An Approach for Improving the Levels of Compaction Achieved by Vector Omission+

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

Abstract

We describe a method referred to as sequence counting to improve on the levels of compaction achievable by vector omission based static compaction procedures. Such procedures are used to reduce the lengths of test sequences for synchronous sequential circuits without reducing the fault coverage. The unique feature of the proposed approach is that test vectors omitted from the test sequence can be reintroduced at a later time. Reintroducing of vectors helps reduce the compacted test sequence length beyond the length that can be achieved if vectors are omitted permanently. Experimental results are presented to demonstrate the levels of compaction achieved by the sequence counting approach.

1. Introduction

Static test compaction is used for reducing the length of a test sequence without reducing its fault coverage. Reduced test lengths result in reduced tester memory and test application time requirements. The study of compacted test sequences can also help in understanding the reasons for inefficiencies in test generators that produced the original (longer) test sequences. In this paper, we propose a method to improve the compaction levels achieved by vector omission based static compaction procedures.

Static test compaction based on vector omission was first proposed in [1]. Static compaction based on vector omission removes (or omits) test vectors from the test sequence as long as this can be done without reducing the fault coverage. A variation of vector omission, called vector restoration, was proposed in [2]. Under a vector restoration procedure, all (or most) of the test vectors are first omitted from the test sequence. Vectors are then restored into the sequence as necessary to restore the fault coverage. Many efficient implementations of vector omission and restoration exist [3]-[8]. These procedures attempt to achieve the same levels of compaction as the procedures of [1] and [2], however, at faster run times.

In this work, we introduce a method called *sequence counting* to improve on the levels of compaction that can be achieved by static compaction procedures based on vector omission. We introduce this method using a specific implementation similar in efficiency to the procedures in [1] and [2]. We expect efficient implementations to be devised later, implementing the same concept at significantly reduced run times.

Similar to the vector omission and vector restoration approaches of [1] and [2], the sequence counting approach maintains the original order of the test vectors. Thus, given a test sequence $T = (t_1, t_2, \cdots, t_L)$, the compacted sequence obtained by either one of the three approaches has the form $T_c = (t_{i_1}, t_{i_2}, \cdots, t_{i_M})$, where $i_1 < i_2 < \cdots < i_M$. The sequence counting approach is different from the vector omission and vector restoration approaches in the following way. Once an iteration of the vector omission or vector restoration approaches terminates, the vectors they omitted (or the vectors that were not restored in the case of the vector restoration procedure) are omitted from the test sequence permanently, and they are not reintroduced at a later time. Under the sequence counting approach, it is possible to reintroduce an omitted vector. For example, if a compacted sequence $T_c = (t_1, t_2, t_4, t_5, \cdots)$ is obtained under the sequence counting approach, the vector t_3 is not permanently excluded from T_c. It may be possible at a later time to obtain the compacted sequence $T_c = (t_1, t_3, t_4, t_5, \cdots)$ (or other variations that include t_3), and potentially omit additional vectors that cannot be omitted if t_3 is excluded. Consequently, the sequence counting approach is less greedy, and performs a more global search for the shortest possible compacted sequence. As a result, it has the potential of producing shorter test sequences.

The paper is organized as follows. In Section 2 we describe the basic idea behind the sequence counting approach. In Section 3 we describe the basic step of the sequence counting approach. In Section 4 we describe a specific compaction procedure based on the sequence counting approach. Experimental results of this procedure are given in Section 5. Section 6 concludes the paper.

2. The basic idea

The basic idea behind the sequence counting approach is illustrated in Figure 1. The first row of Figure 1 depicts a sequence $T = (t_1, t_2, \cdots, t_{10})$ of length 10. The order of the vectors in the sequence is marked by the edges in Figure 1. The vector t_0 is a dummy vector indicating the beginning of the sequence. Under a vector t_i in the first row of Figure 1, we show vectors that may be included in a compacted version of T at time unit i, instead of t_i, if t_i or vectors preceding it are omitted. For example, instead of t_1, a compacted sequence may include t_2, t_3, $\cdots$ or t_{10} at time unit 1. In general, at time unit i, the sequence may include t_{i+1}, t_{i+2}, $\cdots$, t_{10} instead of t_i, depending on the omitted vectors that precede t_i (including t_i itself).

