinations continues based on Procedure 1. Otherwise, the same fault f is selected again, and the next input combination of T' is added to T. Thus, the test sequence T' is eventually added to T to detect f, unless better input combinations can be selected. At the end of the process, we consider all the faults that were marked and dropped from the fault list. Such a fault f may be undetectable after a sequence T is applied, however, starting from the reset state, a test for f may exist [13]. Thus, it cannot be concluded that f is redundant unless test generation for it is attempted starting from the reset state. We perform test generation for each such fault separately, starting from the reset state. If a test sequence T' can be generated, it is added to T with an indication that reset has to be applied before T' is applied. The complete procedure is given next. It is given for the case where all primary input combinations can be considered.

Procedure 2: The test generation procedure

(1) Set $i = 0$. Set $T = \phi$. Set F to contain every target fault. Set $R = \phi$.

(2) For every input combination c, compute $N_{\det}(c)$, $N_{new\_activ}(c)$ and $N_{new}(c)$.

(3) Set C to contain all primary input combinations.
Remove from C every input combination c' such that $N_{\det}(c') < \max\{N_{\det}(c): c \in C\}$.
Remove from C every input combination c' such that $N_{new\_activ}(c') < \max\{N_{new\_activ}(c): c \in C\}$.
Remove from C every input combination c' such that $N_{new}(c') < \max\{N_{new}(c): c \in C\}$.

(4) Randomly select an input combination $c_0 \in C$.

(5) If $N_{det}(c_0) = 0$ and $N_{new\_activ}(c_0) \leq 1$:
   (a) Select the first fault f in F.
   (b) Perform test generation for f to generate a test sequence T', such that $T \cdot T'$ is a test sequence for f. If no test sequence can be generated:
      (i) Remove f from F and add f to R.
      (ii) If $F = \phi$, go to Step 8
      (iii) Go to Step 5(a).
   (c) (T' was generated) Set c_0 to be the first primary input combination in T'.

(6) Add c_0 to T (i.e., set $T = T \cdot c_0$) and drop from F every fault which is detected by the new sequence T.

(7) Set $i = i+1$. If $F \neq \phi$, go to Step 2.

(8) For every $f \in R$:
   (a) Perform test generation for f starting from the reset state.
   (b) If a test sequence T' is generated, add T' to T and indicate that reset has to be applied before T'. If no test sequence can be generated, f is redundant.

Any fault-oriented test generation procedure can be used in Steps 5(b) and 8(a) of Procedure 2. In our implementation, we used a test generation procedure based on the concepts of [14], that generates a minimum length test sequence for any given fault. The details of this procedure are given in Section 4.

In most circuits considered, R contained only redundant faults. The existence of redundant faults (or more generally, faults that cannot be detected by extending the test sequence T), may sometimes bias the measures $N_{new\_activ}(c)$ and $N_{new}(c)$, and consequently it may bias the selection of c_0 added to T in Step 6. For example, if a circuit has N_r redundant faults and all of them can be activated, then $N_{new\_activ}(c)$ and $N_{new}(c)$ for some input combinations c can be higher by N_r than their values when redundant faults are eliminated. Since the input combinations should be selected only for detectable faults, the bias due to redundant faults is undesirable. Two features of Procedure 2 are used to reduce and eventually eliminate the effect of faults that cannot be detected by the test sequence T. (1) During Procedure 2, faults that cannot be detected by T are identified in Step 5 and removed from F. The computation of $N_{new\_activ}(c)$ and $N_{new}(c)$ is done so as to eliminate the effect of these faults, as follows. $N_{new\_activ}(c)$ is computed as the number of yet-undetected faults which are still in F, and are brought to a new activation state by the sequence $T \cdot c$. $N_{new}(c)$ is computed as the number of yet-undetected faults which are still in F, and are brought to a new state by the sequence $T \cdot c$. This extension is easily implemented by computing $N_{new\_activ}(c)$ and $N_{new}(c)$ with respect to the fault list F maintained by Procedure 2. Note that detected faults are omitted from F in Step 6 and faults that cannot be detected by T are omitted from F in Step 5. (2) Once Procedure 2 terminates and the redundant faults are identified in Step 8, we repeat Procedure 2, this time including only the detectable faults in the set of target faults F and omitting the redundant faults.

