7th IEEE Asian Test Symposium, 1998, pp. 452-457

Partitioning and Reordering Techniques for Static Test Sequence Compaction of Sequential Circuits

Michael S. Hsiao[†] *and Srimat T. Chakradhar*[††]
[†]Department of Electrical and Computer Engineering, Rutgers University, Piscataway, NJ
[††]Computer & Communications Research Lab. NEC USA, Princeton, NJ

Abstract

We propose a new static test set compaction method based on a careful examination of attributes of fault coverage curves. Our method is based on two key ideas: (1) fault-list and test-set partitioning, and (2) vector re-ordering. Typically, the first few vectors of a test set detect a large number of faults. The remaining vectors usually constitute a large fraction of the test set, but these vectors are included to detect relatively few hard faults. We show that significant compaction can be achieved by partitioning faults into hard and easy faults. This significantly reduces the computational cost for static test set compaction without affecting quality of compaction. The second technique re-orders vectors in a test set by moving sequences that detect hard faults to the beginning of the test set. Fault simulation of the newly concatenated re-ordered test set results in the omission of several vectors so that the compact test set is smaller than the original test set. Experiments on several ISCAS 89 sequential benchmark circuits and large production circuits show that our compaction procedure yields significant test set reductions in low execution times.

I Introduction

Since cost of testing is directly proportional to the number of test vectors in the test set, short test sequences are desirable. Reduction in test set size can be achieved using static or dynamic test set compaction algorithms. Dynamic techniques [11-14] perform compaction concurrently with the test generation process and often require modification of the test generator. Static compaction techniques, on the other hand, are employed after the test generation process. Thus, static techniques are independent of the test generation algorithm and do not require modifications to the test generator. In addition, static compaction techniques can further reduce the size of test sets obtained after dynamic compaction.

Several static compaction approaches for sequential circuits have been proposed [1-10]. Some of these approaches [1, 2] cannot reduce test sets produced by random or simulation-based test generators. Static compaction techniques based on vector insertion, omission, or selection have been investigated [3]. These techniques require multiple fault simulation passes. The fault simulator is invoked whenever a vector is omitted or swapped to make sure that the fault coverage is not affected. Vector restoration techniques [6, 7] aim to restore sufficient vectors necessary to detect all faults. Fast static test set compaction based on removing recurrence subsequences that start and end on the same or similar states has been reported recently [4, 5]. Though fast, these test sets are not as compact as those achieved by algorithms that use multiple fault simulation passes.

Figure 1 shows two typical fault coverage curves. The curve with a small dip is associated with test sets that are composed of random vectors followed by vectors generated using automatic test pattern generators (ATPG's), while the

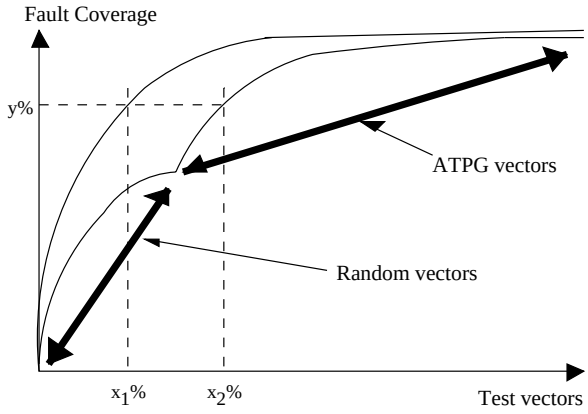

Figure 1: Typical Fault Coverage Curves.

curve without the dip is associated with test sets composed of solely ATPG vectors. In either case, the fault coverage increases rapidly for the first few vectors and eventually levels off. During the flat region, a large number of vectors are required to detect very few additional faults. We formalize this observation using two parameters x and y. The first x% of the vectors detect y% of the faults. For example, it is possible that the first 10% of the vectors in the test set detect 90% of the faults. We also observed that faults detected during the quick rise of the fault coverage curve are also usually detected by vectors generated during the flat region of the curve. Our empirical observations lead to two questions:

1. Since the majority of the test set ($(100 - x)$% vectors) is used to detect a few hard faults ($(100 - y)$% detected faults), can we reduce the execution time by compacting the test set with respect to only the hard faults?
2. If we re-order the test set by placing vectors comprising the last w% of the test set to be at the beginning of the test set, how much of the y% easily detectable faults will still be detected by the re-ordered w% vectors?

