Simulation Based Test Generation for Scan Designs

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

Abstract

We describe a simulation-based test generation procedure for scan designs. A test sequence generated by this procedure consists of a sequence of one or more primary input vectors embedded between a scan-in operation and a scan-out operation. We consider the set of faults that can be detected by test sequences of this form, compared to the case where scan is applied with every test vector. The proposed procedure constructs test sequences that traverse as many pairs of fault-free/faulty states as possible, and thus avoids the use of branch-and-bound test generation techniques. Additional techniques are incorporated into this basic procedure to enhance its effectiveness.

1. Introduction

Simulation-based test generation procedures were shown to be effective in achieving high fault coverages for both combinational and synchronous sequential circuits [1]-[12]. Simulation-based procedures have the advantage that they can be modified relatively easily to consider various levels of circuit description and various fault models. This can be done by replacing the logic and/or fault simulation procedures embedded in them with the appropriate procedures that handle the circuit description or fault model of interest. In addition, these procedures avoid the need for complex test generation procedures based on branch and bound. The procedures of [1]-[12] can be classified into procedures based on randomized and directed searches [1], [9], procedures based on genetic optimization [2]-[8], and procedures based on circuit properties [10]-[12]. The procedure described in this paper belongs to the class of property-based procedures. It is different from previous procedures in this class in that it targets scanned circuits with full or partial scan.

In the proposed procedure, a test sequence consists of a scan-in vector followed by one or more primary input vectors. The sequence ends with another scan operation to observe values of state variables (and possibly scan in the scan-in vector of the next test sequence). Between the two scan operations of a test sequence, flip-flop values are determined through the combinational logic of the circuit, without using any additional scan operations. Thus, even for full-scan circuits, the procedure may apply several consecutive primary input vectors without using the scan chain. This contributes to at-speed testing of the circuit.

After a scan-in operation, the values of the scanned flip-flops are known. Values of non-scanned flip-flops are assumed to be unknown. This allows us to accommodate the situation where the non-scanned flip-flop values cannot be maintained during scan. It also allows us to apply the generated test sequences in any order.

+ Research supported in part by NSF Grant No. MIP-9725053, and in part by SRC Grant No. 98-TJ-645.

We use several techniques used in earlier simulation-based test generation procedures, and adopt them to handle scan-based circuits and test sequences. The techniques we use include the following. (1) Traversing as many different circuit states as possible. This was used in the logic simulation-based procedure *LOCSTEP* [10]. (2) Generating a test sequence based on the fault free circuit and a faulty circuit in the presence of one yet-undetected fault. This was used in *ACTIV – LOCSTEP* [11]. (3) Repeating a primary input vector several times. This technique was introduced in [13], and used in the simulation-based test generation procedure *PROPTEST* [12] for non-scan circuits. (4) Static test compaction. This was used in *PROPTEST* [12] to improve the fault coverage of existing sequences.

The proposed procedure is unique in the way it combines these techniques, and in the fact that it targets full or partial scan circuits. In addition, we consider the following issues.

Instead of the test application process we assume in this work, it is possible to assume that the output of a scanned flip-flop becomes a primary input of the circuit, and the input of a scanned flip-flop becomes a primary output of the circuit. This implies that scan is applied with every primary input vector to control and observe the states of the scanned flip-flops, while holding the values of the non-scanned flip-flops so that they do not change. For partial scan circuits, we show in this work that a fault, which is detectable if scan is used with every primary input vector, may not be detectable if scan is used only at the beginning and at the end of the test sequence. This observation is important since it establishes that a different upper bound on fault coverage may exist depending on the test generation and test application methods used. Nevertheless, we use the test application scheme where a single scan-in and a single scan-out operation are used with every test sequence, since it requires significantly lower test application times, and it removes the need to hold non-scanned flip-flop states. Many commercial test generators for partial scan circuits also make a similar assumption on test application.

In some cases, test application may be simplified if the primary input sequences contained in the different test sequences are of a small number of different lengths. We take this into account indirectly, by applying a static compaction procedure to reduce the total number of test sequences. It is also possible to achieve this goal by modifying the proposed procedure.

The paper is organized as follows. In Section 2 we discuss the dependence of the detectable fault set on the test sequence format. In Section 3 we describe the basic procedure for generating test sequences that traverse new states. In Section 4 we add to the basic procedure of Section 3 a method to extract test sequences for additional faults out of the existing test sequences. We also add a compaction procedure that reduces the test length, and at the same time potentially increases the number of detected faults. Experimental results are given in Section 5. Section 6 concludes the paper.

