On Generating Test Sets that Remain Valid in the Presence of Undetected Faults

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

Abstract

It was shown in [1] that a test set for single stuck-at faults computed by injecting each fault into the fault free circuit and finding a test to distinguish the faulty circuit from the fault free one may be invalidated in the presence of faults that are not detected by the test set. We consider the problem of generating tests for single stuck-at faults that remain valid in the presence of undetected single stuck-at faults. We show that enumeration of all subsets of faults that may occur in the circuit without being detected may be too computation intensive, and is not necessary to obtain high-quality test sets. We present a test generation procedure to generate tests that remain valid in the presence of undetected faults. The procedure targets simultaneously multiple subsets of undetected faults that may be present in the circuit. It thus allows test generation time to be minimized by allowing the number of subsets of faults considered explicitly to be minimized. Based on this test generation procedure, several approximate procedures are also explored.

1. Introduction

Test generation for single stuck-at faults can be viewed as a process of injecting single faults into a fault free circuit to create faulty circuits, and generating tests to distinguish each faulty circuit from the fault free circuit. It was observed in [1] that a test set T generated in this way may be invalidated in the presence of a single stuck-at fault that is not detected by T. In other words, a single stuck-at fault f' detected by T when it is present in the circuit alone may not be detected by T if the circuit-under-test contains, in addition to f', a single stuck-at fault f that is not detected by T. Our goal in this work is to generate test sets for single stuck-at faults that are not invalidated in the presence of undetected single stuck-at faults. To be consistent with our goal of detecting single faults, we assume that faults occur one at a time, and that the circuit is tested at least once between every two occurrences of new faults. This is typical of operation-time testing (as opposed to manufacturing testing). We assume the context of operation-time testing in this work. The following scenario demonstrates the importance of considering the test invalidation problem. Let a test set T be generated by injecting single faults into the fault free circuit, and let f_0 be a fault that remains undetected by T. Now suppose that f_0 occurs in the circuit, invalidating the tests for a fault f_1. Eventually, f_1 may occur in the circuit. The two faults f_0 and f_1 present in the circuit may now invalidate the tests for a third fault, f_2, that would also go undetected, and so on. By ensuring that the test set detects f_1 even if f_0 is present, we can identify that the circuit is faulty as soon as f_1 occurs.

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

We point out that two detectable faults present in the circuit at the same time may also constitute an undetectable double fault. However, we are interested in the single stuck-at fault model and the invalidation of tests for single stuck-at faults. Hence, arbitrary multiple undetectable faults are not considered. An undetectable fault that becomes detectable in the presence of another fault will be detected by the proposed procedure.

A single stuck-at fault f may be left undetected by a test set T because it is undetectable, or because the test generation procedure aborts on f. Redundancy removal and fault removal procedures such as [3-5] alleviate the test invalidation problem partly by eliminating redundant faults and possibly undetectable faults from the circuit. However, these procedures may not result in fully testable circuits and do not eliminate aborted faults. Due to the presence of faults that remain undetected, the problem of test invalidation observed in [1] deserves attention.

In this work, we describe a test generation procedure that generates tests for single stuck-at faults that remain valid even in the presence of the single stuck-at faults that are left undetected. We divide the procedure into two parts. In the first part of the procedure, we generate subsets of faults that may be present without being detected, and may invalidate the tests we generate. These subsets are defined in Section 2. In the second part of the procedure, we generate tests that remain valid in the presence of any subset of faults computed in the first part. In practice, the two parts can be combined. We keep the two parts separate for ease of presentation and comment on the possibility of combining them where appropriate. The second part of the test generation procedure (the construction of a test set) is divided into three steps. In the first two steps, every test we generate remains valid in the presence of as many subsets of faults as possible. We thus minimize the test generation effort, and allow heuristics to be developed that do not require explicit consideration of all subsets of faults. The basic idea is the following. Let F be a set of faults that may be present in the circuit without being detected. Let G be the set of lines corresponding to the faults in F, i.e., for every fault g stuck-at α in F, g is included in G. We assign to the lines in G unspecified values, and generate tests for as many detectable target faults as possible. The tests generated in this way are not invalidated in the presence of the faults in F. In addition, they are not invalidated in the presence of any subset of faults that affects only lines included in G. We use this scheme to generate tests that are not invalidated in the presence of as many subsets of faults as possible. In the third part of the procedure we use specified faulty values for the faults in F only if some target faults remain for which we cannot generate tests when the lines in G are unspecified.