*This research was conducted while M. Hsiao was at NEC USA

The first question motivates us to consider fault-list and test-set *partitioning* for static compaction. We attempt to compact the test set by only considering the hard faults. This substantially reduces the cost of fault simulation because only a few faults have to be considered during multiple fault simulation passes. Also, computationally expensive static compaction techniques that have been proposed in the past can now be re-examined, since test-set and fault-list partitioning can greatly reduce the fault simulation cost.

The second question is in regard to vector re-ordering. Re-ordering of test vectors for sequential circuits must be done carefully because detection of a fault in a sequential circuit requires a specific sequence of vectors. Vector re-ordering is effective if vectors that detect hard faults also detect other faults. Both coarse and fine-grain re-ordering are explored.

The contribution of this work is two fold. First, the computational cost for static test set compaction is substantially reduced by careful fault-list and test-set partitioning. Second, re-ordering of vectors is shown to be very effective in reducing the test set size. Significant compactions have been obtained very quickly for large ISCAS89 sequential benchmark circuits, several synthesized circuits, and production circuits.

The remainder of this paper is organized as follows. Section II describes the main ideas behind partitioning and vector re-ordering. Section III explains the compaction algorithm. Section IV presents experimental results. Finally, Section V concludes the paper.

II Main Ideas

Given a test set T, a subsequence of the test set is represented as $T[v_i, v_{i+1}, ..., v_j]$, where v_i and v_j are the i^{th} and j^{th} vectors in the test set T, respectively. The set of faults detected by a subsequence $T[v_i, \ldots, v_j]$ is denoted as $F_{det}[v_i, \ldots, v_j]$.

Let us consider a test set T with n vectors. Assume that this test set T detects a total of f faults. If a static compaction algorithm requires m fault simulations, then the worst case time required for multiple fault simulation passes is proportional to $m \times n \times f$ single-vector logic simulations. Since n is the original test set size and is a fixed number, one can only reduce the cost of compaction by reducing m, f, or both.

A Test-set and fault-list partitioning

The compaction process using partitioning is illustrated in Figure 2. We begin by splitting the test set T into two subsequences $T[v_1, \ldots, v_i]$ and $T[v_{i+1}, \ldots, v_n]$. Let r be the ratio of total number of detected faults f to the number of detected faults by the second partition $T[v_{i+1}, \ldots, v_n]$: $r = \frac{f}{F_{det}[v_{i+1}, \ldots, v_n]}$. If we compact the test set with respect to only $F_{det}[v_{i+1}, \ldots, v_n]$ (Step 1 of Figure 2), the computational cost can be reduced to $m \times n \times (f/r)$. If r is large, significant savings in computational time can be achieved. For example, if 90% of the detected faults are detected quickly in the first test set partition (conversely, only 10% of detected faults in the second partition), then the time required for multiple fault simulation passes can be reduced by an order of magnitude.

After Step 1, it is possible that the compacted test set $T_{compact}$ may not detect all faults, since only a subset of faults were considered during compaction. A possible solution is to