2. Detectable faults

A test sequence we consider consists of a scan-in operation that specifies the values of the scanned flip-flops, followed by primary input vectors applied without using scan. The values of the scanned flip-flops are then scanned-out. A different test application scheme consists of the use of scan with every primary input vector.

For full scan circuits, every combinationally detectable fault can be detected with either one of the test application schemes above. This can be achieved using a test sequence that consists of a single primary input vector embedded between a scan-in and a scan-out operation. Such a test sequence is applicable under both test application schemes. For partial scan circuits, it may be necessary to use test sequences consisting of more than one primary input vector in order to detect some faults. As a result, some faults that can be detected when scan is applied with every primary input vector may not be detectable if scan is used only at the beginning and the end of the test sequence. The following example demonstrates such a case. We use three-value simulation for this example.

Table 1: An example fault

	Y_1Y_2, z	
y_1y_2	$I = 0$	$I = 1$
00	11,0	10,0
01	00,0	01,0
10	10,0	11,0
11	01,0	00,1/0

Consider the circuit whose state table is shown in Table 1. The fault under consideration changes the primary output value in state 11 under input 1 from 1 to 0 (shown as 1/0 in Table 1). Suppose that y_2 is scanned. Using scan with every primary input vector, it is possible to detect the fault as follows.

1. Set $y_1 = x$. Scan in the value $y_2 = 0$, and apply the primary input value 0. The next state is $y_1 = 1$.
2. Scan in $y_2 = 1$ (bringing the circuit to state 11), and apply the primary input value 1. The primary output will be set to 1 in the fault free circuit, and to 0 in the faulty circuit. The fault is thus detected.

Next, we consider the same fault, this time assuming that scan will be used only at the beginning and the end of the test sequence. The following assignments are possible.

1. It is possible to scan in $y_2 = 0$, or $y_2 = 1$. With $y_1 = x$ and $y_2 = 0$, the next value of y_1 is 1 and the next value of y_2 is unknown. This is independent of the input value. With $y_1 = x$ and $y_2 = 1$, the next value of y_1 is 0 and the next value of y_2 is unknown, again, independent of the input value.
2. From state $1x$ or $0x$, the next state is xx independent of the input value (the test application scheme prevents us from setting the value of y_2 using scan). From state xx, the circuit cannot be brought to state 11 where the fault can be detected.

The example above shows that the fault coverage expected for partial scan circuits using test sequences that perform scan only at the beginning and the end of a test sequence is lower than the fault coverage expected for the same circuits when scan can be used at any time unit. Note that to detect the fault using scan at every time unit requires holding the state of the non-scanned flip-flops during scan. In many partial scan designs this may not be possible.

3. Test sequences traversing new states

In this section, we describe the generation of test sequences that traverse new states. The proposed test generation procedure performs several iterations. In each iteration, test sequences are generated using the fault free circuit and the faulty circuit in the presence of a single target fault f. The goal of test generation is to take the fault free and faulty circuits through as many different states as possible, without repeating any state in a single iteration. The proposed procedure is different from *LOCSTEP* [10] and other procedures proposed earlier in several ways.

(1) The proposed procedure never repeats a state in the same iteration. An iteration terminates when the procedure cannot reach a new state. In *LOCSTEP*, states may be repeated, and there are several heuristics that control the way in which states are repeated. This difference is related to the fact that *LOCSTEP* was developed for non-scan circuits, whereas here, we are interested in circuits with scan. When scan is available, it is possible to scan-in a new state even if one cannot be reached by applying primary input vectors. In addition, it is possible to observe fault effects through scan. Consequently, it is not necessary to traverse previously-visited states in order to achieve these goals. The scan option is not available to *LOCSTEP*, and consequently, it must continue applying input vectors in order to reach new states and propagate fault effects even if these input vectors do not result in new states during one or more time units.

(2) *LOCSTEP* considered only the fault free circuit. Here, we use a single target fault per iteration, and consider the combined fault-free/faulty state. This speeds up the detection of target faults, at the cost of doubling the simulation effort in generating a test sequence. *ACTIV – LOCSTEP* also considered the fault free circuit and one faulty circuit. However, it imposed stricter requirements on the test sequences it generated by requiring that the target fault would be activated, and that the fault free circuit would go into a cycle. Here, we only require that the circuits would go into new states with every additional vector.