A compacted sequence can be represented by a sequence of edges such as the one shown by solid lines in Figure 2. The compacted sequence shown by solid lines in Figure 2 is $T_c = (t_2, t_3, t_4, t_7, t_8, t_{10})$. In general, the edges defining a compacted sequence can go from t_i to t_{i+1} by taking one step to the right, or from t_i to a vector t_j such that $j > i$ by taking one step to the right and any number of steps downward. For example, in the sequence shown by solid lines in Figure 2, we go from t_2 to t_3 by taking one step to the right; and we go from t_4 to t_7 by tak-

+ Research supported in part by NSF Grant No. MIP-9725053.

ing one step to the right and two steps downward. A step downward that skips over l vectors shortens the test sequence by the same number of vectors. In Figure 2, we skip over a total of four vectors (one from t_0 to t_2, two from t_4 to t_7, and one from t_8 to t_{10}). Consequently, the length of the compacted sequence is shorter than the original sequence length by four vectors.

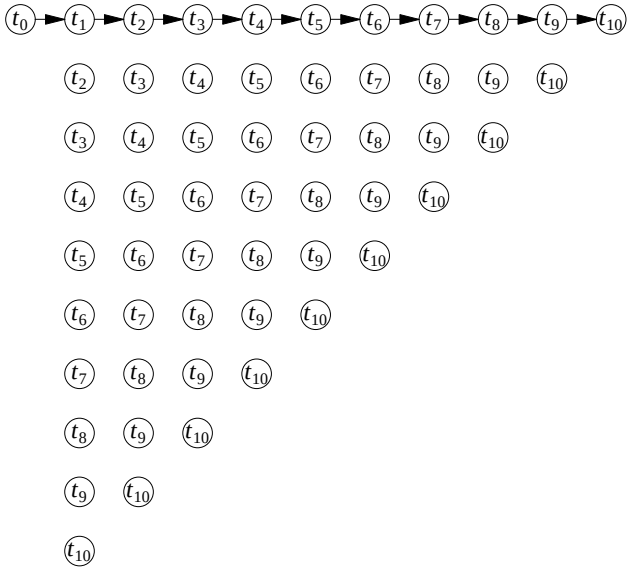

Figure 1: The given test sequence T

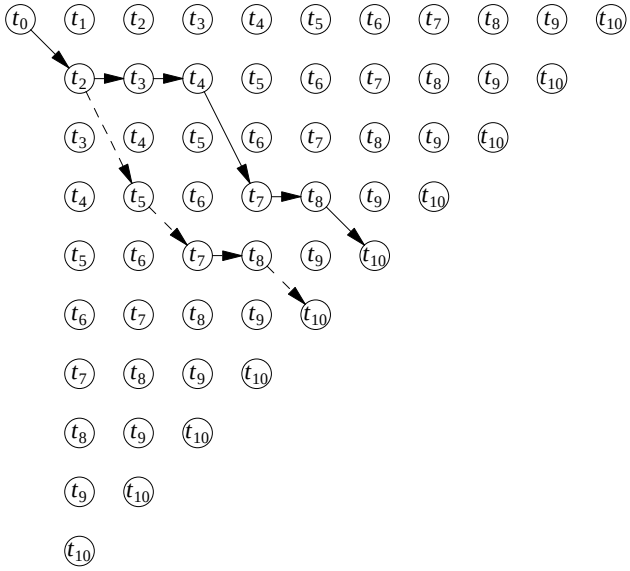

Figure 2: Compacted sequences based on T

To further improve the length of the compacted sequence shown in Figure 2, we can replace a vector t_i that appears at time unit k of the test sequence by a vector t_j, where $j > i$. In Figure 2, this implies lengthening the downward step leading into the vector at time unit k. For example, let us replace t_3 at time unit 2 by t_5. This is shown by the dashed edge from t_2 to t_5 that replaces the solid edge from t_2 to t_3. To complete the sequence, we skip over t_4 (since we already have t_5 at time unit 2, and our goal is to maintain the original order of the vectors). We continue with the first vector that follows t_5 and is included in the sequence shown by solid lines in Figure 2. This vector is t_7. The dashed edge from t_5 to t_7 indicates that t_7 follows t_5 in the new compacted sequence. From t_7, we copy the sequence shown by solid lines in Figure 2. The resulting sequence is shown by the dashed edges in Figure 2. The new compacted sequence is $T_c = (t_2, t_5, t_7, t_8, t_{10})$, and it is shorter by one vector than the previous sequence we obtained. Note that we reintroduced into the compacted sequence the vector t_5 that does not appear in the sequence shown by solid lines in Figure 2.

In the previous example, the step from t_2 to t_5 shortened the compacted sequence. It is also possible to make steps that leave the sequence at the same length. For example, starting from the sequence shown by solid lines in Figure 2, we may replace t_4 by t_5 to obtain the sequence $T_c = (t_2, t_3, t_5, t_7, t_8, t_{10})$ which is also of length six. Such steps are useful in modifying the sequence such that the modified sequence can be compacted further than the unmodified one.

