An Approach to Test Compaction for Scan Circuits that Enhances At-Speed Testing+

Irith Pomeranz
School of Electrical & Computer Eng.
Purdue University
W. Lafayette, IN 47907
pomeranz@purdue.edu

and

Sudhakar M. Reddy
Electrical & Computer Eng. Dept.
University of Iowa
Iowa City, IA 52242
reddy@eng.uiowa.edu

Abstract

We propose a new approach to the generation of compact test sets for scan circuits. Compaction refers here to a reduction in the test application time. The proposed procedure generates an initial test set that is likely to have a low test application time. It then applies an existing static compaction procedure to this initial test set to further compact it. As a by-product, the proposed procedure also results in long primary input sequences, which are applied at-speed. This contributes to the detection of delay defects. We demonstrate through experimental results the advantages of this approach over earlier ones as a method for generating test sets with minimal test application time and long primary input sequences.

1. Introduction

Test compaction procedures for scan designs that attempt to minimize the test application time were described in [1]-[4]. In the procedures of [1]-[4], each test starts with a scan-in operation, followed by one or more primary input vectors applied using the functional clock of the circuit. A test ends with a scan-out operation. The procedures described in [1]-[3] are dynamic compaction procedures. They minimize the test application time by finding an appropriate balance between the number of primary input vectors applied consecutively using the functional clock (i.e., without using the scan chain), and the number of scan operations. For a circuit with N_{SV} scanned state variables, and assuming that the scan clock and the functional clock have the same cycle time, a scan-in/out operation can be replaced by N_{SV} primary input vectors applied using the functional clock, without increasing the test application time. Whenever it is possible to propagate fault effects and set the circuit state using fewer than N_{SV} primary input vectors, scan operations to achieve these goals can be avoided, and thus, the test application time can be reduced.

The procedure of [4] is a static compaction procedure, where test compaction is done as a postprocessing step following test generation. The procedure of [4] is based on the operation of *combining* tests. A test is represented as $\tau_i = (SI_i, T_i, SO_i)$, where SI_i is a scan-in vector to be scanned in at the beginning of the test, T_i is a sequence of primary input vectors to be applied using the functional clock after SI_i is scanned-in, and SO_i is the expected fault-free scan-out vector after T_i is applied. Combining two tests τ_i and τ_j consists of removing the scan vectors SO_i and SI_j, and concatenating T_i and T_j to obtain the primary input sequence $T_i T_j$. The resulting combined test is $\tau_{i,j} = (SI_i, T_i T_j, SO_{i,j})$, where $SO_{i,j}$ is the fault-free scan-out vector expected after SI_i is scanned-in and the sequences of primary input vectors T_i and T_j are applied. The procedure of [4] attempts to combine as many test pairs as possible in order to remove scan operations, thus reducing the test application time. The combination of two tests is accepted only if it does not reduce the fault coverage. The procedure stops when no additional test pairs can be combined.

Experimental results presented in [4] for full-scan circuits show that the test application times obtained by the static compaction procedure of [4] are lower than the test application times obtained by the dynamic compaction procedures of [1]-[3]. This result was obtained by applying the procedure of [4] to a specific initial test set based on a combinational test set.

We observe that the initial test set given to the static compaction procedure of [4] influences the test application time of the final compacted test set to a large extent. As discussed later in Section 2, it can be argued that the initial test set used in [4] has the largest test application time of all the minimal test sets that may be considered. This is because the initial test set consists of a large number of tests each containing a single primary input vector, whereas a compact test set would contain a smaller number of tests with longer primary input sequences. Thus, with the initial test set used in [4], the static compaction procedure has to make significant changes to the test set in order to compact it, and may fail to find a close-to-minimum solution. Consequently, the initial test set in [4] may not be suitable if the goal is to obtain a final test set that has the minimum test application time.

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

DAC 2001, June 18-22, 2001, Las Vegas, Nevada, USA.

