On the Use of Multiple Fault Detection Times in a Method for Built-In Test Pattern Generation for Synchronous Sequential Circuits+

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

Abstract

The first time unit where a fault in a synchronous sequential circuit is detected by a given test sequence T_0 is used by various procedures. One such procedure selects input sequences that are loaded onto an on-chip memory and used as seeds for built-in test pattern generation. Each input sequence is constructed based on a different fault f and is extracted from T_0 around the first detection time of f. In this work, we extend this procedure to consider multiple time units where every target fault f is detected by T_0 in order to select a shorter sequence based on f. The result is reduced storage requirements and test application time for this built-in test pattern generation approach.

1. Introduction

Various procedures for synchronous sequential circuits accept a test sequence T_0, and use as part of their computation the first time unit where each fault is detected by T_0. The built-in test pattern generation method of [1] loads short input sequences into an on-chip memory. Each loaded sequence is expanded on-chip using special hardware. The input sequences to be loaded are computed such that each sequence, after it is expanded on-chip, detects at least one yet-undetected target fault. To ensure this property, the loaded sequences are constructed from a deterministic test sequence T_0. Specifically, a loaded sequence aimed at detecting a target fault f is computed from the shortest subsequence of T_0 that ends at the first time unit where f is detected. Hence the importance of the first detection time of every fault for this procedure.

Another application where the first detection time of a fault is important is that of static compaction. The static compaction procedures of [2] and [3], and procedures derived from them (e.g., [4], [5]), accept a test sequence T_0, and attempt to remove from T_0 as many test vectors as possible without losing fault coverage. These procedures use the first detection time of a fault in order to identify test vectors that need to remain in T_0 so as to retain its fault coverage.

In the applications above, it is convenient to use the first detection time of a fault since it can be identified by conventional fault simulation that drops the fault as soon as it is detected. However, a fault may be detected again at later time units, and sequences extracted around later detection times may be shorter or otherwise more suitable for the application being considered. In terms of fault simulation cost, it is not necessary to perform non-dropping fault simulation in order to find all the detection times of *all* the faults. In all the applications listed above, only a limited number of faults will be selected as target faults and require non-dropping fault simulation. In addition, it is possible to use *all the time units* following the first detection time of a fault. In this case, non-dropping fault simulation is not required.

In this work, we investigate several methods to use multiple detection times of a fault, or multiple time units following the first detection time of a fault, in order to reduce the sequence lengths that need to be loaded by the built-in test pattern generation method of [1]. Reducing the loaded sequence lengths reduces the storage requirements and test application time of this method in several ways. (1) At any given time, a shorter sequence needs to be loaded and stored in the on-chip memory. Thus, a smaller on-chip memory is required. (2) The total lengths of all the sequences that need to be loaded is smaller. Thus, the total loading time is reduced. (3) The overall lengths of the expanded sequences applied to the circuit-under-test is lower. Thus, the test application time is reduced. The reductions in storage requirements and test application time are achieved without affecting the fault coverage, which remains complete.

Other built-in test generation methods for synchronous sequential circuits were described in [6]-[9]. The methods of [6] and [7] rely only on on-chip test sequence generation, and do not achieve complete fault

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

coverage. The method of [8] uses encoding techniques to reduce the memory requirements of a deterministic test sequence. The same encoding techniques can be used for the input sequences loaded by the procedure of [1]. The method of [9] relies on *DFT* in the form of partial reset and observation points to achieve complete fault coverage.

The paper is organized as follows. In Section 2, we review the method of [1]. In Sections 3 to 5, we present extensions to the procedure of [1] aimed at reducing the storage requirements. In Section 6, we present experimental results comparing the various extensions with the procedure of [1]. Section 7 concludes the paper.

2. The built-in test generation method of [1]

Under the built-in test generation method of [1], short input sequences are loaded into an on-chip memory. Each input sequence, after it is loaded, is expanded on-chip into a test sequence. Together, the expanded test sequences achieve the same fault coverage achieved by a given test sequence that was generated off-chip.

We use the following notation to describe this method. An input sequence to be loaded into an on-chip memory is denoted by S. The expanded version of S obtained on-chip is denoted by S_{exp}. The set of all the sequences S is denoted by Σ. The set of all the expanded sequences S_{exp} is denoted by Σ_{exp}.