(3) It was observed in [13] and later in [12] that input sequences where input vectors appear repeatedly are effective in achieving high fault coverages for non-scan circuits. We also allow input vectors to be repeated. However, in [12] and [13], the numbers of repetitions were either predetermined, or determined randomly. Here, we allow an input vector to be repeated only as long as it allows us to reach new states.

We use the following notation to describe the proposed procedure. The i-th test sequence generated in a particular iteration is denoted by T_i. The primary input vector at time unit u of T_i is denoted by $T_i(u)$. The length of T_i is denoted by L_i. The sequence of states the circuit traverses under T_i is denoted by S_i. The state at time unit u of S_i is denoted by $S_i(u)$. The initial state $S_i(0)$ always leaves the non-scanned state variables unspecified. The values of the scanned state-variables are assigned by scanning them in. Each state $S_i(u)$ consists of a fault free state $S_i^{ff}(u)$, and a faulty state $S_i^{fy}(u)$. We have $S_i(u) = S_i^{ff}(u)/S_i^{fy}(u)$.

During the generation of test sequences $T_1, T_2, \cdots, T_m$ in a particular iteration based on a single target fault f, we ensure that $S_i(u)$ does not cover any state $S_j(v)$ such that $j < i$ or $v < u$. An example of two test sequences that satisfy this condition is shown in Table 2. The sequences are generated using the same target fault f. In this example, the circuit has four inputs and three state variables. The first two state variables are scanned. This third one is not scanned, and it therefore assumes unspecified values at time unit 0 of every sequence. Considering the

states by increasing time unit and increasing sequence index, it can be seen that none of the states covers a state that appears before it. State $S_2(2) = 000/000$ is covered by state $S_1(0) = 00x/00x$, however, we allow this relationship since $S_2(2)$ has more specified values than $S_1(0)$ that was reached earlier.

Table 2: An example of test sequences with new states

u	$T_1(u)$	$S_1(u)$	u	$T_2(u)$	$S_2(u)$
0	1111	$00x/00x$	0	1110	$11x/11x$
1	0100	$100/xx0$	1	0011	100/100
2	1100	$001/0x0$	2	0011	000/000
3		101/100	3		010/010

Procedure 1 below describes the generation of test sequences in an arbitrary iteration with a target fault f. In Procedure 1, Ψ is the set of states traversed by the generated test sequences, and T is the set of test sequences generated so far. Initially, $\Psi = \phi$ and T is the set of sequences generated in previous iterations. For the first iteration, $T = \phi$ and $i = 1$.

An initial state for a new test sequence is selected in Step 2. The state is selected by randomly generating N_{RAND} states, and checking whether one of them is new with respect to Ψ. If no new state can be found, the procedure terminates.

A new input vector is added to T_i in Step 4. The input vector is selected out of N_{RAND} input vectors. The first input vector considered is the same as the previous vector in T_i, if it exists. Otherwise, the vector is randomly set. A vector is selected if it brings the circuit into a new state. Construction of T_i stops if none of the vectors brings the circuit into a new state, or when the length of T_i reaches a preselected bound, L.

If construction of T_i stops before its length reaches L, we add one more vector to T_i. This allows us to take advantage of the last state of T_i, in case this state contributes to the detection of any fault.

The procedure terminates when the number of sequences in T reaches a bound N_{SEQ}, or in Step 2 when no new initial state can be selected. Thus, at most N_{SEQ} sequences are generated in an iteration. This number may be lower if T is not empty at the beginning of Procedure 1, or if no new initial states can be selected before the number of sequences in T reaches N_{SEQ}.

Procedure 1: Generation of test sequences for a target fault f

(1) Set $\Psi = \phi$. Let $T = \{T_1, T_2, \cdots, T_m\}$ be the set of test sequences generated so far. Set $i = m + 1$ (the index of the next sequence to be included in T).

(2) For $r = 1, 2, \cdots, N_{RAND}$:
Select a state S where the non-scanned state variables are unspecified, and the remaining state variables are set randomly. If S/S does not cover any state in Ψ, set $S_i(0) = S/S$, add S/S to Ψ, and go to Step 3.
Return T and stop.

(3) Set $u = 0$.

(4) For $r = 1, 2, \cdots, N_{RAND}$:
  (a) Select an input vector t as follows. If $r = 1$ and $u > 0$, set $t = T_i(u - 1)$. Else, set t to be a random vector.
  (b) Simulate the fault free and faulty circuits in the presence of f under t starting from state $S_i(u)$. Let the next state be $Q = Q^{ff}/Q^{fy}$.
  (c) If Q does not cover any state in Ψ, set $T_i(u) = t$, $S_i(u + 1) = Q$, add Q to Ψ, and go to Step 5.
  Go to Step 6.