In this work, we propose a different approach to the generation of compact test sets. Under this approach, we generate a test set that has the characteristics required to yield a low test application time. These characteristics are also discussed in Section 2. The proposed procedure satisfies these characteristics by generating a single test that has a relatively long primary input sequence and detects a large percentage of the target faults. If necessary in order to achieve complete fault coverage, it also uses a small number of additional tests with primary input sequences of length one. Experimental results show that in several cases, including the larger circuits considered, the test application time of the test set generated by the proposed procedure is smaller than that of the final test set obtained after compaction in [4]. The procedure of [4] can further compact this test set to achieve even lower test application times. Thus, the proposed test set is more suitable than the initial test set of [4] if the goal is to obtain a minimal test application time.

We observe that the primary input sequence T_i of every test τ_i is applied at-speed. As a result of the use of a test containing a long primary input sequence, the proposed approach detects most of the faults by primary input vectors which are applied at-speed. At-speed testing is important for detecting delay defects [5], [6].

We point out that an improvement to the basic procedure of [4] was described in [7]. This procedure improves the levels of compaction that can be achieved, but does not affect the initial test set. Thus, it is orthogonal to our goal of improving the initial test set, and we use the procedure of [4] for all our experiments.

We consider full-scan circuits in this work. The proposed procedure can be extended to the case of partial-scan circuits.

The paper is organized as follows. In Section 2, we provide the motivation for the proposed approach. In Section 3, we describe a procedure for generating a test set with the characteristics pointed out in Section 2. Experimental results are given in Section 4. Section 5 concludes the paper.

2. Motivation

Considering full-scan circuits, the procedure of [4] was applied to an initial test set where every test τ_i consists of a scan-in vector SI_i and a primary input sequence $T_i = (t_i)$ of length one. This test set is equivalent to a combinational test set (in a combinational test set, each test specifies the values of the primary inputs and the present state variables). The test τ_i is equivalent to a combinational test where t_i is applied to the primary inputs and SI_i is applied to the present state variables. The initial test set used in [4] is obtained from a compact combinational test set that achieves complete fault coverage for the circuit. In the test set obtained after static compaction using the procedure of [4], the number of tests is reduced, and some of the primary input sequences have length larger than one. Since a compact combinational test set is used in [4], the initial test set of [4] can be viewed as a minimal test set compared to all possible test sets where the sequences T_i are of length one.

Next, we follow the procedure of [4] when applied to an initial test set that contains N tests each with a primary input sequence of length one. After the procedure of [4] combines one pair of tests, the number of tests is $N-1$. The number of primary input vectors contained in all the tests is still N. Therefore, the average length of a primary input sequence is $N/(N-1)$. After combining i pairs of tests, the number of tests is $N-i$, and the average length of a primary input sequence is $N/(N-i)$. As the number of tests is reduced and the average length of a primary input sequence increases, the test application time goes down. This can be seen from the following formula. Consider a test set with k tests where the length of the jth primary input sequence, T_j, is $L(T_j)$. The number of clock cycles required for test application is $N_{cyc} = (k+1)N_{SV} + \sum_{j=1}^{k} L(T_j)$, where N_{SV} is the number of state variables. The first component of the sum is due to $k+1$ scan operations required to apply k tests. The second component is due to the application of primary input vectors. For our discussion, $\sum_{j=1}^{k} L(T_j) = N$, and $N_{cyc} = (k+1)N_{SV} + N$. This number goes down as additional test pairs are combined and k is reduced.

The lowest test application time will be obtained by the procedure of [4] if $N-1$ test pairs are combined. The test set will then contain a single test and the length of its primary input sequence will be N. The number of clock cycles for test application will be $2N_{SV}+N$. However, the procedure of [4] typically cannot reduce the number of tests to one, since certain test pairs cannot be combined without reducing the fault coverage. Thus, a different starting point for the procedure of [4] may be needed in order to minimize the final test application time. More generally, a different approach to compaction that yields fewer tests and longer primary input sequences may result in lower test application times.

In this work, we describe a procedure for generating test sets that contain small numbers of tests, with relatively high average primary input sequence lengths. The small number of tests implies a small number of scan operations, and a small number of clock cycles for scan. This allows us to apply a relatively high number of primary input vectors to achieve a high fault coverage while keeping the test application time low. In the following section, we describe the proposed procedure.