Another point that needs attention is the following. Consider the case where the measure $N_{det}(c)$ may be zero for every input combination c during several time units, say $u_i, u_i+1, \cdots, u_j$. Suppose that it is possible to find input combinations such that $N_{new\_activ}(c_k) > 1$ at time unit u_k for $u_i \leq u_k \leq u_j-1$, however, at time u_j, $N_{new\_activ}(c) \leq 1$ for every c. In this case, the fault-independent procedure would be used for time units $u_i, \cdots, u_{j-1}$ and the fault-oriented procedure would be used at time u_j. However, the fault-independent procedure does not lead to fault detection in this case. Thus, instead of starting the fault-oriented test generation procedure at time u_j, it may be preferable to start it at time u_i, and potentially reduce the test length by $j-i-1$. We incorporated this option into Procedure 2, however, experimental results indicated that it does not affect the test length significantly.

A similar case that needs to be considered is the following. Let the fault-oriented phase be entered with a fault f. Suppose that to detect f, k input combinations are required. Suppose in addition that after applying $i < k$ of these input combinations, there exists an input combination c for which $N_{det}(c) = 0$ and $N_{new\_activ}(c) > 1$. Then c is selected by the fault-independent phase. However, c may not be useful in bringing any fault closer to detection, and eventually, the fault-oriented phase will be entered again with the same fault f. Thus, the input combinations applied by the fault-independent phase unnecessarily lengthen the test sequence. To avoid such a case, we added to Procedure 2 the restriction that after the fault-oriented phase is entered, the only way to return to fault-independent selection of input patterns is if an input combination c is found such that $N_{det}(c) > 0$. To incorporate this change, we define a flag denoted $f\_oriented$. When $f\_oriented = 1$ it forces selection of input patterns by the fault-oriented procedure. Initially, $f\_oriented = 0$. When an input combination is selected in Step 5(c) of Procedure 2, $f\_oriented$ is set to 1. The condition for entering Step 5 is changed to the following:

"If $N_{det}(c_0) = 0$ and either $f\_oriented = 1$ or $N_{new\_activ}(c_0) \leq 1$"

Thus, after the fault-oriented phase is entered once, it is entered again regardless of the value of $N_{new\_activ}(c_0)$, as long as $N_{det}(c_0) = 0$. If Step 5 is not entered (the condition at the beginning of Step 5 is not satisfied), $f\_oriented$ is set to 0. We refer to the modified procedure as Procedure 2'. In the following section we present experimental results of Procedures 2 and 2'.

We point out that Procedures 2 and 2' attempt to generate a single test sequence where reset is applied only once, at the beginning. Multiple resets are used only if there are faults that cannot be otherwise detected (faults detected in Step 8 of the procedure). Alternatively, it is possible to apply reset while generating the test sequence T in Steps 1-7. An appropriate time to apply reset is when a fault f is considered under fault-oriented test generation, and it turns out that a long test sequence is required for its detection. If the test sequence starting from the reset state is shorter, it may be advantageous to apply reset before applying a test for f. We did not pursue this possibility.

To accommodate the case where a reset mechanism is not available, the following changes are made in Procedure 2 (and consequently, Procedure 2'). (1) We replace the initial states of the fault free and faulty circuits with the all-unspecified (all-x) state. (2) The first step of a test generation procedure for circuits without reset is typically to synchronize the fault free and faulty circuits as much as possible. To ensure that Procedures 2 and 2' synchronize the circuits, we added another criterion for the selection of an input combination in the fault-independent phase, as follows. This criterion is similar to the one used in [6]. We denote the number of specified next-state variables in the fault free and in all the faulty circuits corresponding to yet-undetected faults, after a sequence T is applied, by $N_{spec}(T)$. After applying an additional input combination c, the number of specified state variables is $N_{spec}(T \cdot c)$. The benefit from selecting c in terms of