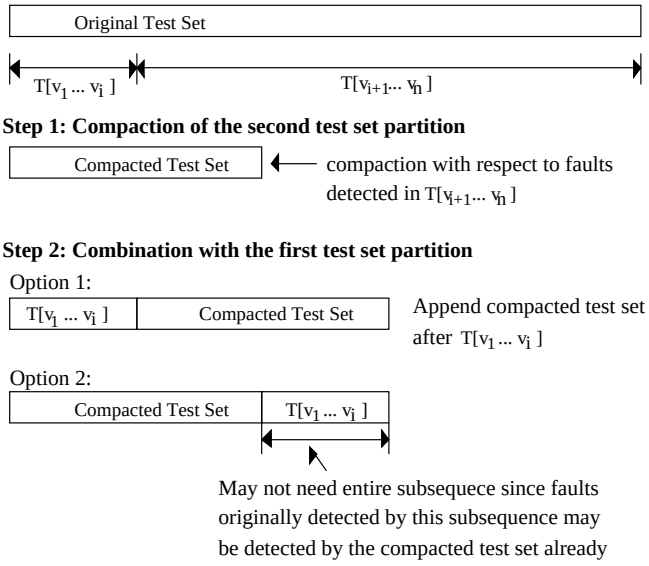

Figure 2: Test Set Compaction Using Partitioning.

combine $T_{compact}$ and the first subsequence $T[v_1, \ldots, v_i]$ (Step 2 of Figure 2). This ensures that all f faults are detected.

We perform static compaction with respect to only a fraction of the faults. Therefore, the computational cost would be less than a method that considers all faults. However, one would expect less compact test sets since only a subset of faults are considered for compaction. Nevertheless, our experiments show that **both** computational cost and the quality of compaction are benefited by partitioning.

B Re-ordering of vectors

Another valid question that stems from the shape of the fault coverage curve illustrated in Figure 1 is whether sequences that detect hard faults can also detect many other, easier faults. In other words, if the sequence of vectors that detect hard faults is copied to the beginning of the test set, can some vectors in the modified test set be omitted?

Again, consider a test set $T[v_1, ..., v_n]$ that detects f faults. If we create a new test sequence by copying the subsequence $T[v_k, \ldots, v_n]$, $1 \leq k \leq n$, to the beginning of the original test set T (see Figure 3), then all f faults are still detectable by the modified test sequence, $T_{new}[v_k, \ldots, v_n, v_1, \ldots, v_n]$, since the original test set is a subset of the modified test sequence. There are now $(n - k + 1) + n$ vectors in the modified test set. Clearly, at least $n - k + 1$ vectors can be omitted from the modified test set. However, it is possible that more than

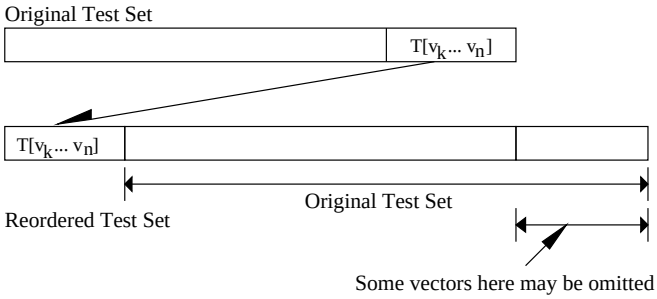

Figure 3: A Reordered Test Set.

$n - k + 1$ vectors can be dropped by omitting vectors at the end of the modified test set. For example, consider two faults f_y and f_z that are detected by the original test set $T[v_1, \ldots, v_n]$. Let us also assume that faults f_y and f_z are detected after vectors v_m and v_n, respectively. Here, v_n is the last vector in the test set and $m < n$. Suppose there exists a k, $m < k < n$, such that subsequence $T[v_k, ..., v_n]$ detects only one fault f_z, then re-ordering $T[v_k, ..., v_n]$ vectors to the beginning of the test set yields the modified test set $T[v_k, ..., v_n, v_1, ..., v_n]$. Figure 4 illustrates this scenario. For this example, the subsequence $T[v_{m+1}, ..., v_n]$ at the end of the modified test set now becomes unnecessary. This subsequence detects only fault f_z and the fault is already detected by the newly re-ordered vectors. The modified test set is $T[v_k, ..., v_n, v_1, ..., v_m]$, with $(n - k + 1) + m$ vectors. Since $k > m$, the new test set size can be less than the original test size n. For example, if $k = m + 3$, then the compacted test set is two vectors smaller than the original test set.