3. The compaction procedure

The proposed procedure generates a single test τ_{seq} that contains a long primary input sequence (significantly longer than one), and detects all or most of the target faults. In addition, it may add to the test set a small number of tests with input sequences of length one if this is necessary in order to achieve complete fault coverage. The test τ_{seq} is generated based on a test sequence T_0 that was generated for the circuit assuming that scan is not available. Such test sequences tend to be long, and they detect large percentages of the circuit faults. They can be generated efficiently by existing test generation procedures. We address the efficiency issue further in Section 4.

For the discussion in this section, we omit SO_i from the description of a test τ_i in order to simplify the notation, and represent τ_i as $\tau_i = (SI_i, T_i)$, where SI_i is a scan-in vector, and T_i

is a primary input sequence.

The proposed procedure has four phases. The first phase creates from T_0 a scan-based test by selecting a scan-in vector SI and a time unit where scan-out will be performed. The resulting test is $\tau_{SO} = (SI, T_{SO})$, where T_{SO} is a prefix of T_0.

The second phase reduces the length of T_{SO} as much as possible in order to reduce the test application time without reducing the fault coverage. Test length reduction is achieved by omitting vectors from T_{SO} [8]. The resulting test is $\tau_C = (SI, T_C)$. Phases 1 and 2 are repeated several times to select the most appropriate SI and the shortest T_C. The final test obtained is denoted by τ_{seq}.

The third phase adds tests for target faults that remain undetected by τ_{seq}.

The fourth phase consists of static compaction of the resulting test set using the procedure from [4].

In the following subsections, we describe each one of the phases in more detail.

3.1 Phase 1: scan-based test

The steps of the first phase, that creates a scan-based test from the test sequence T_0, are the following.

Step 1: Fault simulation to determine the set of faults F_0 detected by T_0 without using scan.

Step 2: Selection of a scan-in state SI to be applied before T_0 is applied to the circuit. The scan-in state is selected so as to maximize the number of faults detected by $\tau_{SI} = (SI, T_0)$. The circuit state is scanned out after the application of T_0. The faults in F_0 are detected for any selection of SI and need not be simulated in this step. We denote by F_{SI} the set of faults detected by τ_{SI}. We have $F_{SI} \supseteq F_0$.

Step 3: Selection of a time unit u_{SO} along T_0 where application of input vectors will end, and the circuit state will be scanned out. This is equivalent to replacing T_0 by the sequence T_{SO} that consists of the first $u_{SO}+1$ time units of T_0 (we start labeling time units from 0). The result of this step is a test $\tau_{SO} = (SI, T_{SO})$. The set of faults detected by τ_{SO} is denoted by F_{SO}. The selection of u_{SO} guarantees that $F_{SO} \supseteq F_{SI}$.

Next, we describe Step 2 in more detail. As a source of candidate scan-in states, we use a combinational test set C (i.e., a test set generated for the combinational logic of the circuit). Each test $c_j \in C$ is divided into a state vector c_{j_s} and a primary input vector c_{j_i}. Every vector c_{j_s} is used as a candidate scan-in vector. When c_{j_s} is considered, the test $\tau_j = (c_{j_s}, T_0)$ is simulated. The set of faults simulated is $F - F_0$, where F is the set of all the target faults, and F_0 is the set of faults detected by T_0. The subset of detected faults is denoted by F_j. Once all the vectors have been considered, the vector c_{j_0} that results in the largest set F_{j_0} is selected and assigned to SI. The set of faults $F_0 \cup F_{j_0}$ detected by $\tau_{SI} = (SI, T_0)$ is denoted by F_{SI}.