the number of additional state variables specified is $N_{spec}(c) = N_{spec}(T \cdot c) - N_{spec}(T)$. We consider $N_{spec}(c)$ as lower in importance than $N_{det}(c)$, but higher than every other criterion. Thus, if $N_{det}(c_1) = N_{det}(c_2)$ and $N_{spec}(c_1) > N_{spec}(c_2)$, then c_1 is selected. In Procedure 2, the fault-oriented phase is entered for the first time with a given fault f if $N_{det}(c) = 0$, $N_{spec}(c) = 0$ and $N_{new\_activ}(c) \leq 1$. In Procedure 2', the fault-oriented phase is entered for the first time with a given fault f if $N_{det}(c) = 0$, and either $f\_oriented = 1$, or $N_{spec}(c) = 0$ and $N_{new\_activ}(c) \leq 1$. (3) The definition of an activation state is modified as follows. Let $S = s_1 s_2 \cdots s_k$ be a state of the fault free circuit, where s_i is the value of state-variable i. Let $Q = q_1 q_2 \cdots q_k$ be a state of the faulty circuit, where q_i is the value of state-variable i. Then S/Q is an activation state if there exists a state-variable i such that $s_i \neq x$, $q_i \neq x$ and $s_i \neq q_i$. (4) The definition of a new state is modified as follows. Using the notation above, we say that state S/Q *covers* a state A/B if (a) $s_i = a_i$ for every i except possibly when $s_i = x$, and (b) $q_i = b_i$ for every i except possibly when $q_i = x$. We consider S/Q reached at the end of a test sequence T as a new state if it does not cover any state reached after a proper subsequence of T is applied.

4. Experimental results

To implement Procedures 2 and 2', we need a fault-oriented test generation procedure in Steps 5 and 8. The following test generation procedure was used. It is based on [14], however, it uses the gate-level description of the circuit instead of its state-table. The procedure is given for circuits with small numbers of primary inputs. Circuits with large numbers of primary inputs are discussed after the procedure is given. The procedure proposed starts from a given fault-free/faulty state A_i/A_j, where A_i and A_j are vectors of state-variable values. In Procedures 2 and 2', this is the state reached by the fault-free/faulty circuits after the partially determined test sequence T is applied. We denote by $B_i \xrightarrow[z_i]{c} C_i$ a transition from state B_i to state C_i under primary input combination c, producing a primary output value z_i.

Procedure 3: Fault-oriented test generation for a fault f

(1) Set $S_0 = \{A_i/A_j\}$. Set $u = 0$.

(2) Set $S_{u+1} = \phi$.
For every pair of states $B_i/B_j \in S_u$:
For every input combination c:
Let $B_i \xrightarrow[z_i]{c} C_i$ and let $B_j \xrightarrow[z_j]{c} C_j$ (C_i, C_j, z_i and z_j are obtained using gate-level simulation for states B_i and B_j under input c).
If $z_i \neq z_j$, stop: A test sequence can be found by tracing the transitions that led into C_i/C_j.
If C_i/C_j has not been reached before, add C_i/C_j to S_{u+1} (considering every state C_i/C_j at most once ensures that the resulting test sequence has minimum length).

(3) Set $u = u+1$. If $S_u \neq \phi$, go to Step 2.

(4) ($S_u = \phi$ and the fault was not detected in Step 2) Stop: The fault cannot be detected.

Procedure 3 is extended to handle circuits with large numbers of primary inputs by considering a limited number of randomly selected primary input combinations, as described in Section 2.2. We also set a limit on the maximum value of u in Step 3 of the procedure. The resulting test generation procedure is not guaranteed to generate a minimum length test sequence. In addition, the resulting procedure is not complete, i.e., it may fail to generate a test sequence for a detectable fault, since it may not consider all the states reachable from the initial state.

Due to the use of randomly selected primary input combinations, the test generation procedure may not give the same test sequence for a given fault f in two different applications. Consider a case where the fault-oriented procedure is entered with a fault f and a test sequence $T' = (t_1, t_2, \cdots, t_k)$ is generated. Then t_1 is used by Procedure 2. Suppose that the fault-oriented procedure is then entered again. We want to ensure that t_2 is found. However, this may not happen since the randomly generated input combinations may change. Consequently, a longer test sequence may be generated, or no test sequence may be generated for f. To prevent this case, we store the test sequence T' and the fault f for which it was generated. If the fault-oriented procedure is entered again immediately with the same fault f, test generation is not repeated and t_2 is selected.

If reset is not available, Procedure 3 is started from fully unspecified initial states. A state C_i/C_j is considered as new and entered into S_{u+1} in Step 2 if it does not cover a state D_i/D_j that exists in S_v for $v \leq u+1$.