(5) Set $u = u + 1$. If $u < L$, go to Step 4.

(6) If the length of T_i is smaller than L:
  (a) Select an input vector t as follows. If $r = 1$ and $u > 0$, set $t = T_i(u - 1)$. Else, set t to be a random vector.
  (b) Set $T_i(u) = t$.

(7) Set $i = i + 1$. If $i < N_{SEQ}$, go to Step 2.

Procedure 1 considers a single target fault f, and it may result in a set of sequences T that contains N_{SEQ} sequences. Consequently, additional calls to Procedure 1 may not produce new sequences. In Procedure 2 given below, we call Procedure 1 repeatedly with a new target fault every time. Fault simulation of the new sequences added to T in Step 5 of Procedure 2 identifies sequences in T that do not detect any yet-undetected faults. Such sequences are removed from T in Step 5 of Procedure 2. Note that Ψ is initialized every time Procedure 1 is called. Thus, test sequences generated in different iterations may traverse the same states.

One of the parameters used by Procedure 1 is the maximum length of a test sequence, L. There are several competing effects that need to be considered in determining the value of L. (1) Shorter test sequences take advantage of scan more often in order to set initial states and observe fault effects. (2) Longer test sequences use scan less often and thus have reduced test application time. In addition, some faults may require longer sequences for propagation to primary outputs or scanned state variables. As a compromise, we start with longer test sequences, and then reduce the test length in steps. Initially, L is set equal to a preselected constant L_s. It is then reduced until it reaches a final value L_f. The procedure reduces the value of L after N_{SAME} iterations with no improvement in fault coverage. The procedure terminates after N_{SAME} iterations with $L = L_f$ that do not improve the fault coverage, or when the list of target faults F becomes empty.

Procedure 2: Test generation based on new states

(1) Set $T = \phi$. Let F be the set of target faults. Set $F_{targ} = \phi$. Set $n_{same} = 0$. Set $L = L_s$.

(2) If $F_{targ} = \phi$, set $F_{targ} = F$.

(3) Select a fault $f \in F_{targ}$. Remove f from F_{targ}.

(4) Call Procedure 1 to generate test sequences based on f, with an upper bound of L on test sequence length.

(5) Fault simulate the new test sequences added to T in Step 4. Remove from T every sequence that does not detect any new faults. Drop the detected faults from F and from F_{targ}. If any faults are dropped, set $n_{same} = 0$. Else, set $n_{same} = n_{same} + 1$.

(6) If F is empty, stop.

(7) If $n_{same} < N_{SAME}$, go to Step 2.

(8) If $L \neq L_f$, reduce L to its next value, and go to Step 2.

4. Adding subsequences and compaction

In this section, we extend the procedure of the previous section by extracting subsequences that detect target faults, and by performing compaction.

To demonstrate how subsequences of the sequences included in T can be useful in detecting target faults, consider the test sequence T_3 shown in Table 3. The circuit is the same one considered in Table 2, where the first two state variables are scanned. The faulty states shown belong to a fault f which is not

detected by T_1, T_2 or T_3. It can be seen that although T_3 does not detect f, it is possible to detect f by scanning out the state at time unit 2 of T_3. Thus, the sequence T_4 shown in Table 3, which consists of the scan-in vector of T_3 and its first two primary input vectors, detects f.

Table 3: An example of subsequence extraction

u	$T_3(u)$	$S_3(u)$	u	$T_4(u)$	$S_4(u)$
0	1111	$00x/00x$	0	1111	$00x/00x$
1	0000	000/000	1	0000	000/000
2	1100	001/010	2		001/010
3	0011	111/110			
4		111/111			

We identify subsequences such as T_4 by capturing for every sequence $T_i \in T$ the time units where fault effects reach scanned state variables. If the effects of a yet-undetected fault f reach a scanned state variable at time unit u under a sequence $T_i \in T$, then the subsequence consisting of the scan-in vector of T_i and its first $u-1$ primary input vectors detects f. We add this subsequence to T.

We search for subsequences to detect yet-undetected faults after every call to Procedure 1. This is shown in Step 3 of Procedure 3 below. Procedure 3 is similar to Procedure 2, except that the extensions described in this section have been added to it. At odd iterations of Procedure 3, we call Procedure 1. At even iterations, we try to extract subsequences of the sequences generated in the previous iteration.