To describe the selection of u_{SO} in Step 3, we use the following notation. For a sequence A, we denote by $A[u_1, u_2]$ the subsequence of A that starts at time unit u_1 and ends at time unit u_2. We denote the length of A by $L(A)$. We define a test $\tau_{SO,i} = (SI, T_0[0,i])$ that consists of the scan-in vector SI and the prefix of T_0 that ends at time unit i. To select the time unit where the state will be scanned out, we consider the tests $\tau_{SO,i} = (SI, T_0[0,i])$ for $i = 0, 1, \cdots, L(T_0)-1$. For every i, we fault simulate $\tau_{SO,i}$ until one of the following conditions is satisfied. (1) A fault $f \in F_{SI}$ is found, which is not detected by $\tau_{SO,i}$. Since our goal is to maximize the number of detected faults, we do not allow any fault detected by τ_{SI} to be undetected by changing the scan-out state. The simulation effort is minimized by stopping the simulation process as soon as such a fault is identified. (2) All the faults have been simulated, and all the faults in F_{SI} are detected. In this case, we consider $\tau_{SO,i}$ as a candidate to replace τ_{SI}, and we store the detected faults in a set $F_{SO,i}$. Since $T_0[0, L(T_0)-1] = T_0$, it is guaranteed that at least one candidate test $\tau_{SO,i}$ will be found, for $i = L(T_0)-1$. Of all the candidates found, we select τ_{SO,i_0} that has the smallest value of i_0. The resulting test is denoted by τ_{SO}, and the set of detected faults is denoted by F_{SO}.

Note that we select τ_{SO,i_0} that has the smallest value of i_0 in order to reduce the primary input sequence length as much as possible. Alternatively, it is possible to select the test τ_{i_1} with the largest set F_{SO,i_1} and the smallest value of i_1. This would maximize the number of detected faults as well as reduce the sequence length. However, our experiments indicated that using i_1 instead of i_0 results in input sequences that are significantly longer, while the increase in the number of detected faults is marginal compared to the use of i_0. Overall, we obtained better results by using i_0 and not i_1.

3.2 Phase 2: vector omission

This phase starts from a test $\tau_{SO} = (SI, T_{SO})$ that detects the set of faults F_{SO}. The goal of this phase is to omit as many vectors as possible from T_{SO} without losing the detection of any of the faults in F_{SO}. This will reduce the test application time. Vector omission is done similar to [8], and therefore, we do not provide a detailed description here. The resulting test is denoted by $\tau_C = (SI, T_C)$.

3.3 Iterative application of Phases 1 and 2

Starting from T_0, Phases 1 and 2 first define a scan-based test $\tau_{SI} = (SI, T_0)$. The scan-out state is then adjusted to obtain a test $\tau_{SO} = (SI, T_{SO})$. Finally, T_{SO} is compacted to obtain $\tau_C = (SI, T_C)$. It is now possible to repeat the process using T_C instead of T_0 as the initial test sequence.

Let us consider an arbitrary iteration i of this process. Let the previous iteration, $i-1$, result in $\tau_{C_{i-1}} = (SI_{i-1}, T_{C_{i-1}})$. Iteration i starts from the sequence $T_{0_i} = T_{C_{i-1}}$. The sequence T_{0_i} is simulated without using scan in order to identify faults detected independent of the scan-in state. A scan-in state SI_i for T_{0_i} is then selected out of the states defined by the combinational test set C used earlier. The scan-out state is adjusted to obtain $\tau_{SO_i} = (SI_i, T_{SO_i})$, and vectors are omitted from T_{SO_i} to obtain a new test $\tau_{C_i} = (SI_i, T_{C_i})$. Thus, at every iteration, the selection of

a new scan-in state and a new scan-out time unit may increase the number of faults detected. The new scan-out time unit and the application of vector omission may reduce the input sequence length and the test application time. Vector omission may also increase the number of detected faults in some cases [8].

The iteration over Phases 1 and 2 terminates when the selected scan-in state has already been selected in an earlier iteration. This is implemented as follows. The tests in C are initially marked *unselected*. If a scan-in state SI_i is selected based on $c \in C$, c is marked *selected*. When considering the tests in C, preference is given to the selection of a test marked *unselected*. Thus, if two tests c_1 and c_2 yield the same fault coverage but one of them is marked *selected*, the one marked *unselected* is used. Only if a higher fault coverage is achieved by a *selected* test, such a test is used. At that point, the iteration over Phases 1 and 2 terminates. Thus, if C contains K test vectors, at most K iterations of Phases 1 and 2 will be performed.