We applied Procedure 2 to MCNC finite-state machine benchmarks and to some of the smaller ISCAS-89 benchmark circuits, assuming that reset to the all-0 initial state is available. We also applied Procedure 2' to some of the circuits. We compared the test sequence lengths to the test sequence lengths obtained in [8] and in [9]. The results are reported in Table 2. All the procedures included in Table 2 detected all irredundant faults in each circuit. In Table 2, after circuit name, we give the number of primary inputs, the number of primary outputs and the number of state variables. The number of faults is given next, followed by the number of redundant faults in parentheses. The test sequence lengths from [8] and from [9] are given next. They are followed by the test length obtained by Procedure 2, referred to as SEQCOM, and Procedure 2' (when it was applied), referred to as SEQCOM'. It can be seen that except for one circuit (*s*344, explained below), the test sequence length produced by SEQCOM and SEQCOM' is lower than the test sequence lengths produced by the other procedures, sometimes significantly. For example, for *s*1488, the reduction in test sequence length is over three times compared to [8] and [9]. CPU times and results for larger circuits are omitted, since program efficiency was not considered while implementing Procedures 2, 2' and 3. However, to give an indication of the CPU times involved, we measured the CPU time for several of the circuits on a SUN SPARC 2 workstation. For *dk*15, the CPU time was 24 seconds; and for *bbara*, the CPU time was 53 seconds. As for *s*344, we speculate that the reason for the high test length obtained by SEQCOM is related to the use of a single reset at the beginning of the test sequence. In [9], reset is applied 9 times along a sequence with a total length of 48 input combinations.

In Table 3, we report results for circuits where the number of inputs is too large to consider all input combinations at every iteration. Instead, we consider a random selection of 100 input combinations in Procedures 2, 2' and 3. Similar reductions to those of Table 2 can be seen in test length. For these circuits, all the procedures reported derived tests for all irredundant faults.

We also applied Procedures 2 and 2' to circuits without reset. Results for four of the circuits considered above are shown in Table 4. Procedure 2' proved to be more effective than Procedure 2, and we therefore report results only for Procedure 2'. The number of faults in parentheses is the number of faults that cannot be detected under the single observation time approach

Table 2: Experimental results for circuits with reset (small number of inputs)

			st	faults		test sequence length			
circuit	inp	out	var	total	(red)	[8]	[9]	SEQCOM	SEQCOM'
bbara	4	2	3	119	(0)	70	-	53	
bou	2	6	5	617	(7)	137	-	129	127
cse	7	7	4	432	(0)	319	-	248	
dk14	3	5	3	241	(0)	61	-	52	
dk15	3	5	2	170	(1)	32	-	30	
dk16	2	3	5	529	(6)	162	-	145	141
dk17	2	3	3	159	(0)	68	-	61	
planet	7	19	6	1077	(13)	527	673	316	
styr	9	10	5	1087	(1)	752	-	365	
s27	4	1	3	32	(0)	13	-	8	
s208	11	2	8	215	(65)	-	192	109	
s298	3	6	14	308	(35)	-	119	118	111
s344	9	11	15	324	(5)	-	48	81	
s382	3	6	21	399	(20)	-	1028	710	
s386	7	7	6	384	(70)	186	168	124	
s400	3	6	21	421	(26)	-	1091	751	
s444	3	6	21	474	(35)	-	1026	833	
s1488	8	19	6	1486	(40)	1301	1031	320	
s1494	8	19	6	1506	(51)	1362	1040	310	

Table 3: Experimental results for circuits with reset (large number of inputs)

			st	faults		test sequence length		
circuit	inp	out	var	total	(red)	[8]	[9]	SEQCOM
s420	19	2	16	430	(226)	-	187	111
s510	19	7	6	564	(0)	-	584	203
s641	35	24	19	467	(59)	-	134	77
s953	16	23	29	1079	(10)	-	578	180

[13] using three value simulation. The results are compared to the test lengths from [10], [11] and [12], which also use the single observation time approach and three value simulation. Except for the procedure of [12], all the procedures included in Table 4 derived tests to detect all the faults which are detectable using the single observation time strategy. For the procedure of [12], which did not find tests for some detectable faults, we give in parentheses the number of faults that were left undetected. These faults are left undetected in addition to the faults left undetected by all other test generation procedures.

Table 4: Experimental results for circuits without reset (small number of inputs)

	faults		test length				
circuit	total	(und)	[10]	[11]	[12]		SEQCOM'
s208	215	(78)	184	294	NA		114
s298	308	(43)	306	203	161		160
s386	384	(70)	311	292	154	(19)	131
s1488	1486	(42)	1294	1270	243	(52)	358

In Table 5 we show the results obtained for circuits with larger numbers of inputs, again, without using reset. For each procedure we report the number of faults detected and the test length. The test sets obtained can be seen to be smaller than those of [10], [11] and [12].