The sequence S_{exp} is obtained from S as follows. The sequence S is repeated n times to obtain $S'_{exp} = S^n$, where A^n is the sequence A repeated n times. The sequence S'_{exp} is complemented to obtain $S''_{exp} = S'_{exp} \cdot \overline{S'_{exp}}$ (here, $\cdot$ stands for concatenation). The sequence S''_{exp} is repeated, this time shifting every vector by one position to the left. We obtain $S'''_{exp} = S''_{exp} \cdot S''_{exp} \ll 1$, where $A \ll 1$ is the sequence A with every vector shifted to the left by one position. Finally, S'''_{exp} is reversed to obtain $S_{exp} = S'''_{exp} \cdot rS'''_{exp}$, where rA is the sequence A after reversing the order of its vectors. For example, starting from the sequence S shown in Table 1 and using $n = 2$, we obtain the sequence S_{exp} also shown in Table 1.

Table 1: An example of S_{exp}

S	000	110						
S_{exp}	000	110	000	110	111	001	111	001
	000	101	000	101	111	010	111	010
	010	111	010	111	101	000	101	000
	001	111	001	111	110	000	110	000

To ensure that complete fault coverage would be obtained by the sequences in Σ_{exp}, the set Σ is derived based on a test sequence T_0 that was generated off-chip and achieves the desired fault coverage. We assume that T_0 detects all the target faults when the fault free and faulty circuits start from the all-unspecified initial states. The sequences S_{exp} are also applied under the same assumption. The set Σ consists of selected subsequences of T_0 that are further compacted to reduce their lengths. Each one of the sequences is expanded before it is applied to the circuit, such that the total length of the expanded sequences may exceed the length of T_0. However, at any given time, the length of a stored sequence is much shorter than T_0. In addition, the total length of the sequences is typically smaller than the length of T_0.

Each sequence in Σ is loaded into an on-chip memory at the tester speed. After each sequence $S \in \Sigma$ is loaded, its expanded sequence S_{exp} is produced on-chip and applied to the circuit at-speed. In this scheme, the size of the memory that stores the sequences in Σ need only be large enough to hold the longest sequence contained in Σ. There is no need to hold the circuit state during the loading of a sequence, since the sequences are computed under the assumption that the circuit state is unknown before the application of each expanded sequence. The memory used for storing the loaded sequences may be a memory that already exists on the chip, or a memory added for test application.

Next, we review the procedure for constructing Σ. The procedure is given as Procedure 1 below. The given test sequence T_0 is first simulated. The set of faults detected by T_0 is denoted by F. For every fault $f \in F$, the first time unit where f is detected is denoted by $u_{det}(f)$. The set F_{targ} contains all the faults in F which are not yet detected by the expanded versions of the sequences in Σ. Initially, $\Sigma = \phi$, and $F_{targ} = F$. In each iteration of Procedure 1, a fault $f \in F_{targ}$ is considered. A sequence S is constructed based on f by calling Procedure 2 (described below). Procedure 2 constructs S such that its expanded version S_{exp} detects f. The sequence S is added to Σ. The expanded sequence S_{exp} is fault simulated, and all the faults it detects are dropped from F_{targ}. Thus, in each iteration of Procedure 1, at least one additional fault is detected, and the procedure is guaranteed to terminate with a set of sequences Σ such that Σ_{exp} detects every fault in F.

The fault f selected for the construction of the next input sequence S is the one with the highest detection time $u_{det}(f)$ of all the faults in F_{targ}. The reason for this choice is that faults with higher detection times tend to be more difficult to detect, and test sequences that detect them tend to be longer, and tend to detect a larger number of additional faults.

Procedure 1: Sequence selection

(1) Simulate T_0 to find the set of detected faults F. For every $f \in F$, store in $u_{det}(f)$ the first time unit where f is detected. Set $F_{targ} = F$. Set $\Sigma = \phi$.

(2) Let $u_{det,max} = \max\{u_{det}(f): f \in F_{targ}\}$. Select a fault $f \in F_{targ}$ such that $u_{det}(f) = u_{det,max}$.

(3) Call Procedure 2 below to construct a sequence S based on f. Add S to Σ.