The proposed test generation procedure contains a fault simulation procedure that can be used to check whether a given test set may be invalidated by undetected faults. As may be

expected (and demonstrated in Section 3), test invalidation does not occur often. For the cases where it does not occur, the fault simulation procedure can be used to verify this fact. When test invalidation does occur, the proposed test generation procedure can be used to generate an appropriate test set.

We start by describing a complete procedure that guarantees that every detectable single stuck-at fault would be detected, independent of the undetected single stuck-at faults present in the circuit at the same time. The complexity of this procedure is high due to the number of subsets of faults that may be present without being detected. We therefore consider heuristics to reduce its complexity. The proposed procedures are applied to small synchronous sequential circuits. Consideration of small circuits allows us to explore the tradeoffs between the number of subsets considered, the test generation complexity, and the effectiveness of the resulting test sequence. The heuristic procedures developed in this way are applicable to circuits of any size.

The paper is organized as follows. In Section 2 we include the definitions and notation we use in this work. In Section 3 we describe a complete two-part test generation procedure and present experimental results. In Section 4 we discuss the more complex steps of the procedure described in Section 3 and introduce heuristics to reduce their complexity. Experimental results are included to demonstrate the effectiveness of the heuristics. In Section 5 we present a different procedure where generation of the subsets of faults and generation of tests are combined. Section 6 concludes the paper.

2. Preliminaries

The single stuck-at fault model is considered in this work. By "fault" we refer to a single stuck-at fault, unless otherwise stated. Our goal is to generate test sets that remain valid in the presence of undetected faults. Since several undetected faults may be present in the circuit at the same time, we must consider subsets of faults (or multiple faults). It is possible to require that the test set would remain valid in the presence of any undetected fault, of any multiplicity. However, undetected multiple faults are difficult to compute. In addition, the test set we compute targets single faults, and may not detect some detectable multiple faults. To be consistent with our goal of detecting single faults, we assume that faults occur one at a time, and that the circuit is tested at least once between every two occurrences of new faults (this is consistent with the context of operation-time testing that we assume). Under this assumption, a multiple fault needs to be considered only if the single faults it contains can occur one after the other, leaving the circuit indistinguishable from the fault free circuit at all times. The subsets of faults of interest are thus defined as follows.

Definition 1: The test set of a circuit C has to be valid under a subset of faults F of C if it is possible to order the faults in F as $(f_1, f_2, \cdots, f_{|F|})$, such that faulty circuit C^{F_i} obtained by injecting $F_i = \{f_1, f_2, \cdots, f_i\}$ into C is indistinguishable from C for $i = 1, 2, \cdots, |F|$.

Given a fault free circuit C, a test set T for single stuck-at faults can be generated by injecting each fault f into C to obtain a faulty circuit C_f, and finding a test to distinguish C from C_f. If a subset of faults F may go undetected by T, it is possible to generate a test set that is not invalidated in the presence of the faults in F as follows. A circuit C^F is defined, which is the fault free circuit into which every fault in F is injected. Test generation is then performed by injecting single faults into C^F. When a fault f is injected into C^F, a faulty circuit C_f^F is obtained. A test t must then be found to distinguish C_f^F from the original fault free circuit C. If all the faults in F escape detection, and if the fault f occurs in the circuit in the presence of the faults in F, then t will distinguish the resulting faulty circuit from the fault free circuit C.

To represent a subset of faults F present in the circuit at the same time, we define a *mask* $M = [m_1 m_2 \cdots m_{N_L}]$, as follows. M has an entry m_i for every circuit line g_i. If F contains the fault g_i stuck-at α_i ($\alpha_i \in \{0,1\}$), then $m_i = \alpha_i$. Otherwise, $m_i = U$. The value U indicates that the line is fault free. We exclude the presence of two faults on the same line, since a line can be either stuck-at 0 or stuck-at 1, but not both. In the fault free circuit, we have $m_i = U$ for every line g_i. During test generation and fault simulation, the value of a line g_i is determined as follows. If $m_i \neq U$, then g_i is set to m_i. Otherwise, the conventional logic value implication rules are used to determine the value of g_i. This fault injection method allows us to use the same test generation procedure regardless of whether the fault under consideration is injected into the fault free circuit or into a faulty circuit.

To describe the test generation process, we use S_1/S_2 to indicate that the fault free circuit is in state S_1 and that the faulty circuit is in state S_2. We use z_1/z_2 to indicate that the fault free circuit produces an output z_1 and that the faulty circuit produces an output z_2. A transition from state S to state P under input x, producing an output z, is denoted by $S \xrightarrow[z]{x} P$. A fault g stuck-at α is denoted g/α.