Procedure 3: Test generation

(1) Set $T = \phi$. Let F be the set of target faults. Set $F_{targ} = \phi$. Set $n_{same} = 0$. Set $n_{iter} = 1$. Set $L = L_s$.

(2) If $F_{targ} = \phi$, set $F_{targ} = F$.

(3) If n_{iter} is odd:

  (a) Select a fault $f \in F_{targ}$. Remove f from F_{targ}.

  (b) Call Procedure 1 to generate test sequences based on f, with an upper bound of L on test sequence length.

  Else:

  For every sequence added to T in the previous iteration, check if there exist subsequences that detect at least one fault out of F. Add all such subsequences to T.

(4) Fault simulate the new test sequences added to T in Step 3. Remove from T every sequence that does not detect any new faults. Drop the detected faults from F and from F_{targ}. If any faults are dropped, set $n_{same} = 0$. Else, set $n_{same} = n_{same} + 1$.

(5) Reduce the lengths of the sequences added to T in the current iteration as much as possible without reducing the fault coverage. If any new faults are detected, drop them from F and from F_{targ}.

(6) If F is empty, stop.

(7) Set $n_{iter} = n_{iter} + 1$. If $n_{same} < N_{SAME}$, go to Step 2.

(8) If $L \neq L_f$, reduce L to its next value, and go to Step 2.

Step 5 of Procedure 3 is a compaction step where we attempt to reduce the lengths of the test sequences added to T in the current iteration. Length reduction for test sequences of scan circuits was considered in [14]. Let F_i be the set of faults detected by a test sequence T_i. Let $F_{i,PO}$ be the set of faults detected by T_i on primary outputs, and let $F_{i,SO}$ be the set of faults detected during the scan-out operation at the end of T_i. For every fault $f \in F_{i,PO}$, let $u_{det}(f)$ be the time unit where f is detected by T_i. Let the length of T_i be L_i. This length is reduced to $\hat{L}_i$ that satisfies the following conditions.

(1) $\hat{L}_i > u_{det}(f)$ for every $f \in F_{i,PO}$. This ensures that every fault detected by T_i on the primary outputs will continue to be detected on the same primary outputs after the length of T_i is reduced.

(2) Every $f \in F_{i,SO}$ is propagated to a scanned flip-flop at time unit $\hat{L}_i - 1$. This ensures that the faults in $F_{i,SO}$ will continue to be detected by T_i after its length is reduced.

In the worst case, $\hat{L}_i = L$ will satisfy these conditions. The length of T_i is reduced to $\hat{L}_i$ in Step 5 of Procedure 3.

Following Procedure 3, we also perform reverse order simulation of the test sequences in T, and drop sequences that do not detect any faults.

5. Experimental results

In this section, we report the results of the procedure described in the previous sections. We consider ISCAS-89 and ITC-99 benchmark circuits. We use the following parameters.

We generate up to $N_{SEQ} = 500$ test sequences in every iteration. The initial test length $L = L_s$ is equal to the number of state variables in the circuit, or to 100 if the number of state variables is larger than 100. The value of L is reduced to 10, then to 5, and finally it is reduced by one until it reaches 1. A test length of 1 is useful for full scan circuits. We allow at most $N_{SAME} = 100$ iterations with no improvement in fault coverage before switching to the next value of L, or terminating the procedure. For example, for a circuit with 120 state variables, we consider $L = 100$, 10, 5, 4, 3, 2 and finally 1, each value for at most $N_{SAME} = 100$ iterations with no improvement. For a circuit with 8 state variables, we consider $L = 8$, 5, 4, 3, 2 and 1. We select a state or input vector out of $N_{RAND} = 100$ candidates.

We measure the fault simulation time throughout all the iterations of Procedure 3, and denote this time by R_{sim}. We also measure the time to generate test sequences throughout all the iterations of Procedure 3, and denote this time by R_{gen}. We report the normalized time R_{gen}/R_{sim}.

The results for full scan circuits are shown in Table 4, and the results for circuits with 90% scan are shown in Table 5. To obtain 90% scan, we arbitrarily select 90% of the circuit flip-flops, and assume that they are scanned. The tables are organized as follows. After the circuit name, we show the total number of faults, the number of detectable faults, and the number of faults detected by the proposed procedure. The number of detectable faults excludes faults that are known to be combinationally redundant, but includes all other faults. Thus, this number is an upper bound on the number of detectable faults for partial scan circuits. Next, we show the number of iterations until the procedure terminates, and the number of iterations until the fault coverage reaches its final value. We then show the number of test sequences in T at the end of Procedure 3, and the total length of all the test sequences in T (the length of a test sequence is the number of primary input vectors it contains). In the last column, we show the normalized run time R_{gen}/R_{sim}. The following points can be seen from Tables 4 and 5.