(4) Simulate the faults in F_{targ} under the expanded version S_{exp} of S. Drop from F_{targ} every fault which is detected by S_{exp}.

(5) If $F_{targ} \neq \phi$, go to Step 2.

To describe Procedure 2, we denote by $T_0[u_1, u_2]$ the subsequence of T_0 that starts at time unit u_1 and ends at time unit u_2. The goal of Procedure 2 is to find the shortest possible sequence S such that S_{exp} detects f. Procedure 2 first finds a time unit u_{start} and a subsequence $S = T_0[u_{start}, u_{det}(f)]$ of T_0 such that the expanded version S_{exp} of S detects f. For this purpose, the procedure starts with $u_{start} = u_{det}(f)$, and reduces u_{start} until f is detected by S_{exp}. Thus, the procedure considers $T_0[u_{det}(f), u_{det}(f)]$, $T_0[u_{det}(f)-1, u_{det}(f)]$, $T_0[u_{det}(f)-2, u_{det}(f)]$, $\cdots$, $T_0[0, u_{det}(f)]$, and selects the first sequence whose expanded version detects f. We note that it is always possible to find such a sequence S. In the worst case, $S = T_0[0, u_{det}(f)]$ will detect f when $u_{start} = 0$ (this is because T_0 detects f at time unit $u_{det}(f)$, and the expanded sequence S_{exp} will also detect f in this case).

Once S is found, we further reduce its length by omitting test vectors from it. The test vector at time unit u of S, denoted by $S[u]$, can be omitted if, after the omission, S_{exp} still detects f. Procedure 2 considers all the test vectors of S in a random order. If S_{exp} after the omission of $S[u]$ from S detects f, the omission is accepted, and S is redefined to be the sequence without the omitted vector. Following this, all the time units along the sequence are considered again. Otherwise, if $S[u]$ cannot be omitted, $S[u]$ is restored into S. The procedure terminates when all the time units have been considered without being able to omit any additional vector.

The set Σ obtained by Procedure 1 may contain redundant sequences whose detected faults are also detected by other sequences. We therefore apply to Σ a procedure similar to reverse order simulation in order to drop as many redundant sequences as possible.

3. Using all the detection times

In Procedure 2, a sequence S based on a fault f is computed such that the detection time of f, $u_{det}(f)$, defines the end of the sequence. Procedure 2 finds a time unit u_{start} such that the expanded version of the sequence $S = T_0[u_{start}, u_{det}(f)]$ detects f. Once u_{start} is found, S is further compacted by omitting test vectors from it. By using $u_{det}(f)$ as the end of S, it is guaranteed that it will be possible to find S that detects f.

To compute $u_{det}(f)$, Procedure 1 performs fault simulation with fault dropping of the sequence T_0 in Step 1. Thus, Procedure 1 finds the first detection time of every fault. However, a fault may be detected by T_0 more than once. If a fault f is detected by T_0 at time units $u_{det,1}(f), u_{det,2}(f), \cdots, u_{det,m}(f)$, then Procedure 2 can be used to construct m different sequences $S_1, S_2, \cdots, S_m$, each one ending at a different detection time $u_{det,i}(f)$ of f. Procedure 2 proceeds as follows when provided with a detection time $u_{det,i}(f)$. The procedure first finds a time unit $u_{start,i}$ such that the expanded version of the sequence $S_i = T_0[u_{start,i}, u_{det,i}(f)]$ detects f. Once $u_{start,i}$ is found, S_i is further compacted by omitting test vectors from it. Assuming that $u_{det,1}(f)$ is the lowest detection time of f, we have that $u_{det,1}(f) = u_{det}(f)$ is the detection time computed by Procedure 1, and $S_1 = S$ is the sequence computed by Procedure 2 based on $u_{det}(f)$.

After generating sequences $S_1, S_2, \cdots, S_m$ based on $u_{det,1}(f), u_{det,2}(f), \cdots, u_{det,m}(f)$, Procedure 1 can select the shortest sequence S_i, and include it in Σ. In this way, the maximum length of any sequence in Σ is potentially reduced, and the total length of all the sequences in Σ is reduced.