3. A complete procedure

In this section we describe a complete and accurate two-step procedure to generate test sets that are not invalidated in the presence of the faults they leave undetected.

3.1. Background

The following procedures are used in this section. The fault classification procedure from [2] allows us to classify every fault as detectable or undetectable. We assume that every undetectable fault will go undetected, even if a partial test for it exists. The test generation procedure from [6], modified to allow fault injection using the mask M defined above, allows us to generate a test for every detectable fault (the case where aborted faults exist is treated in a similar way, and it is considered below). Given a fault, the procedure of [6] generates a test sequence that distinguishes every initial state of the fault free circuit from every initial state of the faulty circuit. For this purpose, it considers separately every pair of initial states of the fault free circuit and of the faulty circuit. Initially, the test sequence T is empty. When initial state S_1/S_2 is considered, the test sequence T is first simulated starting from S_1/S_2. If T distinguishes S_1 from S_2, the next pair of initial states is considered. Otherwise, the final state P_1/P_2 reached when T is applied starting from S_1/S_2 is recorded. A test sequence T' is generated starting from the final state P_1/P_2, and T' is concatenated to T. The test sequence T' for P_1/P_2 is generated by exploring successors of P_1/P_2. During the test generation process, if a fault $f = g_0/\alpha_0$ is considered in the presence of a subset of faults $F = \{g_1/\alpha_1, g_2/\alpha_2, \cdots, g_k/\alpha_k\}$, then M is set as follows for the faulty circuit. $m_j = \alpha_j$ for $0 \leq j \leq k$ and $m_j = U$ for every other line.

If we omit from the test generation procedure described above the part where an additional test sequence T' is computed and added to T, we obtain a fault simulation procedure. Given a test sequence T, the fault simulation procedure considers every pair of states S_1/S_2, as above. If, for any initial state S_1/S_2, it turns out that T does not distinguish S_1 from S_2 and additional input vectors are needed, fault simulation stops, indicating that the fault is not detected. If T distinguishes the faulty circuit from the fault free circuit for every pair of initial states, then f is detected by T. This observation applies to the test generation procedure described in Section 3.3 as well. Thus, we can obtain a fault simulation procedure simply by omitting the step where an additional test sequence T' is computed, and instead declaring that the fault under consideration is not detected by the given input sequence whenever a sequence T' is required.

3.2. Generating subsets of undetectable faults

In the first step of the test generation procedure, we generate the subsets of faults under which the test set has to be valid (cf. Definition 1). Procedure 1 shown in Figure 1 is used for this purpose. At an arbitrary stage of the procedure, we have subsets of faults $\tilde{F} = \{F_0, F_1, F_2, \cdots, F_k\}$. Initially, we have $\tilde{F} = \{F_0\}$, where $F_0 = \phi$ (corresponding to the fault free circuit). For every subset of faults $F_i \in \tilde{F}$, Procedure 1 considers the circuit C^{F_i} (C into which F_i is injected). It finds the undetectable faults in C^{F_i}, say $\{f_{i1}, f_{i2}, \cdots, f_{im_i}\}$. It then defines a new subset of faults $F_i \cup \{f_{ij}\}$ for every undetectable fault f_{ij}, $1 \le j \le m_i$. This process continues until no additional subsets are created (no additional undetectable faults are found for any of the subsets of faults generated).

Procedure 1: Generating subsets of faults by Definition 1

(1) Set $\tilde{F} = \{\phi\}$.

(2) If all the subsets included in $\tilde{F}$ have been considered, stop: $\tilde{F}$ is the required set of fault subsets.

(3) Select a subset $F \in \tilde{F}$ that has not been considered yet.

(4) For every fault $f = g/\alpha$ such that $g/\beta \notin F$ for any $\beta \in \{0,1\}$:

  (a) Use the procedure of [2] to determine whether $F \cup \{f\}$ is detectable.

  (b) If $F \cup \{f\}$ is undetectable, add it to $\tilde{F}$.

(5) Go to Step 2.

Figure 1: Procedure 1

The number of subsets of faults created by Procedure 1 cannot be determined based on the number of undetectable faults in the original circuit alone. This is because certain detectable faults become undetectable in the presence of undetectable faults, increasing the number of subsets; and certain undetectable faults become detectable in the presence of other undetectable faults, reducing the number of subsets. The numbers of subsets we found in small synchronous sequential circuits are shown in Table 1. The circuits are MCNC finite-state machine benchmarks. In the second column of Table 1 we show the number of collapsed single stuck-at faults. In the third column of Table 1 we show the number of subsets of size one (the undetectable faults in the original circuit). In the fourth column of Table 1 we show the total number of subsets of faults that need to be considered. The total number of subsets includes the empty subset, corresponding to the original circuit. In the last column of Table 1 we show the size of the largest subset of faults generated by Procedure 1.