In most of the circuits with full scan and some of the circuits with 90% scan, we obtain 100% fault coverage. For such circuits, the number of iterations is relatively small. Larger numbers of iterations are typically required for circuits that have

Table 4: Full scan circuits

circuit	flts	det. able	detect	iterations all	to f.c	seq	length	n.time
s208	215	215	215	69	69	29	50	45.99
s298	308	308	308	1	1	16	124	4.05
s344	342	342	342	1	1	11	92	4.41
s382	399	399	399	5	5	26	194	0.70
s420	430	430	425	1075	975	58	75	8.31
s444	474	460	460	705	5	26	121	0.52
s526	555	554	554	711	11	43	378	3.31
s641	467	467	467	69	69	34	442	11.70
s820	850	850	850	555	555	91	221	3.40
s1196	1242	1242	1242	27	27	90	886	1.61
s1423	1515	1501	1501	713	13	49	770	0.27
s1488	1486	1486	1486	137	137	78	218	1.76
s5378	4603	4563	4563	1383	1283	242	1960	0.01
b01	135	135	135	3	3	11	32	2.22
b02	72	72	72	5	5	8	19	5.00
b03	452	452	452	9	9	22	122	1.38
b04	1396	1396	1394	711	111	32	889	0.82
b06	206	206	206	1	1	13	28	25.84
b09	436	436	436	619	619	30	143	2.15
b10	522	522	522	6	6	33	176	1.80
b11	1175	1175	1166	723	23	35	515	0.57

Table 5: Circuits with 90% scan

circuit	flts	det. able	detect	iterations all	to f.c	seq	length	n.time
s208	215	215	175	649	49	18	33	2.37
s298	308	308	291	775	75	19	135	4.59
s344	342	342	342	5	5	11	120	8.07
s382	399	399	391	703	3	23	173	4.21
s420	430	430	366	875	175	46	73	1.41
s444	474	460	457	711	11	26	96	1.84
s526	555	554	458	1075	675	51	153	0.32
s641	467	467	467	65	65	32	229	10.44
s820	850	850	651	1209	809	61	217	0.27
s1196	1242	1242	1242	17	17	90	884	1.27
s1423	1515	1501	1450	977	477	53	1026	0.07
s1488	1486	1486	822	961	361	41	100	0.13
s5378	4603	4563	4507	1173	1073	256	2062	0.01
b01	135	135	135	7	7	12	36	1.65
b02	72	72	72	41	41	9	22	1.88
b03	452	452	445	771	171	25	164	1.93
b04	1396	1396	1376	711	111	35	857	0.24
b06	206	206	194	635	35	9	59	7.88
b09	436	436	426	729	629	31	198	1.06
b10	522	522	500	859	259	25	218	2.98
b11	1175	1175	1136	753	553	38	397	0.29

combinationally redundant faults, for partial scan circuits with undetectable faults, or when the proposed procedure otherwise fails to detect some faults. In these cases, the fault coverage reaches its final value after a smaller number of iterations, but additional iterations are required to satisfy the termination condition. For example, $s526$ has one combinationally redundant fault that causes the proposed procedure to go through 711 iterations before it terminates in the case of full scan, even though the final fault coverage is achieved after only 11 iterations. The extra iterations can be saved by identifying undetectable faults in advance, or by setting different termination conditions.

We do not have information about undetectable faults in the partial scan circuits considered. Therefore, we do not know whether the faults that remain undetected by the proposed procedure are undetectable. However, we observe that the fault coverage is high in all the cases. To provide additional information, we experimented with all possible selections of 90% partial scan in $s208$, $s298$ and $s1488$. We only considered one test length, $L = L_s$, in Procedure 3. For $s208$ and $s1488$, 90% partial scan implies that one flip-flop remains unscanned. For $s298$, 90% partial scan implies that two flip-flops remain unscanned. In Table 6, we report on the scan configuration that resulted in the highest fault coverage. Table 6 is organized similar to Tables 4 and 5, except that after the circuit name, we show the indices of the unscanned flip-flops. Table 6 shows that the partial scan selection can have a significant effect on fault coverage.