To find $u_{det,1}(f), u_{det,2}(f), \cdots, u_{det,m}(f)$, it is necessary to simulate f under T_0 without fault dropping. However, this does not need to be done in advance for all the faults. Rather, fault simulation without fault dropping can be done for f only when f is selected as the next fault to be considered by Procedure 1. As a result, only a limited number of faults targeted explicitly by Procedure 1 will require non-dropping fault simulation. The modified Procedure 1 is given next.

Procedure 1_all_det_times: Sequence selection considering all the detection times

(1) Simulate T_0 to find the set of detected faults F. For every $f \in F$, store in $u_{det}(f)$ the first time unit where f is detected. Set $F_{targ} = F$. Set $\Sigma = \phi$.

(2) Let $u_{det,max} = \max\{u_{det}(f): f \in F_{targ}\}$. Select a fault $f \in F_{targ}$ such that $u_{det}(f) = u_{det,max}$.

(3) Simulate f under T_0 to find all the detection times of f.

(4) For every detection time $u_{det,i}(f)$:
Call Procedure 2 to construct a sequence S_i based on f and $u_{det,i}(f)$.

(5) Select the shortest sequence S_i computed in Step 4. Add S_i to Σ.

(6) Simulate the faults in F_{targ} under the expanded version $S_{i,exp}$ of S_i. Drop from F_{targ} every fault which is detected by $S_{i,exp}$.

(7) If $F_{targ} \neq \phi$, go to Step 2.

4. All the time units after the first detection

In Procedure 1_all_det_times, we consider all the detection times of a fault. The use of a detection time $u_{det,i}(f)$ to define the end of the sequence S_i ensures that the fault can be detected by S_i. In this section, we note that if f is detected by T_0 for the first time at time unit $u_{det}(f)$, then any time unit $u_{end,i} \geq u_{det}(f)$ can be used as the last time unit of a sequence S_i, and still guarantee that f would be detected by $S_{i,exp}$. Procedure 2 can still find a time unit $u_{start,i}$ such that $S_i = T_0[u_{start,i}, u_{end,i}]$ detects f. In the worst case, $u_{start,i} = 0$ will be used. It is then possible to further compact S_i by omitting test vectors from it.

Using $u_{end,i} > u_{det}(f)$, it is possible that $u_{start,i}$ obtained for $u_{end,i}$ would be the same as u_{start} obtained for $u_{det}(f)$. This situation is illustrated in Figure 1. However, even in this case, it is possible that using $u_{end,i} > u_{det}(f)$ will result in a shorter sequence S_i than the sequence S obtained for $u_{det}(f)$. This is because of the omission of test vectors from S and S_i after u_{start} and $u_{start,i}$ are found.

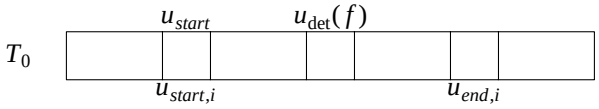

Figure 1: An example using $u_{end,i} > u_{det}(f)$

The modified Procedure 1 that considers every time unit $u_{end,i} \geq u_{det}(f)$ and selects the shortest sequence does not need to explicitly compute every detection time of f. Thus, it does not require non-dropping fault simulation.

5. All the time units after the first detection or until the maximum sequence length is reached

Consideration of all the time units following the detection time of a fault may result in the generation of a large number of sequences for the fault, from which the shortest one will be selected. It is possible to reduce the number of sequences generated without increasing the maximum length of any sequence in Σ by the following extension.

Let the maximum length of any sequence in Σ be $L_{\max}$. Initially, when Σ is empty, we have $L_{\max} = 0$. When a fault f is considered at an arbitrary iteration, we generate candidate sequences S_i based on time units $u_{end,i} = u_{det}(f), u_{det}(f)+1, \cdots, L-1$, where L is the length of T_0. We generate sequences S_i for increasing values of $u_{end,i}$ until we find the first sequence S_{i_0} whose length does not exceed $L_{\max}$, or until $u_{end,i} = L-1$ has been considered. In the former case, we include S_{i_0} in Σ without considering additional values of $u_{end,i}$. The following example demonstrates the reduction in the number of sequences generated.