Table 1: The number of fault subsets

circuit	faults	subsets size 1	subsets all	largest subset
beecount	112	2	4	2
dk512	124	2	4	2
ex4	176	5	32	5
lion9	62	9	1476	11
train11	104	4	100	8

3.3 Test generation

In this section, we describe a test generation procedure that produces test sets that are not invalidated in the presence of the subsets of faults we computed in Section 3.2.

3.3.1 Preliminaries

The set of target faults for test generation is defined as follows. Let $F_{s.s.a}$ be the set of all (collapsed) single stuck-at faults in the original circuit. Let $\tilde{F} = \{F_0, F_1, \cdots, F_K\}$ be the set of all subsets of faults produced by Procedure 1. Our goal is to produce a test set that detects every fault $f_i \in F_{s.s.a}$ in the presence of every subset of faults $F_j \in \tilde{F}$, if f_i is detectable in the presence of F_j. A target fault of the test generation procedure thus has the form $\{f_i\} \cup F_j$, where $f_i \in F_{s.s.a}$ and $F_j \in \tilde{F}$. The fault $\{f_i\} \cup F_j$ is excluded from the list of target faults only if (1) $f_i \in F_j$ (in this case, the fault is undetectable), or (2) $f_i = g/\bar{\alpha}$ and $g/\alpha \in F_i$ (in this case, $\{g/\bar{\alpha}\} \cup F_i$ is undefined).

In the first part of the test generation procedure, we generate tests that are not invalidated in the presence of *any* subset of faults in $\tilde{F}$. In other words, when we generate a test for a fault f_i, the test is valid for every fault $\{f_i\} \cup F_j$, where $F_j \in \tilde{F}$. In the second part of the procedure, we generate tests that are not invalidated in the presence of as many subsets of faults as possible. In the third part of the procedure, we generate tests for the remaining target faults. Every part starts with simulation of the test sequence generated in the previous part, followed by test generation for the remaining faults. Theorem 2 of [2] guarantees that it is always possible to extend a given test sequence to detect a detectable fault. Thus, starting from the test sequence already generated in previous steps does not limit our ability to detect new faults. The three parts of the procedure are described next. We use the following notation. Corresponding to a subset of faults F_i we define a subset of lines $G_i = \{g: g/\alpha \in F_i\}$. We also define $\tilde{G} = \cup \{G_i: F_i \in \tilde{F}\}$.

3.3.2 Part 1 - using unspecified values on $\tilde{G}$

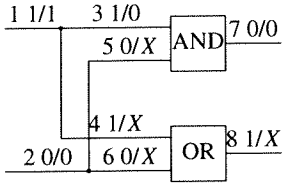
We start with an example to demonstrate how a test that is not invalidated in the presence of any subset of faults in $\tilde{F}$ can be generated for a fault f.

Example: Consider the circuit shown in Figure 2 with the set of faults $F_{s.s.a} = \{3/0, 3/1, 4/0, 4/1, 5/0, 5/1, 6/0, 6/1\}$. Line 7 is a primary output. Line 8 is a next state variable, and line 2 is its present state variable. Procedure 1 produces the undetectable subsets of faults shown in Table 2. We have $\tilde{G} = \{4, 5, 6\}$ (recall that $\tilde{G}$ is the union of all the lines included in any subset in $\tilde{F}$). In the first part of the test generation process, we set $m_4 = m_5 = m_6 = X$. Here, X is an unspecified value that can be either 0 or 1. During test generation, a line with $m_i = X$ is assigned the value X in the faulty circuit. For illustration, we generate a test for the fault 3/0 when lines 4, 5 and 6 are unspecified in the faulty circuit. In this case, $M = [UU0XXXUU]$. Considering initial state 0/0, the results of applying the input value 1