Based on the previous discussion, the steps of the sequence counting approach consist of making downward steps in Figure 1 or 2, or lengthening existing downward steps. This is equivalent to replacing a vector t_i at time unit k by a vector t_j, where $j > i$. The rest of the sequence is then adjusted accordingly. The lower the sequence of edges in Figure 1 or 2, the shorter it is (thus, the dashed sequence of Figure 2 is shorter than the solid sequence of Figure 2, which is shorter than the sequence of Figure 1). After each step that changes the sequence, we resimulate the sequence to ensure that the fault coverage is maintained. Steps that reduce the fault coverage are reversed.

The reason we refer to the approach described above as sequence counting is as follows. Consider the three sequences shown in Figures 1 and 2, $T = (t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10})$, $T_{c_1} = (t_2, t_3, t_4, t_7, t_8, t_{10})$, and $T_{c_2} = (t_2, t_5, t_7, t_8, t_{10})$. Let us consider only the vector indices. From T, we obtain the sequence of vector indices $I = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$; from T_{c_1} we obtain $I_{c_1} = (2, 3, 4, 7, 8, 10)$; and from T_{c_2} we obtain $I_{c_2} = (2, 5, 7, 8, 10)$. The sequences of indices are considered in chronological order during the compaction procedure. Here, a sequence I_1 is higher in the chronological order than a sequence I_2 if, scanning the sequences from left to right and looking for the first position where the two sequences are different, I_1 has a higher number than I_2 in that position. Thus, if we count all possible sequences in chronological order starting from the given uncompacted sequence, the later a sequence appears in the count the shorter it is likely to be. Due to this similarity between counting and compaction, we refer to the proposed approach as sequence counting.

It is important to point out that in the compaction procedure based on sequence counting, we skip over many of the sequences that would have been obtained if all the sequences had been counted in chronological order. In addition, we point out that Figures 1 and 2 are only used to introduce the sequence counting approach. In an implementation of a compaction procedure based on sequence counting, there is no need to store each vector multiple times as in the figures.

Finally, we point out that not every vector may be reintroduced after it is omitted. Let the compacted sequence $T_c = (t_{i_1}, t_{i_2}, \cdots, t_{i_m})$ be obtained. If $i_1 > 1$, then $t_1, \cdots, t_{i_1-1}$ cannot be reintroduced into the sequence. To ensure that low-index vectors are utilized by the sequence counting procedure as much as possible, sequence counting steps that replace the first vector

of the sequence should be small, and should be done only after exhausting all the options with the current initial vector. Alternatively, it is possible to perform steps where the vector t_{i_1} is replaced by a vector t_j such that $j < i_1$. We do not explore these options here.