3.4 Phase 3: complete fault coverage

Phases 1 and 2 terminate with a test $\tau_{seq} = (SI_{seq}, T_{seq})$ that detects a set of faults F_{seq}. The test set at this point consists of a single test, τ_{seq}. If any fault f out of the set of target faults F is left undetected (i.e., $f \in F - F_{seq}$), we add a test to detect it. The tests to be added are selected based on the combinational test set $C = \{c_1, c_2, \cdots, c_K\}$, as described next.

For every $c_j \in C$, we define a scan-based test $\tau_j = (SI_j, T_j)$ where $SI_j = c_j[s]$ is the state vector implied by c_j, and $T_j = (c_j[i])$ is a sequence of length one that contains the primary input vector implied by c_j. Since C is a complete test set that detects all the target faults, and τ_j detects the same faults detected by c_j, the tests $\{\tau_j : 1 \le j \le K\}$ can be used to obtain complete fault coverage.

We simulate every test τ_j under the faults in $F - F_{seq}$, and find the set of faults $F_j \subseteq F - F_{seq}$ detected by τ_j. In addition, for every fault $f \in F - F_{seq}$, we find the number of tests $n(f)$ out of $\{\tau_1, \tau_2, \cdots, \tau_K\}$ that detect it, and the index $last(f)$ of the last test that detects it.

We select tests as follows. Let $F_{undet} = F - F_{seq}$. We find the fault $f \in F_{undet}$ that has the minimum value of $n(f)$. We select the test $\tau_{last(f)}$ that detects it, add it to the test set, and drop from F_{undet} all the faults detected by $\tau_{last(f)}$. We repeat this process until F_{undet} is empty. Note that if $n(f) = 1$, it is necessary to include $\tau_{last(f)}$ in the test set. Such tests are selected first by the procedure we use.

The resulting test set consists of τ_{seq}, and tests $\tau_1, \tau_2, \cdots, \tau_M \in \{\tau_1, \tau_2, \cdots, \tau_K\}$ selected to detect the remaining target faults.

3.5 Phase 4: static compaction

The test set $\{\tau_{seq}, \tau_1, \tau_2, \cdots, \tau_M\}$ consists of $M+1$ tests, M of them containing primary input sequences of length one. It may be possible to compact the test set by applying the static compaction procedure of [4]. The static compaction procedure of [4] is especially likely to be able to combine the tests $\tau_1, \tau_2, \cdots, \tau_M$ into tests containing longer input sequences, thus reducing the number of scan operations and the test application time.

4. Experimental results

We applied the proposed procedure to ISCAS-89 benchmark circuits and to ITC-99 benchmark circuits. For ISCAS-89 benchmark circuits, we used the combinational test sets from [9], and the test sequences generated by STRATEGATE [10] and compacted by the static compaction procedure of [11]. For ITC-99 benchmark circuits, the combinational test sets were selected out of a large set of random patterns, and the test sequences were the ones generated by PROPTEST [12].

PROPTEST has relatively low test generation times, and it can be applied efficiently to large circuits to obtain the test sequence T_0 required by the proposed compaction procedure. For cases where the test generation time in [10] or [12] is too high, we also consider later in this section the use of random input sequences for the initial test sequence T_0.

The results using test sequences from [10] and [12] are shown in Tables 1, 2, 3 and 4. In Table 1, after the circuit name, we show the number of circuit flip-flops and the size of the combinational test set C. We then show the total number of single stuck-at faults. Under column *detected*, we show the number of faults detected by the test sequence T_0, the number of faults detected by the test $\tau_{seq} = (SI_{seq}, T_{seq})$ obtained at the beginning of Phase 3, and the number of faults detected by the final test set obtained at the end of Phase 3. All the detectable target faults are detected by the final test set.