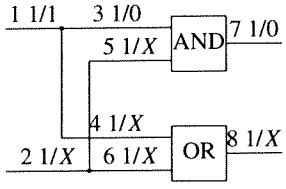
are shown in Figure 2(a). The next state is $1/X$ and the output is 0/0. The results of applying the input value 0 are a next state $0/X$ and an output value 0/0. Applying input values 0 or 1 in state $0/X$ does not take us closer to fault detection. Applying the input value 1 in state $1/X$ results in an output value 1/0 as shown in Figure 2(b), and states 0/0 are distinguished by the input sequence (1,1). Considering states 0/1, 1/1 and 1/0, it turns out that the input sequence (1,1) is sufficient to distinguish these states as well. Since we set all the lines in $\tilde{G}$ to X, we ensure that the test (1,1) for the fault 3/0 is not invalidated in the presence of any of the fault subsets in $\tilde{F}$. □

Table 2: The fault subsets of the circuit of Figure 2

ϕ	{4/0}	{4/0, 5/1}	{4/1, 6/1}	{4/0, 5/1, 6/0}
	{4/1}	{4/0, 6/1}	{5/1, 6/0}	{4/0, 5/1, 6/1}
	{5/1}	{4/1, 5/1}	{5/1, 6/1}	{4/1, 5/1, 6/0}
	{6/1}	{4/1, 6/0}		{4/1, 5/1, 6/1}



(a) Input value 1 in state 0/0



(b) Input value 1 in state $1/X$

Figure 2: Test generation using unspecified values

In general, in Part 1 of the test generation procedure we perform test generation for every fault $f_i = g_i/\alpha_i \in F_{s.s.a}$ such that $g_i \notin \tilde{G}$. We use for this purpose a mask where $m_i = \alpha_i$, $m_j = X$ for every $g_j \in \tilde{G}$, and $m_k = U$ for every other line. A line with $m_i = X$ is assigned an unspecified value X in the faulty circuit, regardless of the value it would have assumed if we had used conventional logic simulation to determine its value. If a test for f_i is found, we mark every target fault $\{f_i\} \cup F_j$, for every $F_j \in \tilde{F}$, as detected.

The numbers of target faults, the numbers of target faults detected in Part 1 and the test length after Part 1 of the test generation procedure are shown in Table 3 under columns "target" and "Part 1" subcolumns "det" and "len", respectively. The circuits are the ones considered in Table 1. It can be seen that large numbers of faults are detected in Part 1. For example, for *lion*9, 59,040 target faults are detected by explicitly considering only $|F_{s.s.a}| = 62$ faults.

Table 3: Test generation results for the circuits of Table 1

		Part 1		Part 2		Part 3	
circuit	target	det	len	det	len	det	len
beecount	440	416	48	428	51	440	51
dk512	488	144	27	236	74	488	74
ex4	5472	1952	62	2275	157	5472	157
lion9	74904	59040	19	62727	24	74904	27
train11	9680	4500	21	5050	52	9680	52

3.3.3 Part 2 - using unspecified values on $\{G_i\}$

A fault f may not be detected in Part 1 because of the large set of unspecified values we fix in the faulty circuit. In Part 2, we use smaller subsets of unspecified values. We consider only target faults that were left undetected after Part 1.

We consider individual subsets F_j. To generate tests which are valid in the presence of as many subsets of faults as possible, we assign to the lines in G_j unspecified values in the faulty circuit. To generate a test for a fault $f_i = g_i/\alpha_i$, we use the following mask for the faulty circuit. $m_i = \alpha_i$, $m_k = X$ for every $g_k \in G_j$, and $m_l = U$ for every other line. The fault f_i is first simulated under the test sequence already generated in Part 1, using this mask. It is possible that the fault is detected, however, this was not observed before because of the unspecified values used in Part 1. If the fault is not detected, an attempt is made to extend the test sequence to detect it. If f_i is detected, then the target faults $\{f_i\} \cup F_k$ are detected for every k such that $G_k \subseteq G_j$. For example, let us consider the subset of faults F_j = {5/1, 6/1} in Table 2 and the fault 3/0. The corresponding mask is $M = [UU0UXXUU]$. If a test is generated for the fault 3/0 under F_j, then the same test detects the fault 3/0 under the subsets {5/1, 6/0}, {5/1, 6/1}, {5/1} and {6/1}.

To maximize the number of target faults detected by each test, we start with the largest sets G_j, and consider a smaller subset G_k only if there exist target faults involving F_k that are not detected yet.

The results of applying Part 1 followed by Part 2 to the circuits of Table 1 are shown in Table 3 under column "Part 2".

3.3.4 Part 3 - using specified values

In Part 3 of the test generation procedure we consider individual subsets F_j and use the faulty values for each subset of faults separately. As before, we consider only target faults that were left undetected after Parts 1 and 2.