3. The basic step of sequence counting

Procedure 1 given below describes the basic step of the sequence counting procedure. The original, uncompacted sequence is $T = (t_1, t_2, \cdots, t_L)$. The test sequence on which the sequence counting step is performed is an arbitrary (compacted) sequence $T_c = (t_{i_1}, t_{i_2}, \cdots, t_{i_M})$. In Procedure 1, we randomly select a time unit k, where $1 \leq k \leq M$. We also select a number Δ. We replace t_{i_k} by $t_{i_k+\Delta}$, and then update the sequence T_c as follows.

If $i_k + \Delta > L$, we terminate T_c at time unit $k-1$. For example, suppose that $T = (t_1, \cdots, t_{10})$, $T_c = (t_1, t_3, t_5, t_8, t_9, t_{10})$, $k = 5$ and $\Delta = 2$. Note that $t_{i_1} = t_1$, $t_{i_2} = t_3$, $t_{i_3} = t_5$, and so on. In this case, $i_k = i_5 = 9$, and we replace t_9 by $t_{9+2} = t_{11}$. Since t_{11} does not exist in T, we terminate T_c at time unit 4 and obtain $T_c = (t_1, t_3, t_5, t_8)$.

If $i_k + \Delta < L$, we find the first time unit k' such that $i_{k'} > i_k + \Delta$. We copy the part of T_c from time unit k' to time unit M such that it immediately follows time unit k. For example, suppose that $T = (t_1, \cdots, t_{10})$, $T_c = (t_1, t_3, t_5, t_8, t_9, t_{10})$, $k = 2$ and $\Delta = 4$. In this case, $i_k = i_2 = 3$, and we replace t_3 by $t_{3+4} = t_7$. The first time unit k' where $i_{k'} > 7$ is $k' = 4$ with $i_{k'} = 8$. We copy the subsequence (t_8, t_9, t_{10}) starting at time unit $k+1 = 3$. We obtain the sequence $T_c = (t_1, t_7, t_8, t_9, t_{10})$. As another example, consider the same sequences with $k = 5$ and $\Delta = 1$. In this case, $i_k = i_5 = 9$, and we replace t_9 by $t_{9+1} = t_{10}$. There is no time unit k' with $i_{k'} > 10$, therefore, we do not copy any part of T_c. The resulting sequence is $T_c = (t_1, t_3, t_5, t_8, t_{10})$.

The value of Δ by which we increment i_k is between 1 and $max\{M/10, 1\}$. Thus, we may skip up to $M/10$ vectors, which is a tenth of the length of T_c. Larger constants may be used to reduce the length of T_c faster. Procedure 1 is given next.

Procedure 1: A basic sequence counting step

(1) Let $T = (t_1, t_2, \cdots, t_L)$ be the original test sequence, and let $T_c = (t_{i_1}, t_{i_2}, \cdots, t_{i_M})$.
(2) Randomly select a time unit k, where $1 \leq k \leq M$.
(3) Randomly select a number Δ, where $1 \leq \Delta \leq max\{M/10, 1\}$.
(4) If $i_k + \Delta > L$, terminate T_c at time unit $k-1$, and stop.
(5) Replace t_{i_k} by $t_{i_k+\Delta}$.
(6) Find the first time unit k' such that $i_{k'} > i_k + \Delta$.
(7) Set $m = 0$. While $k' + m \leq M$, copy $t_{i_{k'+m}}$ to time unit $k+1+m$ of T_c and set $m = m+1$.
(8) Terminate T_c at time unit $k+m$.

The complexity of Procedure 1 is $O(M)$, where M is the length of the compacted sequence. This is determined by Step 7 that copies at most M test vectors.