Table 6: Considering all scan configurations

circuit	unscan	det. flts	able	detect	iterations all	to f.c	seq	length	n.time
s208	4	215	215	192	181	81	27	40	3.59
s298	7,11	308	308	307	101	1	18	150	36.02
s1488	1	1486	1486	1235	425	325	58	227	0.26

Next, we present the results of applying the compaction procedure of [14] to some of the circuits reported in Tables 4 and 5. The procedure of [14] reduces the test application time by *combining* test sequences. To combine test sequences T_1 and T_2, the procedure of [14] removes the scan-out operation at the end of T_1, removes the scan-in operation at the beginning of T_2, and concatenates the primary input vectors of T_1 and T_2. This is done without reducing the fault coverage. The removal of scan operations reduces the test application time of the test set. In some cases, combining test sequences increases the fault coverage. The fact that compaction can increase the fault coverage was observed in previous works on compaction as well.

In applying the procedure of [14], our goal is also to reduce the number of different test sequence lengths in the test set. The number of different lengths affects the complexity of test application. With fewer lengths that the tester needs to support, the test application process is simplified. To improve the ability of the procedure from [14] to reduce the number of lengths, we apply the procedure in a way that ensures the following properties. (1) The minimum sequence length is as high as possible, and the maximum sequence length is as low as possible. This ensures that the range of possible lengths is as low as possible, resulting in a small number of different possible lengths. (2) For every sequence length that exists in the final test set, we obtain as many sequences of the same length as possible. This also ensures that we obtain as small a number of different lengths as possible. We achieve these properties by combining sequences in the following way. We define a variable L_{targ} that assumes values out of $\{2, 3, 4 \cdots\}$. For every value of L_{targ}, we apply the procedure of [14] only to pairs of test sequences whose total length is equal to L_{targ}. Denoting the length of the test sequence T_i by L_i, we combine T_1 and T_2 only if $L_1 + L_2 = L_{targ}$. In this way, we control the resulting test sequence lengths.

The value of L_{targ} is determined as follows. Suppose that in the current test set, we have test sequences of lengths $\{L_1, L_2, \cdots, L_N\}$. Suppose that $L_1 < L_2 < \cdots < L_N$. In the first phase of the compaction procedure, we consider $L_{targ} = L_2, \cdots, L_N$ (in this order). For every value of L_{targ}, we combine as many test sequences as possible whose total length is L_{targ}. By starting with low values of L_{targ} that already exist in the test set, we ensure that as many short sequences as possible will be combined, increasing the minimum sequence length in the test set without adding new lengths. To obtain additional compaction, we perform a second phase of compaction where we

allow the procedure to add new test lengths by considering $L_{targ} = 2, 3, \cdots, 2L_N$. Again, most of the compaction occurs for short sequences. When we reach the high sequence lengths, the test set is already compacted, and it is less likely to be possible to combine the longer sequences. This keeps the maximum sequence length low. Finally, in the third phase of the procedure, we again consider $L_{targ} = L_2, \cdots, L_N$, where $\{L_1, L_2, \cdots, L_N\}$ are the lengths in the current test set. This ensures that any additional compaction would be done without increasing the number of different lengths.

We report the results of compaction in Table 7. For space considerations, we only report on several of the ISCAS-89 benchmark circuits with 100% scan. Under column *before compact*, we repeat the results from Table 4 regarding the number of detected faults, the number of sequences, and the total sequence length. We also show the number of different test sequence lengths under subcolumn *dif.l*. Under column *compact phase 1*, we show the same information after the first phase of the compaction procedure. Under column *compact phase 3*, we show the same information after all three phases of the compaction procedure. We mark by asterisks cases where the number of different test lengths is lower than the number before compaction. The following points should be noted.

Table 7: Results after compaction

	det.	before compact				compact phase 1				compact phase 3			
circuit	able	det	seq	len	dif.l	det	seq	len	dif.l	det	seq	len	dif.l
s298	308	308	16	124	7	308	14	124	*6	308	10	124	7
s382	399	399	26	194	9	399	20	194	*8	399	19	194	*8
s444	460	460	26	121	8	460	21	121	*7	460	19	121	9
s526	554	554	43	378	18	554	35	378	*16	554	29	378	19
s641	467	467	34	332	18	467	21	329	*8	467	7	329	*7
s820	850	850	91	221	5	850	59	221	*4	850	45	221	9
s1196	1242	1242	90	886	17	1242	58	881	*10	1242	15	881	*12
s1423	1501	1501	49	770	21	1501	32	770	*15	1501	28	770	*17

(1) Compaction reduces the number of test sequences in all cases. In some cases, it also reduces the total number of vectors. Both of these effects reduce the test application time by reducing the number of scan operations and/or the number of input vectors that need to be applied. (2) Compaction can reduce the number of different sequence lengths, especially in the first phase of the compaction procedure.

