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

Irit 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.

In this paper, 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 \( T = (t_1, t_2, \ldots, t_n) \), the compacted sequence obtained by either one of the three approaches has the form \( T_c = (t_{i_1}, t_{i_2}, \ldots, t_{i_M}) \), where \( t_{i_1} < t_{i_2} < \cdots < t_{i_M} \). 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 \( T_c = (t_1, t_2, t_3, t_4, \ldots) \) 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 \( T_c = (t_1, t_3, t_4, t_5, \ldots) \) (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 \( T = (t_1, t_2, \ldots, t_{10}) \) 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 \( t_2, t_3, \ldots, t_{10} \) at time unit 1. In general, at time unit \( i \), the sequence may include \( t_{i_1}, t_{i_2}, \ldots, t_{10} \) 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 \( T_c = (t_2, t_3, t_4, t_7, t_{10}) \). In general, the edges defining a compacted sequence can go from \( t_i \) to \( t_{i_M} \) 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-
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 \( T_c = (t_2, t_3, t_5, t_7, t_8, t_{10}) \), and it is shorter by one vector than the previous sequence we obtained. Note that we reintroduced into the compacted sequence the vector \( t_4 \) 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, \( T = (t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}) \), \( T_{c1} = (t_2, t_3, t_4, t_7, t_8, t_{10}) \), and \( T_{c2} = (t_2, t_3, t_7, t_8, t_{10}) \). 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 \( T_{c1} \) we obtain \( I_{c1} = (2, 3, 4, 7, 8, 10) \); and from \( T_{c2} \) we obtain \( I_{c2} = (2, 5, 7, 8, 10) \). 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 \( T_c = (t_{i_1}, t_{i_2}, \ldots, t_{i_{n-1}}) \) be obtained. If \( i_1 > 1 \), then \( t_{i_1}, \ldots, t_{i_{n-1}} \) 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 $t_i$ is replaced by a vector $t_j$ such that $j < i$. 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 $T = (t_1, t_2, \ldots, t_L)$. The test sequence on which the sequence counting step is performed is an arbitrary (compacted) sequence $T_c = (t_1, t_2, \ldots, t_{L_c})$. In Procedure 1, we randomly select a time unit $k$, where $1 \leq k \leq M$. We also select a number $\Delta$. We replace $t_i$ by $t_{i+\Delta}$ and then update the sequence $T_c$ as follows.

If $i + \Delta > L$, we terminate $T_c$ at time unit $k - 1$. For example, suppose that $T = (t_1, \ldots, t_{10})$, $T_c = (t_1, t_2, t_3, t_4, t_5, t_6)$, $k = 5$ and $\Delta = 2$. Note that $t_5 = t_3$, $t_6 = t_4$, and so on. In this case, $i_k = i_9$, and we replace $t_9$ by $t_{10}$. Since $t_{11}$ does not exist in $T$, we terminate $T_c$ at time unit $5$ and obtain $T_c = (t_1, t_2, t_3, t_4)$.

If $i_k + \Delta \leq L$, we find the first time unit $k'$ such that $i_{k'} > i_k + \Delta$. We copy the part of $T$ from time unit $k'$ to time unit $M$ such that it immediately follows time unit $k$. For example, suppose that $T = (t_1, \ldots, t_{10})$, $T_c = (t_1, t_2, t_3, t_4, t_5, t_6)$, $k = 2$ and $\Delta = 4$. In this case, $i_2 = i_9 = 3$, and we replace $t_2$ by $t_{10} = t_3$. The first time unit $k'$ where $i_{k'} > 7$ is $k' = 4$ with $i_4 = 8$. We copy the subsequence $(t_8, t_9, t_{10})$ starting at time unit $4$ and set $k = 4$. We obtain the sequence $T_c = (t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8)$. As another example, consider the same sequences with $k = 5$ and $\Delta = 1$. In this case, $i_5 = i_9 = 9$, and we replace $t_9$ by $t_{10}$. There is no time unit $k'$ with $i_{k'} = 9$, therefore, we do not copy any part of $T$. The resulting sequence is $T_c = (t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8)$.

The value of $\Delta$ by which we increment $i$ 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 $T = (t_1, t_2, \ldots, t_L)$ be the original test sequence. Set $T_c = T$. Set $n_{same} = 0$.
2. Simulate $T_c$ under the faults in $F_D$. Set $T_{c, prev} = T_c$.
3. Call Procedure 1 with $T$ and $T_c$ as input.
4. Simulate $T_c$ under the faults in $F_D$.
5. Set $T_{c, prev} = T_c$.
6. If $F_{D,c} \neq F_D$, set $T_c = T_{c, prev}$ and $n_{same} = n_{same} + 1$.
7. Else, if the length of $T_c$ is smaller than the length of $T_{c, prev}$, set $n_{same} = 0$.
8. Else, set $n_{same} = n_{same} + 1$.
9. If $n_{same} \leq N_{SAME}$, 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 $s35932$.
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](image)

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

![Table 2: Compaction of STRATEGATE sequences](image)

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

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 s35932 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