To generate tests which are valid in the presence of F_j, we assign to the lines in G_j their faulty values in the faulty circuit. To generate a test for a fault $f_i = g_i/\alpha_i$, we use the following mask. $m_i = \alpha_i$, $m_k = \alpha_k$ for every $g_k/\alpha_k \in G_j$, and $m_l = U$ for every other line. For example, let us consider the subset of faults F_j = {4/1, 5/1, 6/1} in Table 2 and the fault 3/0. The corresponding mask is $M = [UU0111UU]$. The fault f_i is first simulated under the test sequence already generated, using this mask. It is possible that the fault is detected, however, this was not observed before because of the unspecified values used. If the fault is not detected, an attempt is made to extend the test sequence to detect f_i. If f_i is detected, then only the target fault $\{f_i\} \cup F_j$ is detected. The results of applying Part 3 following Parts 1 and 2 are shown in Table 3 under column "Part 3".

3.3.5 Experimental results

The complete test generation procedure is given in Figure 3 as Procedure 2. The results of applying Procedure 2 to the circuits of Table 1 are shown in Table 3. Many of the faults are detected in Part 1, by tests which are valid in the presence of any undetectable subset of faults. In many cases, simulation is sufficient in Parts 2 and 3 to show that a fault is detected by the test sequence already generated in Part 1. Only a small number of faults require test generation in Parts 2 and 3. The fault coverage after Part 3 is always 100%.

Procedure 2: Test generation

(1) Apply Procedure 1 to generate the subsets of undetectable faults.
(2) Apply Part 1 of the test generation procedure.
(3) Apply Part 2 of the test generation procedure starting from the test sequence generated in Part 1.
(4) Apply Part 3 of the test generation procedure starting from the test sequence generated in Part 2.

Figure 3: Procedure 2

3.3.6 Extension

As an intermediate step between Parts 2 and 3 of the procedure, it is possible to consider multiple subsets, as follows. Consider two subsets of faults $F_1 = \{g_1/\alpha_1, g_2/\alpha_2, \cdots, g_k/\alpha_k\}$ and $F_2 = \{g_1/\alpha_1, g_2/\alpha_2, \cdots, g_m/\alpha_m\}$ such that $m < k$. Suppose we can generate a test for a target fault $\{f\} \bigcup F_1$, where $g_{m+1} = \alpha_{m+1}, g_{m+2} = \alpha_{m+2}, \cdots g_k = \alpha_k$ in the fault free circuit at all time units. Then the same test detects the target fault $\{f\} \bigcup F_2$. The reason for this is that by setting $g_{m+1}, \cdots, g_k$ in the fault free circuit to their faulty values, we ensure that the faults in $F_1 - F_2$ are never activated. Under this condition, the test that detects f in the presence of F_1 also detects f in the presence of F_2.

This observation is most effective in detecting a large number of target faults when F_1 is large and F_2 is small. If $F_2 = \phi$, then the test is valid in the presence of any subset of F_1. A procedure to take advantage of this observation should therefore consider subsets F_1 in decreasing order of size and their subsets F_2 in increasing order of size.

4. Heuristics

The main limitation of Procedure 2 is that the number of subsets of faults that need to be considered is, in the worst case, exponential in the number of faults. When the number of undetectable faults is large, the number of subsets of faults that need to be considered may prevent the application of the procedure. We propose in this section several approximations to reduce the number of subsets that need to be considered. We first study them separately, and then use them together.

We first reduce the number of subsets generated by Procedure 1 by imposing an upper bound on the sizes of the subsets it generates. The rationale for this approximation is as follows. As discussed above, our model accommodates multiple faults only if they can occur one after the other without being detected. Consider a sequence of single stuck-at faults occurring in a circuit one after the other. If a fault detected by our test set ever occurs, then according to our assumptions, it will be detected when the test set is applied next. Given that most of the target faults are detectable, the likelihood of a large number of undetectable faults occurring in sequence, without a detectable fault ever occurring in between, goes to zero as the number of faults in the sequence increases. A test set that remains valid in the presence of a small number of undetectable faults may thus be sufficient to overcome the test invalidation problem in practice. To demonstrate the effects of limiting the subset size, we applied Procedure 1 with an upper bound of 2 on the sizes of the fault subsets. We then applied Procedure 2 to the resulting subsets. The results are shown in Table 4. In the second column of the table we show the number of fault subsets of size ≤ 2 generated by Procedure 1. In the third column we show the number of target faults based on the subsets generated. In the following columns we show the number of target faults detected and the test length after each part of Procedure 2 is applied using only subsets of size ≤ 2. In the last column of the table we give the coverage of the test sequences described in the table, when *all* subsets of faults, of all sizes, are considered. In parentheses we repeat the number of target faults detected in Table 3, when all subsets are considered. It can be seen that test invalidation may occur if more than two undetected faults are present. This happens for *lion*9, where 74,868 of 74,904 target faults are detected when test sequences that were generated for bounded subsets of faults are applied to *all* subsets of faults. However, subsets of size three or more are not likely to occur.