6. Concluding remarks

We described a simulation-based test generation procedure for full and partial scan designs. A test sequence generated by this procedure consisted of primary input vectors embedded between a scan-in and a scan-out operation. In an odd iteration (starting from the first iteration), the proposed procedure considers one target fault, and constructs test sequences that traverse as many pairs of fault-free/faulty states as possible. Repetition of input vectors is used whenever possible to enhance the effectiveness of the procedure. In an odd iteration, subsequences that detect yet-undetected faults are extracted from the existing test sequences. This is possible if a fault is activated but not detected during an existing test sequence. By defining a new test sequence that terminates when the fault is activated, it is possible to detect the fault by a scan-out operation. An iteration of the procedure is followed by fault simulation to drop detected faults, and compaction to reduce the sequence lengths. Experimental results were presented to demonstrate the effectiveness of the procedure for full and partial scan circuits.

We also considered the set of faults that can be detected by test sequences where primary input vectors are embedded between two scan operations, compared to the set of detectable faults when scan is used with every primary input vector. We showed that in partial scan circuits, there are faults detectable by the latter method that cannot be detected by the former method. Nevertheless, the former method is preferable in terms of test application time, and since it removes the need to hold non-scanned flip-flop states as required by the latter method.

References

[1] J. Snethen, "Simulation-Oriented Fault Test Generator", in Proc. 14th Design Autom. Conf., June 1977, pp. 88-93.

[2] D. G. Saab, Y. G. Saab and J. A. Abraham, "CRIS: A Test Cultivation Program for Sequential VLSI Circuits," Intl. Conf. Computer-Aided Design, Nov. 1992, pp. 216-219.

[3] 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.

[4] P. Prinetto, M. Rebaudengo and M. Sonza Reorda, "An Automatic Test Pattern Generator for Large Sequential Circuits based on Genetic Algorithms", in Proc. 1994 Intl. Test Conf., Oct. 1994, pp. 240-249.

[5] F. Corno, P. Prinetto, M. Rebaudengo, M. Sonza Reorda and R. Mosca, "Advanced Techniques for GA-based Sequential ATPGs", in Proc. 1996 Europ. Design & Test Conf., March 1996, pp. 375-379.

[6] D. G. Saab, Y. G. Saab and J. A. Abraham, "Iterative [Simulation-Based Genetics + Deterministic Techniques] = Complete ATPG", in Proc. 1994 Intl. Conf. on Computer-Aided Design, Nov. 1993, pp. 40-43.

[7] E. M. Rudnick and J. H. Patel, "Combining Deterministic and Genetic Approaches for Sequential Circuit Test Generation", in Proc. 32rd Design Autom. Conf., June 1995, pp. 183-188.

[8] I. Pomeranz and S. M. Reddy, "On Improving Genetic Optimization based Test Generation", in Proc. 1997 European Design & Test Conf., March 1997, pp. 506-511.

[9] V. D. Agrawal, K. T. Cheng, and P. Agrawal, "A Directed Search Method for Test Generation Using Concurrent Simulator," IEEE Trans. on Computer-Aided Design, Feb. 1989, pp. 131-138.

[10] I. Pomeranz and S. M. Reddy, "LOCSTEP: A Logic Simulation Based Test Generation Procedure", in Proc. 25th Fault-Tolerant Computing Symp., June 1995, pp. 110-119.

[11] I. Pomeranz and S. M. Reddy, "ACTIV-LOCSTEP: A Test Generation Procedure Based on Logic Simulation and Fault Activation", in Proc. 27th Fault-Tolerant Comp. Symp., June 1997, pp. 144-151.

[12] R. Guo, S. M. Reddy and I. Pomeranz, "PROPTEST: A Property Based Test Pattern Generator for Sequential Circuits Using Test Compaction", in Proc. 36th Design Autom. Conf., June 1999, pp. 653-659.

[13] L. Nachman, K. K. Saluja, S. Upadhyaya and R. Reuse, "Random Pattern Testing for Sequential Circuits Revisited", in Proc. 26th Fault-Tolerant Computing Symp., June 1996, pp. 44-52.

[14] I. Pomeranz and S. M. Reddy, "Static Test Compaction for Scan-Based Designs to Reduce Test Application Time", in Proc. 7th Asian Test Symp., Dec. 1998, pp. 198-203.