Finally, comparison of the sequences generaed by the proposed procedures to random sequences shows that the circuits considered are not random pattern testable, and that the proposed procedures are effective in generating short test sequences and detecting larger numbers of faults than random sequences.

Table 5: Experimental results for circuits without reset (large number of inputs)

		[10]		[11]		[12]		SEQCOM'	
circuit	flts	det	len	det	len	det	len	det	len
s420	430	179	218	179	172	NA	NA	179	149
s641	467	404	216	403	185	404	139	404	80
s1196	1242	1239	453	1238	376	1232	347	1232	238

5. Concluding remarks

We presented a procedure to generate short test sequences for synchronous sequential circuits. The proposed procedure constructs a test sequence by selecting one input combination at a time. The procedure switches between a fault-independent phase and a fault-oriented phase. During a fault-independent phase, an input combination is selected to maximize the number of detected faults and to minimize the additional sequence length required to detect other faults. During a fault-oriented phase, the first input combination of a test sequence for a given fault is added to the test sequence. Experimental results were presented to demonstrate the effectiveness of this procedure.

References

[1] I. Pomeranz, L.N. Reddy and S.M. Reddy, "COMPACTEST: A Method To Generate Compact Test Sets for Combinational Circuits", IEEE Trans. on CAD, July 1993, pp. 1040-1049.

[2] G. Tromp, "Minimal Test Sets for Combinational Circuits," 1991 Intl. Test Conf., Oct. 1991, pp. 204-209.

[3] M. Abramovici, K. B. Rajan and D. T. Miller, "FREEZE!: A New Approach for Testing Sequential Circuits", in Proc. 29th Design Autom. Conf., June 1992, pp. 22-25.

[4] R. K. Roy, T. M. Niermann, J. H. Patel, J. A. Abraham and R. A. Saleh, "Compaction of ATPG-Generated Test Sequences for Sequential Circuits", in Proc. Intl. Conf. on Computer-Aided Design, Nov. 1988, pp. 382-385.

[5] M. Abramovici, J.J. Kulikowski, P.R. Menon and D.T. Miller, "SMART and FAST: Test Generation for VLSI Scan-Design Circuits," IEEE Design and Test of Comp., August 1986, pp. 43-54.

[6] V. D. Agrawal, K. T. Cheng, and P. Agrawal, "A Directed Search Method for Test Generation Using Concurrent Simulator", IEEE Trans. CAD of ICS, February 1989, pp. 131-138.

[7] T. P. Kelsey and K. K. Saluja, "Fast Test Generation for Sequential Circuits", Intl. Conf. Comp. Aided Design, Nov. 1989, pp. 354-357.

[8] I. Pomeranz and S.M. Reddy, "Test Generation for Synchronous Sequential Circuits Based on Fault Extraction", 1991 Intl. Conf. on Computer-Aided Design, Nov. 1991, pp. 450-453.

[9] H. Cho, G. D. Hachtel and F. Somenzi, "Redundancy Identification/Removal and Test Generation for Sequential Circuits Using Implicit State Enumeration", IEEE Trans. on CAD, July 1993, pp. 935-945.

[10] T. Niermann and J. H. Patel, "HITEC: A Test Generation Package for Sequential Circuits", European Design Autom. Conf., 1991, pp. 214-218.

[11] D.-H. Lee and S. M. Reddy, "A New Test Generation Method for Sequential Circuits", 1991 Intl. Conf. on Computer-Aided Design, Nov. 1991, pp. 446-449.

[12] E. M. Rudnick, J. H. Patel, G. S. Greenstein and T. M. Niermann, "Sequential Circuit Test Generation in a Genetic Algorithm Framework", in Proc. Design Autom. Conf., June 1994, pp. 698-704.

[13] I. Pomeranz and S.M. Reddy, "Classification of Faults in Synchronous Sequential Circuits", IEEE Trans. on Computers, September 1993, pp. 1066-1077.

[14] I. Pomeranz and S.M. Reddy, "On Achieving a Complete Fault Coverage for Sequential Machines Using the Transition Fault Model", 28th Design Autom. Conf., June 1991, pp. 341-346.