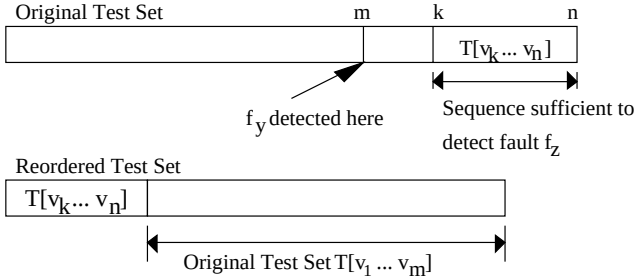

Figure 4: A Shorter Reordered Test Set.

Computing the exact k for the last detected fault f_z may be computationally expensive. Instead, we can simply pick an *arbitrary* k and re-order the subsequence $T[v_k, ..., v_n]$ to the beginning of the test set. Fault simulation of the modified test sequence will determine whether some vectors can be removed. This process can be repeated until every vector in the original test set has been re-ordered. The size of the subsequence being re-ordered plays a significant role in determining the amount of possible compaction. If the re-ordered subsequence consists of 5% of vectors in the test set, then it would take at most 20 passes to finish re-ordering of all the vectors in the original test set. On the other hand, if the re-ordered subsequence consists of only 1% of the vectors, then up to 100 passes of fault simulation may be required. More compaction is achievable with smaller re-order sizes, but at a higher cost of fault simulation. Note that if a large number of vectors are omitted during the first few passes, the total number of passes required can be less than the maximum.

III Test Set Compaction Algorithm

Test-set partitioning involves splitting of the test set into two subsequences $T[v_1, \ldots v_i]$ and $T[v_{i+1}, \ldots v_n]$. These subsequences imply a fault-list partition. Only faults detected by the second subsequence are considered for compaction. The specific value of i $(1 \leq i \leq n)$ has a significant impact on the execution time and quality of the resulting compacted test set. This value can be determined in several ways:

1. Choose a value for i such that the subsequence $T[v_{i+1}, \ldots v_n]$ has a pre-determined number of vectors, or
2. Choose a value of i such that the subsequence $T[v_1, \ldots v_i]$ detects a pre-determined number of faults, or
3. Choose a value for i based on a pre-determined number of vectors and faults.

The value of i can also be chosen using more elaborate methods. There are advantages and disadvantages of each choice. If we split the test set based on a pre-determined number of vectors, then we run into the risk of fewer detected faults by the first subsequence. This can result in less savings in computation costs. On the other hand, if we partition by including a sufficient number of vectors in the first subsequence so that a pre-determined percentage of faults are detected, then the first subsequence can have too many vectors, resulting in a less compact test set. Finding the optimal value of i may be as difficult as the compaction problem itself. In our present work, we have chosen the value of i based on a *pre-determined percentage of faults* that have to be detected by the first subsequence.

The size of subsequences considered for re-ordering has a significant impact on the execution time and compaction quality, as explained in the previous section. In general, coarse-grain (more vectors) re-ordering may be better for large test set sizes, since fine-grain re-ordering can require a large number of fault simulation passes, and the computing resources required can be prohibitive. However, fine-grain re-ordering can lead to good compaction. Therefore, we develop a hybrid approach. We first reduce the test set quickly using coarse-grain (subsequence of 5% test size) re-ordering. Then, we switch to fine-grain (subsequence of 1% test size) re-ordering to further reduce the test set. This two-step re-ordering has proven to be effective for many circuits.