4. The sequence counting procedure

Procedure 1 can be applied as part of a compaction procedure in one of several ways. One of the parameters to consider in applying Procedure 1 is the number of calls to Procedure 1 performed before checking that the fault coverage of the original sequence is maintained. If more than one call to Procedure 1 is performed before considering the fault coverage, then one or more changes introduced by Procedure 1 may have to be undone if the fault coverage is not maintained. The advantage of performing several calls to Procedure 1 before performing fault simulation is that the total time spent on fault simulation may be reduced. In this work, we use an implementation where fault simulation is performed after every call to Procedure 1. The procedure is given as Procedure 2 below.

In Procedure 2, before the current compacted test sequence T_c is modified by Procedure 1, it is stored in a sequence $T_{c,prev}$. Fault simulation is carried out for T_c after it is modified. The previous sequence $T_{c,prev}$ is restored if the fault coverage of T_c, after it is modified by Procedure 1, is smaller than the fault coverage of the original sequence. A variable called n_{same} counts the number of calls to Procedure 1 that do not reduce the test length. This includes calls that reduce the fault coverage and cause the previous compacted sequence to be restored, and calls that keep the fault coverage and the test length at their current levels. The variable n_{same} is reset to zero initially, and every time a call to Procedure 1 yields a step that reduces the test length without reducing the fault coverage. Procedure 2 terminates when n_{same} reaches a preselected constant denoted by N_{SAME}, i.e., after N_{SAME} calls to Procedure 1 that do not reduce the test length.

Procedure 2: A sequence counting procedure

(1) Let $T = (t_1, t_2, \cdots, t_L)$ be the original test sequence. Set $T_c = T$. Set $n_{same} = 0$.
(2) Simulate T. Let the set of detected faults be F_D.
(3) Set $T_{c,prev} = T_c$.
(4) Call Procedure 1 with T and T_c as input.
(5) Simulate T_c under the faults in F_D. Let the set of detected faults be $F_{D,c}$.
(6) If $F_{D,c} \neq F_D$, set $T_c = T_{c,prev}$ and $n_{same} = n_{same} + 1$.
Else, if the length of T_c is smaller than the length of $T_{c,prev}$, set $n_{same} = 0$.
Else, set $n_{same} = n_{same} + 1$.
(7) If $n_{same} < N_{SAME}$, go to Step 3.

When simulating T_c in Step 5 of Procedure 2, it is only necessary to simulate the faults in F_D until the first undetected fault is encountered. An undetected fault causes the step made by Procedure 1 to be rejected, and there is no need to complete the simulation of all the faults in F_D in this case. In addition, if the step made by Procedure 1 changes the sequence at time units k and on, there is no need to simulate faults detected by $T_{c,prev}$ before time unit k. This is because the subsequence until time unit $k-1$ does not change.

The complexity of Procedure 2 depends on the number of iterations it performs. In each iteration, there is a call to Procedure 1 that has complexity $O(M)$, where M is the sequence length, and there is a fault simulation step to determine whether the fault coverage has changed. In the worst case, complete fault simulation has to be performed in every iteration; however, in practice, relatively small numbers of faults need to be simulated based on the discussion above.

5. Experimental results

We applied Procedure 2 using $N_{SAME} = 1000$ to test sequences produced by the test generation procedure *HITEC* [9], and to test sequences produced by the test generation procedure *STRATEGATE* [10]. We compare the results to the results produced by the procedures from [1] and [2]. We consider circuits for which the comparison is possible, and in addition, we consider *s*35932

under the *STRATEGATE* sequence. Other procedures [3]-[8] are based on the ideas in [1] and [2], and do not produce better levels of compaction overall. The results are shown in Tables 1 and 2.