Consider a circuit with seven faults $f_0, f_1, \cdots, f_6$. Let $u_{det}(f_0) = u_0$, $u_{det}(f_1) = u_{det}(f_2) = u_{det}(f_3) = u_1$, $u_{det}(f_4) = u_{det}(f_5) = u_2$ and $u_{det}(f_6) = u_3$. Suppose that $u_0 < u_1 < u_2 < u_3$, and that u_3 is the last time unit of T_0. Initially, $\Sigma = \phi$ and $L_{\max} = 0$. Since faults are considered by all the versions of Procedure 1 from the highest detection time to the lowest, f_6 is considered first. A single sequence is generated for f_6 based on time unit u_3. This sequence will be included in Σ, and $L_{\max}$ will be set equal to its length. Suppose that f_4 and f_6 are detected by the expanded sequence. The next fault to be considered is f_5. For f_5, sequences are generated based on time units $u_2, u_2+1, \cdots, u_3$, or until the first sequence of length $L_{\max}$ or shorter is generated. Suppose that this happens for u_2+1. In this case, instead of generating $u_3 - u_2 + 1$ candidate sequences for f_5, we only generate two sequences, and include the second one in Σ. Suppose that f_0, f_1, f_2 and f_5 are detected by the expanded version of the sequence selected for f_5. The next fault to be considered is f_3. For f_3, sequences are generated based on time units $u_1, u_1+1, \cdots, u_3$, or until the first sequence of length $L_{\max}$ or shorter is generated. There are $u_3 - u_1 + 1$ candidate time units for f_3; however, not all of them may have to be considered. After adding the sequence for f_3 to Σ, all the target faults are detected, and the procedure terminates.

The modified Procedure 1 is given next.

Procedure 1_all_times_max_len: Sequence selection considering all the time units after the first detection time or until the maximum sequence length is reached

(1) Simulate T_0 to find the set of detected faults F. For every $f \in F$, store in $u_{det}(f)$ the first time unit where f is detected. Set $F_{targ} = F$. Set $\Sigma = \phi$. Set $L_{\max} = 0$.

(2) Let $u_{det,max} = \max\{u_{det}(f): f \in F_{targ}\}$. Select a fault $f \in F_{targ}$ such that $u_{det}(f) = u_{det,max}$.

(3) For every time unit $u_{end,i} \geq u_{det}(f)$:
Call Procedure 2 to construct a sequence S_i based on f and $u_{end,i}$. If the length of S_i is $L_{\max}$ or shorter, go to Step 4.

(4) Select the shortest sequence S_i computed in Step 3. Add S_i to Σ.

(5) Simulate the faults in F_{targ} under the expanded version $S_{i,exp}$ of S_i. Drop from F_{targ} every fault which is detected by $S_{i,exp}$.

(6) If $F_{targ} \neq \phi$, go to Step 2.

Procedure 1_all_times_max_len will minimize the maximum length of the sequences in Σ similar to the procedure of the previous section; however, it may result in

longer sequences for faults that require shorter than the maximum length. This may result in a larger total length of the sequences in Σ.

6. Experimental results

In [1], experimental results were reported for ISCAS-89 benchmark circuits. The test sequences used as T_0 in the construction of Σ were the ones generated by *STRATEGATE* [10] and compacted by the static compaction procedure of [3]. For each circuit, a number of repetitions n was selected in [1] to construct the set of expanded sequences Σ_{exp}. We use the same test sequences T_0 and the same values of n in the experiments reported here. In addition, we apply the various procedures to several ITC-99 benchmarks. As test sequences for these circuits, we use the ones generated by the test generation procedure *PROPTEST* [11].

In Tables 2 and 3, we show the results for ISCAS-89 benchmark circuits. In Table 2, we include the results of the basic procedure from [1] (Procedure 1), and the extension proposed here that uses all the detection times of a fault to construct a sequence S for the fault (Procedure 1_all_det_times). After the circuit name, we show the length of the test sequence T_0, and the fault coverage it achieves (the same fault coverage is achieved by the proposed procedures). We then show the number of repetitions n. For each of the two procedures being compared, we show under column *seq* the number of sequences in Σ. The total length of the sequences in Σ is shown under column *tot*.*len*, followed by the ratio of the total length to the length of the original sequence T_0. For example, for *s*298 under the basic procedure, the ratio is 0.23, implying that a total of 0.23 of the original sequence needs to be loaded into the chip to achieve the same fault coverage as the original sequence. The maximum length of any sequence in Σ is shown under column *max*.*len*, followed by the ratio of the maximum length to the length of T_0. For example, for *s*298 under the basic procedure, the ratio is 0.15, implying that at any given time, only 0.15 of the original sequence needs to be stored on-chip. In the last row, we show the sum of all the lengths in columns showing test lengths, and average values for columns showing ratios. In Table 3, we include the results for the procedure that uses all the time units following the first detection time or until the maximum sequence length is reached (Procedure 1_all_times_max_len).