The algorithm for vector-re-ordering with partitioning is shown in Figure 5. The algorithm first picks a partitioning point. Next, coarse and fine-grain re-ordering is performed

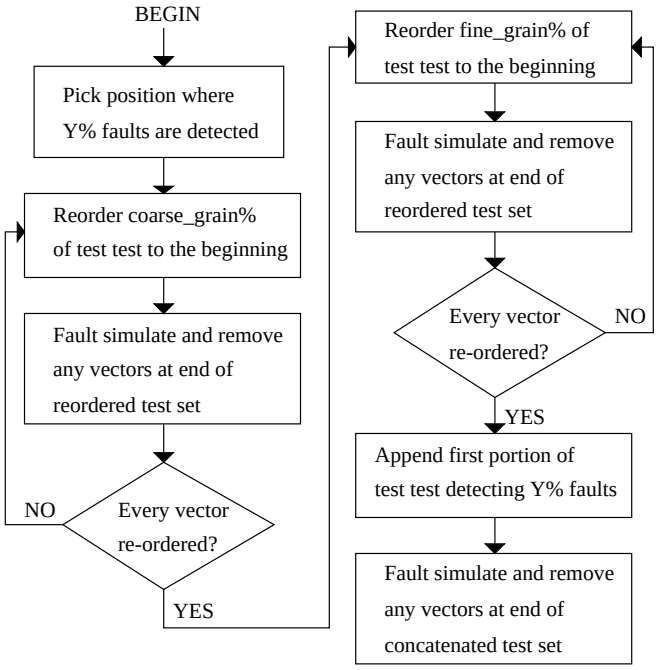

Figure 5: Test Set Compaction Algorithm.

with respect to only (100-Y)% of the partitioned faults. When the re-ordering is complete, the first partition of the vectors is appended. Fault simulation is again applied to remove any non-contributing vectors from the first partition. If the user wishes no partitioning, then we simply skip the partitioning and the concatenation steps in the algorithm, and set Y equal to 0%.

IV Experimental Results

The proposed static test set compaction algorithm was implemented in C and the program repetitively invoked a commercial fault simulator via system calls. HITEC [15] and STRATEGATE [20] test sets generated for ISCAS89 sequential benchmark circuits [17] and several synthesized circuits [19] to evaluate the effectiveness of the algorithms. HITEC is a state-of-the-art deterministic test generator while STRATEGATE is a genetic-algorithms-based test generator that generates test sets with very high fault coverages. Results for several production circuits are also reported. All experiments were performed on a Sun UltraSPARC with 256 MB RAM. Because our algorithms are computationally inexpensive, our execution times are still very reasonable even with the overhead of system calls (overhead involved reading in the circuit, fault list, set-up of data structures, etc.)

The compaction results are shown in Tables 1 and 2 for HITEC and STRATEGATE vectors, respectively. Both tables show the number of faults, the number of vectors in the original test set, and the number of faults detected by the test set. For each test set, we generate compact test sets using two methods. One method compacts the test set by only considering re-ordering of vectors. Results for this experiment are shown in column *No-partition*. The number of vectors in the compact test set are shown in column *Vec*, the percentage reduction in test vectors as compared to the original test set is shown in column *% R*, the number of faults detected by the compact test set is shown in column *Det* and the CPU seconds required is shown in column *Time*. The second method uses both partitioning and re-ordering. Results for this experiment are shown in column *Partition*.

In our experiments, the partitioning technique splits the test set such that the first subsequence detected 80% of the detected faults. Therefore, compaction of the entire test set is done with respect to only 20% of the faults.

Vector-re-ordering is based on a two-step process. First, we consider subsequences that include 5% of the test set (coarse-grain re-ordering), followed by the the second step that re-orders by sizes of 1% (fine-grain re-ordering) of the test set.

For most circuits, significant reductions in test set sizes were achieved by vector re-ordering with or without partitioning. On average, 35.1% and 41.1% reductions were obtained for HITEC vectors with and without partitioning, respectively, with a maximum test set reduction of 72.2% for circuit s35932. Similarly, averages of 46.4% and 48.9% reductions were achieved for STRATEGATE vectors with and without partitioning, with maximum test set reduction of 88.4% for s5378. Compaction quality is slightly better without partitioning, but at a much higher computation cost. The two tables show that execution times are significantly lower with partitioning. For smaller circuits, partitioning reduces the execution time by about 50%. For the larger circuits, the execution time is reduced by a factor up to 4.32. Fault coverages for the compacted test sets are always greater than or equal to the original fault coverages. For example, in circuit s35932, the compacted HITEC test set detected more faults. Since STRATEGATE vectors already provide high fault coverages, no additional faults were detected after compaction.