Table 1: Detected faults ([10]-[12])

		comb		detected		
circuit	ff	tsts	flts	T0	scan	final
s298	14	24	308	265	279	308
s344	15	15	342	329	339	342
s382	21	25	399	364	379	399
s400	21	24	421	380	395	415
s526	21	50	555	454	480	554
s641	19	22	467	404	412	467
s820	5	94	850	814	818	850
s1423	74	26	1515	1414	1480	1501
s1488	6	101	1486	1444	1452	1486
s5378	179	100	4603	3639	3817	4563
s35932	1728	94	39094	35100	35110	35110
b01	5	24	135	133	135	135
b02	4	15	70	68	69	70
b03	30	43	452	334	341	452
b04	66	97	1346	1168	1203	1344
b06	9	22	202	186	198	202
b09	28	44	420	339	350	420
b10	17	82	512	467	476	512
b11	30	107	1089	997	1003	1078

In Table 2, under column *seq length* we show the length of T_0, and the length of the primary input sequence T_{seq}. In the last column of Table 2, we show the number of tests added in Phase 3 to achieve complete fault coverage.

In Table 3, we compare the numbers of clock cycles required for the application of several test sets. For a given test set $\{\tau_1, \tau_2, \cdots, \tau_k\}$, the number of clock cycles is computed as $(k+1)N_{SV} + \sum_{i=1}^{k} L(T_i)$, where N_{SV} is the number of state variables

Table 2: Test lengths ([10]-[12])

circuit	seq length T0	seq length scan	added c.tst
s298	117	68	10
s344	57	36	2
s382	516	445	8
s400	611	561	7
s526	1006	694	24
s641	101	81	12
s820	491	339	8
s1423	1024	917	11
s1488	455	447	8
s5378	646	585	100
s35932	150	105	0
b01	66	51	0
b02	45	22	1
b03	136	92	16
b04	168	129	13
b06	37	26	2
b09	279	196	13
b10	190	103	18
b11	676	629	20

and $L(T_i)$ is the length of the sequence T_i included in τ_i. Under column [2,3] of Table 3, we show the number of clock cycles obtained by the dynamic compaction procedure of [2] and [3]. The results in [2] and [3] are better than the results obtained in [1]. Under column [4], we show the number of clock cycles required for applying the initial test set used in [4], and the number of clock cycles obtained after static compaction of this test set using the procedure of [4]. The initial test set compacted in [4] is based on the same combinational test set C used for our experiments. Under column *proposed* subcolumn [10]-[12] we show the number of clock cycles required for application of the test set generated by the procedure proposed here before it is further compacted by the procedure of [4] (i.e., at the end of Phase 3). We then show the number of clock cycles required for application of the test set generated by the procedure proposed here after it is compacted by the procedure of [4] (i.e., at the end of Phase 4). The last two column of Table 3 will be explained later.

In Table 4, we compare the lengths of the primary input sequences, which are applied at-speed, under the following test sets. (1) The compacted test sets obtained in [4]. (2) The compacted test sets obtained by the procedure proposed here. For every test set, we show the average length of a primary input sequence, and the range of primary input sequence lengths. The last two columns of Table 4 will be explained later.

From Tables 1 and 2 it can be seen that τ_{seq} detects a large percentage of the circuit faults. In most cases, a small number of tests needs to be added to achieve complete fault coverage. Table 3 shows that the initial test application time of the test sets proposed here is in many cases lower than the initial test application time in [4], and sometimes it is even lower than the test application time obtained in [4] after compaction. In many cases, the final test application time obtained by the proposed procedure is reduced by applying the compaction procedure of [4]. Table 4 shows that the procedure proposed here results in significantly longer primary input sequences than the sequences obtained in [4]. These sequences are applied at-speed, contributing to the detection of delay defects.