Table 4: Test generation with bounded subset sizes ≤ 2

			Part 1		Part 2		Part 3		coverage	
circuit	subs	targ	det	len	det	len	det	len	all subsets	
beecount	4	440	416	48	428	51	440	51	440	(440)
dk512	4	488	144	27	236	74	488	74	488	(488)
ex4	16	2736	976	62	1279	157	2736	157	5472	(5472)
lion9	48	2511	1920	19	2310	24	2511	24	74868	(74904)
train11	15	1476	675	21	990	52	1476	52	9680	(9680)

Observing that most of the faults are detected by the test sequence generated in Part 1 of the test generation procedure, we propose next an approximation that does not use the subsets of faults produced by Procedure 1 at all during test generation. Procedure 3 shown in Figure 4 is used, as explained below.

Procedure 3: An approximate test generation procedure

(1) Find the set of undetectable faults in the original circuit. Let this set be F_{undet}.
(2) Generate a test sequence T by applying Part 1 of Procedure 2, using G_{undet}.
(3) Starting from T, apply Part 3 of Procedure 2, injecting single stuck-at faults only into the fault free circuit.

Figure 4: Procedure 3

In Procedure 3, we replace $\tilde{G}$ by G_{undet}. Part 1 of Procedure 2 is applied only to G_{undet}. If new faults become undetectable as faults from F_{undet} are injected, they are ignored in Procedure 3. The resulting test sequence is then complemented in Step 3 by considering the original circuit and generating tests for the remaining single faults.

To check the effectiveness of Procedure 3, we applied it to the circuits of Table 1. We then simulated the resulting test sequences on the complete sets of target faults that take into account all the sets of undetectable faults produced by Procedure 1 (with unlimited sizes). The results are given in Table 5, as follows. Under column "accurate" of Table 5 we repeat the number of detected target faults and the test length from Table 3. Under column "approximate" we give the number of faults detected using Procedure 3 and the corresponding test length. It can be seen that some loss of fault coverage occurs for *lion*9, where 74,868 of 74,904 target faults are detected, and for *train*11, where 9,670 of 9,680 target faults are detected. This can be overcome by performing test generation for the remaining target faults after fault simulation. Note also that the test length for the other circuits never increased by Procedure 3, and even decreased in the case of *ex*4.

Combining the information derived from the experiments above, we propose the following procedure. Due to the low probability of a long sequence of undetectable faults occurring in the circuit, we limit the subset sizes to two. In addition to the low probability of having subsets of size larger than two, Table 4 shows that the loss of fault coverage compared to consideration

Table 5: Test generation results by Procedure 3

circuit	accurate det	accurate len	approximate det	approximate len
beecount	440	51	440	51
dk512	488	74	488	74
ex4	5472	157	5472	149
lion9	74904	27	74868	23
train11	9680	52	9670	40

of all subsets of faults is low, and does not justify the high complexity of deriving all subsets of faults. We generate a test sequence using Procedure 3 that was shown to achieve high fault coverage without considering a large number of subsets. In this case, Procedure 3 is applied only to subsets of size ≤ 2. We then simulate the test sequence using all target faults based on subsets of size ≤ 2. The results are reported in Table 6 for the circuits considered above and for additional circuits. In column "faults" of Table 6 we show the total number of single stuck-at faults and the number of undetectable faults. The number of target faults is given next. In Phase 1, test generation is done with the lines in G_{undet} being unspecified. In Phase 2, test generation is done only for the fault free circuit into which each single stuck-at fault is injected. In addition, each target fault is simulated in the presence of each subset of undetectable faults of size ≤ 2 using specified faulty values. It can be seen that the fault coverage after Phase 2 is 100% in 7 of the 11 circuits considered. When the fault coverage achieved by Procedure 3 is significantly lower than 100%, it is possible to generate tests for the remaining target faults, using the test generation procedure described in Section 3, which is guaranteed to achieve 100% fault coverage.