In Table 4, we show the normalized run times of Procedure 1 and its extensions, and of the compaction procedure used for dropping redundant sequences from Σ (this procedure was applied following Procedure 1 and every one of its extensions). The run time is normalized by dividing it by the time it takes to fault simulate T_0.

Table 2: All the detection times (ISCAS-89 benchmarks)

	orig			basic proc.				all det.times		
circuit	len	f.c.	n	seq	tot len	max len		seq	tot len	max len
s298	117	86.04	16	4	27 0.23	17 0.15		5	16 0.14	8 0.07
s344	57	96.20	8	5	14 0.25	6 0.11		5	14 0.25	6 0.11
s382	516	91.23	16	5	272 0.53	94 0.18		7	264 0.51	94 0.18
s400	611	90.26	16	5	259 0.42	100 0.16		5	174 0.28	100 0.16
s526	1006	81.80	16	9	637 0.63	122 0.12		10	468 0.47	92 0.09
s641	101	86.51	16	13	29 0.29	8 0.08		13	33 0.33	8 0.08
s820	491	95.76	4	45	454 0.92	15 0.03		47	447 0.91	15 0.03
s1196	238	99.76	4	100	137 0.58	2 0.01		100	136 0.57	2 0.01
s1423	1024	93.33	8	21	422 0.41	82 0.08		30	459 0.45	59 0.06
s1488	455	97.17	8	15	220 0.48	44 0.10		19	122 0.27	25 0.05
s5378	646	79.06	8	38	326 0.50	29 0.04		38	318 0.49	25 0.04
tot/ave	5262				2797 0.48	519 0.10			2451 0.42	434 0.08

Table 3: All the time units after the first detection time (ISCAS-89 benchmarks)

	orig		all times max.len				
circuit	len	n	seq	tot len		max len	
s298	117	16	4	14	0.12	7	0.06
s344	57	8	6	11	0.19	4	0.07
s382	516	16	6	272	0.53	88	0.17
s400	611	16	6	56	0.09	15	0.02
s526	1006	16	11	607	0.60	89	0.09
s641	101	16	18	25	0.25	2	0.02
s820	491	4	45	417	0.85	12	0.02
s1196	238	4	100	137	0.58	2	0.01
s1423	1024	8	24	496	0.48	39	0.04
s1488	455	8	20	165	0.36	16	0.04
s5378	646	8	41	314	0.49	21	0.03
tot/ave	5262			2514	0.41	295	0.05

Normalization allows for a more fair comparison between the different procedures.

Table 4: Comparison of run times

	basic proc.		all det.times		all times max.len	
circuit	Proc.1	comp.	Proc.1	comp.	Proc.1	comp.
s298	30.62	64.59	54.93	37.35	215.27	33.69
s344	10.99	19.16	24.15	19.34	109.63	20.95
s382	308.27	137.66	2970.16	89.81	17935.48	125.42
s400	224.93	147.31	5168.93	114.03	67590.02	35.30
s526	328.57	93.67	37307.94	72.72	62674.54	75.81
s641	43.76	62.44	192.99	58.94	255.62	46.98
s820	83.03	71.49	544.72	69.79	565.29	72.44
s1196	13.27	47.14	14.24	47.16	13.46	47.27
s1423	103.10	56.45	905.78	46.14	4687.40	71.17
s1488	41.16	77.17	17069.81	50.98	664.31	63.13
s5378	9.46	20.74	2055.16	20.17	85.01	21.82

Results for ITC-99 benchmark circuits are given in Tables 5 and 6.