Table 1: Compaction of *HITEC* sequences

circuit	orig	seq.count	omit	restore
s298	259	100	87	153
s344	108	54	53	56
s400	2069	280	405	278
s420	166	122	124	125
s641	211	81	96	119
s820	968	350	424	708
s1238	478	270	247	269
s1488	1192	416	607	646
s5378	900	185	363	458
total	6351	1858	2406	2812

Table 2: Compaction of *STRATEGATE* sequences

circuit	orig	seq.count	omit	restore
s298	194	76	109	117
s344	86	39	45	57
s382	1486	574	490	516
s400	2424	486	896	611
s444	1945	607	546	608
s526	2642	778	1305	1006
s641	166	70	78	101
s820	590	314	361	491
s1196	574	235	226	238
s1423	3943	577	651	1024
s1488	593	355	445	455
s5378	11481	458	806	646
total	26124	4569	5958	5870
s35932	257	127	-	150

In Table 1, after the circuit name, we show the original sequence length produced by *HITEC*, followed by the sequence length obtained after applying the proposed sequence counting procedure. For comparison, we show the test length obtained after applying the omission procedure from [1], and after applying the restoration procedure from [2] to the original sequences produced by *HITEC*. In row *total* we show the sum of the test lengths for all the circuits in the corresponding column.

In Table 2, after the circuit name, we show the original sequence length produced by *STRATEGATE*, followed by the sequence length obtained after applying the proposed procedure. For comparison, we show the test length obtained after applying the omission procedure from [1], and after applying the restoration procedure from [2]. In row *total* we show the sum of the test lengths for all the circuits in the corresponding column excluding *s*35932 to which the omission based procedure was not applied.

From Tables 1 and 2 it can be seen that the sequence counting procedure results in lower test lengths for most of the circuits considered. In addition, the total test length obtained by the counting procedure for all the circuits considered is lower by over 20% than the total test length obtained by the omission procedure, and lower than the total test length obtained by the restoration procedure.

In its current implementation, Procedure 2 performs large numbers of fault simulations to achieve the reported reductions in test length. Methods to reduce the number of simulations while retaining the compaction levels will be investigated. However, the current implementation was sufficient for demonstrating the levels of compaction that can be achieved by allowing test vectors to be reintroduced into the test sequence.

6. Concluding remarks

We presented an approach to improve the levels of compaction achieved for synchronous sequential circuits by static test compaction procedures based on vector omission. Under the proposed approach, referred to as sequence counting, test vectors omitted from the test sequence can be reintroduced at a later time. Reintroducing of vectors helps reduce the compacted test sequence length beyond the length that can be achieved if vectors are omitted permanently. The basic step of the proposed procedure consisted of replacing a vector t_i at time unit k by a vector t_j where $j > i$, and adjusting the rest of the sequence accordingly. Experimental results comparing the proposed approach with the previously proposed vector omission and vector restoration approaches showed that higher levels of compaction can be achieved by the proposed approach. This is to a large extent related to the ability to reintroduce vectors that have already been omitted.

References

[1] 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.

[2] 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.

[3] M. S. Hsiao, E. M. Rudnick and J. H. Patel, "Fast Algorithms for Static Compaction of Sequential Circuit Test Vectors", in Proc. VLSI Test Symp., April 1997, pp. 188-195.

[4] M. S. Hsiao and S. T. Chakradhar, "State Relaxation Based Subsequence Removal for Fast Static Compaction in Sequential Circuits", in Proc. Conf. on Design Autom. and Test in Europe, Feb. 1998, pp. 577-582.

[5] 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.

[6] 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.

[7] S. K. Bommu, S. T. Chakradhar and K. B. Doreswamy, "Static Compaction Using Overlapped Restoration and Segment Pruning", in Proc. Intl. Conf. on Computer-Aided Design, Nov. 1998, pp. 140-146.

[8] R. Guo, I. Pomeranz and S. M. Reddy, "On Speeding-Up Vector Restoration Based Static Compaction of Test Sequences for Sequential Circuits", in Proc. 7th Asian Test Symp., Nov. 1998, pp. 467-471.

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

[10] M. S. Hsiao, E. M. Rudnick, and J. H. Patel, "Sequential Circuit Test Generation Using Dynamic State Traversal", 1996 Europ. Design & Test Conf., March 1996, pp. 22-28.