Table 6: Procedure 3 with subsets of size ≤ 2

circuit	faults total	faults undet	target	Phase 1 det	Phase 1 len	Phase 2 det	Phase 2 len
bbara	130	8	5300	2592	81	5262	129
bbsse	235	3	1414	1284	120	1410	199
beecount	110	2	440	416	48	440	51
cse	355	2	1772	1304	240	1772	266
dk512	122	2	488	144	27	488	74
ex4	171	5	2736	976	62	2736	149
ex5	138	14	14490	14490	63	14490	63
ex7	149	11	11347	792	13	11326	72
lion9	53	9	2511	1800	19	2511	23
mark1	197	7	7201	2117	70	7201	107
train11	100	4	1476	495	21	1473	40

Finally, in this work, the test generation procedure of [6] is used. To apply Procedure 3 to large circuits, it is possible to use the procedure of [8], or to use any test generation procedure such as [9] that uses three-value logic under the single observation time approach [2].

5. A combined procedure

Until now, we kept the generation of the fault subsets and the generation of test sequences separate. This required that the faults that remain undetected would be determined in a preprocessing step. In practice, undetected faults may be found during test generation. Procedure 4 given in Figure 5 combines subset generation with test sequence generation.

6. Concluding remarks

We considered the problem of generating tests that remain valid in the presence of undetected faults. We presented a procedure to enumerate all the relevant subsets of faults that may occur in a circuit without being detected. We then presented a test generation procedure to generate tests that remain valid in the presence of these subsets of undetected faults. In the first two parts of the procedure, multiple subsets of undetectable faults were considered simultaneously. The faults that remained undetected were then targeted individually. Several approximate procedures were also explored, and resulted in a procedure where only bounded subsets of faults are considered. Direct test generation was initially done only for two subsets of faults, complemented by test generation for the remaining faults.

Procedure 4: Combined fault subset and test generation

(1) Set $\tilde{F} = \{\phi\}$, $\tilde{F}_{targ} = \{\phi\}$, and $T = \phi$. Let $F_{s.s.a}$ contain all collapsed single stuck-at faults.

(2) For every $F \in \tilde{F}_{targ}$:

  (a) For every fault $f = g/\alpha \in F_{s.s.a}$ such that $g/\beta \notin F$ for $\beta \in \{0,1\}$:

    (i) Extend T to detect f in the presence of F.

    (ii) If f cannot be detected, add $F \cup \{f\}$ to $\tilde{F}$ and to $\tilde{F}_{targ}$.

  (b) Remove F from $\tilde{F}_{targ}$.

(3) If $\tilde{F}_{targ} = \phi$ or the sizes of the sets in $\tilde{F}_{targ}$ exceed a predetermined bound, stop: T is the required test sequence.

(4) Let $F_{s.s.a}$ contain all collapsed single stuck-at faults.

(5) Set $\tilde{G}_{targ} = \{g: g/\alpha \in F \text{ for some } \alpha \text{ and some } F \in \tilde{F}_{targ}\}$.

(6) For every fault $f = g/\alpha \in F_{s.s.a}$ such that $g \notin \tilde{G}_{targ}$:

  (a) Extend T to detect f when the lines in $\tilde{G}_{targ}$ are set to X.

  (b) If f is detected, remove it from $F_{s.s.a}$.

(7) Go to Step 2.

Figure 5: Procedure 4

References

[1] A. D. Friedman, "Fault Detection in Redundant Circuits", IEEE Trans. on Computers, Feb. 1966, pp. 66-73.

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

[3] K.-T. Cheng, "On Removing Redundancy in Sequential Circuits", in Proc. of 28th Design Autom. Conf., June 1991, pp. 164-169.

[4] M. A. Iyer and M. Abramovici, "Warning: 100% Fault Coverage May be Misleading!!", in Proc. Intl. Test Conf., 1992, pp. 662-668.

[5] I. Pomeranz and S. M. Reddy, "On Achieving Complete Testability of Synchronous Sequential Circuits with Synchronizing Sequences", 1994 Intl. Test Conf., Oct. 1994, pp. 1007-1016.

[6] I. Pomeranz and S. M. Reddy, "The Multiple Observation Time Test Strategy", IEEE Transactions on Computers, May 1992, pp. 627-637.

[7] V. D. Agrawal and S. T. Chakradhar, "Combinational ATPG Theorems for Identifying Untestable Faults in Sequential Circuits", 1993 European Test Conf., pp. 249-253.

[8] I. Pomeranz and S. M. Reddy, "Application of Homing Sequences to Synchronous Sequential Circuit Testing", IEEE Transactions on Computers, May 1994, pp. 569-580.

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