Table 3: Numbers of clock cycles

		[4]		proposed [10]-[12]		proposed rand	
circuit	[2,3]	init	comp	init	comp	init	comp
s298	376	374	318	246	218	322	280
s344	166	255	195	98	98	107	107
s382	680	571	529	663	663	646	583
s400	-	549	465	757	715	596	512
s526	1551	1121	995	1264	1222	1150	1024
s641	-	459	326	359	302	403	327
s820	617	569	309	397	392	512	322
s1423	3222	2024	2024	1890	1816	3098	3024
s1488	641	713	335	515	509	605	377
s5378	36937	18179	18179	18943	18585	19178	19178
s35932	-	164254	98572	3561	3561	-	-
b01	-	149	54	61	61	53	53
b02	-	79	41	35	35	33	33
b03	-	1363	724	648	588	511	481
b04	-	6565	2115	1132	1066	2010	1878
b06	-	229	101	64	64	69	69
b09	-	1304	680	629	573	740	656
b10	-	1493	514	461	427	613	494
b11	-	3347	1315	1309	1159	1573	1273
total*	-	39343	29219	29471	28493	32219	30671

* totals are computed without s35932

Table 4: At-speed test lengths

	[4]		proposed [10]-[12]		proposed rand	
circuit	ave	rang	ave	range	ave	rang
s298	1.20	1-2	8.67	1-68	8.17	1-84
s344	1.36	1-2	12.67	1-36	15.67	1-45
s382	1.09	1-2	50.33	1-445	3.43	1-55
s400	1.20	1-2	94.67	1-563	4.84	1-70
s526	1.14	1-3	31.22	1-694	1.80	1-30
s641	1.47	1-3	9.30	1-81	9.00	1-86
s820	2.24	1-11	43.38	1-340	3.81	1-66
s1423	1.00	1-1	84.36	1-918	39.46	1-1001
s1488	2.66	1-8	56.88	1-448	4.60	1-95
s5378	1.00	1-1	6.92	1-586	10.99	1-1000
s35932	1.36	1-5	105.00	105-105	-	-
b01	4.80	3-7	51.00	51-51	43.00	43-43
b02	2.17	1-4	11.50	1-22	25.00	25-25
b03	1.55	1-4	7.20	1-92	7.58	1-79
b04	2.30	1-15	10.92	1-129	3.69	1-69
b06	2.50	1-6	9.33	1-26	11.00	1-31
b09	1.64	1-3	17.42	1-196	1.90	1-17
b10	2.88	1-10	7.12	1-103	3.74	1-57
b11	2.12	1-5	40.56	1-630	3.59	1-87

Results using random input sequences instead of the test sequences generated in [10] or [12] are shown in the last two columns of Tables 3 and 4, and in Table 5. The number of clock cycles at the end of Phase 3 of the proposed procedure (before the test set is compacted using the procedure of [4]) and at the end of Phase 4 (after the test set is compacted using the procedure of [4]) are shown in the last two columns of Table 3. The lengths of the primary input sequences in the final test set are shown in the last two columns of Table 4. For all the circuits considered, we use a random input sequence of length 1000 as the sequence T_0. The results show that it is not necessary to spend the test generation effort required in [10] and [12] in order to produce scan test sets with low test application times. Specifically, the following points can be seen from Tables 1-5.

Table 5: Results for random sequences

circuit	detected T0	detected scan	detected final	seq length T0	seq length scan	added c.tst
s298	186	274	308	1000	84	14
s344	328	339	342	1000	45	2
s382	49	84	399	1000	54	25
s400	51	87	415	1000	70	22
s526	48	159	554	1000	30	49
s641	340	424	467	1000	85	14
s820	327	389	850	1000	64	73
s1423	628	786	1501	1000	1000	26
s1488	860	1065	1486	1000	89	72
s5378	2919	3352	4563	1000	1000	99
b01	126	135	135	1000	43	0
b02	67	70	70	1000	25	0
b03	292	410	452	1000	79	12
b04	1003	1158	1344	1000	69	27
b06	184	199	202	1000	31	2
b09	95	277	420	1000	17	23
b10	338	422	512	1000	57	29
b11	207	927	1078	1000	87	46

In most cases, a random test sequence T_0 detects significantly fewer faults than the test sequences from [10] or [12]. In some cases, the proposed procedure increases the number of detected faults (by selecting a scan-in state, a scan-out time unit and by omitting vectors) such that the number of additional tests required in Phase 3 is only slightly higher than that required for the test sequences from [10] or [12]. In other cases, the number of faults detected is relatively low, and a large number of additional tests are required in Phase 3. In the latter case, the initial test set is somewhat closer to the initial test set used in [4]. The effect on the number of clock cycles varies. In some cases, the number of clock cycles is higher than that required using the test sequences from [10] or [12]. In other cases, it is lower. Especially when the initial test set of [4] is better than the test set proposed here, using a random sequence that detects fewer faults and using more input sequences of length one improves the results. The average length of the primary input sequences is typically lower than when test sequences from [10] or [12] are used, but still higher than that of [4].