It can be seen that the extensions of Procedure 1 allow the total length of the sequences in Σ and the maxi-

Table 5: All the detection times (ITC-99 benchmarks)

	orig			basic proc.			all det.times		
circuit	len	f.c.	n	seq	tot len	max len	seq	tot len	max len
b01	66	98.52	8	3	11 0.17	5 0.08	3	11 0.17	5 0.08
b02	45	97.14	4	2	11 0.24	6 0.13	3	10 0.22	5 0.11
b03	136	73.89	8	8	43 0.32	10 0.07	9	41 0.30	7 0.05
b04	168	86.78	4	21	78 0.46	7 0.04	18	72 0.43	7 0.04
b06	37	92.08	2	2	6 0.16	3 0.08	2	6 0.16	3 0.08
b09	279	80.71	8	4	84 0.30	32 0.11	5	93 0.33	32 0.11
b10	190	91.21	8	12	77 0.41	16 0.08	13	78 0.41	16 0.08
b11	676	91.55	8	11	150 0.22	28 0.04	19	97 0.14	19 0.03
tot/ave	1597				460 0.29	107 0.08		408 0.27	94 0.07

Table 6: All the time units after the first detection time (ITC-99 benchmarks)

	orig			all times max.len				
circuit	len	f.c.	n	seq	tot len		max len	
b01	66	98.52	8	3	11	0.17	5	0.08
b02	45	97.14	4	3	10	0.22	5	0.11
b03	136	73.89	8	10	44	0.32	7	0.05
b04	168	86.78	4	22	72	0.43	5	0.03
b06	37	92.08	2	2	6	0.16	3	0.08
b09	279	80.71	8	4	57	0.20	17	0.06
b10	190	91.21	8	12	77	0.41	16	0.08
b11	676	91.55	8	16	104	0.15	13	0.02
tot/ave	1597				381	0.26	71	0.06

mum length of any sequence in Σ to be reduced at the cost of increased run times. The reduction in the maximum sequence lengths is especially significant when Procedure 1_all_times_max_len is used. This contributes to reductions in storage requirements, as discussed above. On the average, for ISCAS-89 benchmark circuits, the maximum sequence length is reduced approximately by a factor of 20 compared to the original sequence length and by a factor of 2 compared to the results of the basic procedure from [1].

7. Concluding remarks

A built-in test pattern generation method proposed before was based on loading input sequences and expanding them on-chip into test sequences. Each input sequence was extracted from a fault detection test sequence T_0, and ensured the detection of a yet-undetected fault. An input sequence aimed at detecting a target fault f was computed from the shortest subsequence of T_0 that ended at the first time unit where f was detected by T_0. We extended this procedure to consider multiple time units where a fault f is detected. In addition, we considered time units u that follow the first detection time of f even if f is not detected at time unit u. This resulted in several candidate sequences for every fault f. Of all the candidate sequences, we selected the shortest one. As a result, the storage requirements for built-in test pattern generation as well as the test application time were reduced compared to those of the basic procedure.

References

[1] I. Pomeranz and S. M. Reddy, "Built-In Test Sequence Generation for Synchronous Sequential Circuits Based on Loading and Expansion of Test Subsequences", in Proc. 36th Design Autom. Conf., June 1999, pp. 754-759.

[2] 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.

[3] 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.

[4] R. Guo, I. Pomeranz and S. M. Reddy, "Procedures for Static Compaction of Test Sequences for Synchronous Sequential Circuits Based on Vector Restoration", in Proc. Conf. on Design Autom. and Test in Europe, Feb. 1998, pp. 583-587.

[5] S. K. Bommu, S. T. Chakradhar and K. B. Doreswamy, "Static Test Sequence Compaction based on Segment Reordering and Accelerated Vector Restoration", in Proc. 1998 Intl. Test Conf., Oct. 1998, pp. 954-961.

[6] 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.

[7] I. Pomeranz and S. M. Reddy, "Built-In Test Generation for Synchronous Sequential Circuits", in Proc. Intl. Conf. on Computer-Aided Design, Nov. 1997, pp. 421-426.

[8] V. Iyengar, K. Chakrabarty, and B. T. Murray "Built-in Self Testing of Sequential Circuits Using Precomputed Test Sets," in Proc. VLSI Test Symp., April 1998, pp. 418-422.

[9] M.-L. Flottes, C. Landrault and A. Petitqueux, "Partial Set for Flip-Flops Based on State Requirement for Non-scan BIST Scheme", in Proc. Europ. Test Workshop, May 1999, pp. 104-109.

[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] 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.