For an overall comparison of test application time, we include in Table 3 the total number of clock cycles obtained by every method. It can be seen that both the initial and the final test sets of the method proposed here require overall a lower number of clock cycles than those of [4].

5. Concluding remarks

We proposed a new approach to the generation of test sets with reduced test application times for scan circuits. The proposed procedure generates a test $\tau_{seq} = (SI_{seq}, T_{seq})$ where T_{seq} is a primary input sequence significantly longer than one. This test detects a large percentage of the circuit faults. In addition, if necessary, the test set consists of tests $\tau_i = (SI_i, T_i)$ where T_i includes a single primary input vector. These tests are added to detect faults that are left undetected by τ_{seq}. The proposed procedure was motivated by the observation that an existing static compaction procedure reduces the test application time of a test set by combining tests, thus reducing the number of tests and increasing the lengths of their primary input sequences. With the proposed procedure, the resulting test set has characteristics similar to those of a test set obtained after compaction. We demonstrated through experimental results the advantages of this approach over earlier ones as a method for generating test sets with minimal test application time. We also showed that the proposed procedure generates longer primary input sequences. These sequences are applied at-speed, and the use of longer sequences enhances the detection of delay defects.

References

[1] D. K. Pradhan and J. Saxena, "A Design for Testability Scheme to Reduce Test Application Time in Full Scan", in Proc. 10th VLSI Test Symp., April 1992, pp. 55-60.

[2] S. Y. Lee and K. K. Saluja, "An Algorithm to Reduce Test Application Time in Full Scan Designs", in Proc. 1992 Intl. Conf. on Computer-Aided Design, Nov. 1992, pp. 17-20.

[3] S. Y. Lee and K. K. Saluja, "Test Application Time Reduction for Sequential Circuits with Scan", IEEE Trans. on Computer-Aided Design, Sept. 1995, pp. 1128-1140.

[4] 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.

[5] P. C. Maxwell, R. C. Aitken, K. R. Kollitz and A. C. Brown, "IDDQ and AC Scan: The War Against Unmodelled Defects", in Proc. 1996 Intl. Test Conf., Oct. 1996, pp. 250-258.

[6] "Best Methods for At-Speed Testing?", Panel 3, 16th VLSI Test Symp., April 1998, p. 460.

[7] I. Pomeranz and S. M. Reddy, "Reducing Test Application Time for Full Scan Circuits by the Addition of Transfer Sequences", in Proc. 9th Asian Test Symp., Nov. 2000, pp. 317-322.

[8] I. Pomeranz and S. M. Reddy, "On Static Compaction of Test Sequences for Synchronous Sequential Circuits", in Proc. 33rd Design Autom. Conf., June 1996, pp. 215-220.

[9] S. Kajihara, I. Pomeranz, K. Kinoshita and S. M. Reddy, "Cost-Effective Generation of Minimal Test Sets for Stuck-at Faults in Combinational Logic Circuits", IEEE Trans. on Computer-Aided Design, Dec. 1995, pp. 1496-1504.

[10] M. S. Hsiao, E. M. Rudnick, and J. H. Patel, "Sequential Circuit Test Generation Using Dynamic State Traversal", in Proc. 1997 Europ. Design & Test Conf., March 1997, pp. 22-28.

[11] I. Pomeranz and S. M. Reddy, "Vector Restoration Based Static Compaction of Test Sequences for Synchronous Sequential Circuits", in Proc. Intl. Conf. on Computer Design, Oct. 1997, pp. 360-365.

[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.