Ideally, by considering only 20% faults during compaction, we can expect a 5-fold speed up. The fixed overheads from fault-free logic simulation makes ideal speedups difficult to obtain. For instance, if fault-free simulation constitutes 40% of total simulation cost, then the best speed-up we can achieve is $\frac{1}{(20\% \times 60\%)+40\%} = 1.9$. Because greater fractions of the fault-free simulation are needed for smaller circuits, speedups are smaller for them. On average, partitioning accelerated the compaction process by 2.90 times for HITEC vectors and by 3.14 times for STRATEGATE vectors across all circuits.

The size of compacted test sets derived using partitioning is slightly larger for most circuits when compared to those derived without partitioning. However, the differences are often insignificant. There are cases where marginally smaller compacted test sets are achieved by the partitioning case. This happens when the first subsequence in the test set partition is small.

Close examination of HITEC and STRATEGATE test sets reveals that a *smaller* percentage of STRATEGATE vectors are necessary to detect 80% of detected faults. Therefore, more compaction were obtained for STRATEGATE vectors. For HITEC test sets, one can always partition at a lower fault coverage (e.g., at 60% or 70%) to reduce the number of vectors in the first partition, but this can increase the execution times during compaction. There are always exceptions. For instance, in the HITEC test set for s444, compaction with partitioning achieves a significantly more compact test set in much shorter execution time. This is because the first 80% of the detected faults are also detected by vectors in the second partition that detects the remaining 20% faults.

We also applied our static compaction method to a few partial-scanned large production circuits. The number of non-scanned flip-flops ranged from 137 to 995. These circuits have several non-Boolean primitives, such as tristate buffers, bidirectional buffers and buses. In addition, they have set/reset flip-flops and *multiple clocks*. Original test sets for these circuits were derived using a commercial test generator. Compaction results are shown in Table 3. For these circuits, it was not possible to run experiments without partitioning due to prohibitively long run times. Thus, compaction with partitioning had to be used to reduce the execution times. Significant reductions in test set sizes have been achieved as compared to the original test set. Furthermore, fault coverages obtained by the compacted test sets were often higher.

When comparing our approach with existing static test set compaction algorithms, the amount of reduction in original test set size, the resulting fault coverages, and execution times are all of interest. Execution times are difficult to compare since different platforms were used (HP 9000 J200 was used in [4], HP C180 (64-bit processor) for [6, 7], UltraSPARC in ours, and [3] did not report execution times). Furthermore,

Table 1: Compaction results for HITEC test sets

Ckt	Total Faults	Original		No-Partition				Partition			
		Vec	Det	Vec	% R	Det	Time	Vec	% R	Det	Time
s298	308	322	265	**184**	**42.8**	265	105 s	**184**	**42.8**	265	74.5 s
s344	342	127	328	**51**	**59.8**	328	41.4 s	84	33.9	328	28.6 s
s382	399	2074	312	**704**	**66.6**	325	240 s	764	63.2	312	91.5 s
s400	426	2214	352	**878**	**60.3**	352	324 s	967	56.3	352	117 s
s444	474	2240	389	826	63.1	389	360 s	**516**	**77.0**	389	118 s
s526	555	2258	361	**1328**	**41.2**	361	528 s	**1328**	**41.2**	361	202 s
s641	467	209	404	**114**	**45.5**	404	92.8 s	120	42.6	404	65.0 s
s713	581	173	476	**110**	**36.4**	476	92.3 s	129	25.4	476	79.6 s
s820	850	1115	813	882	20.9	813	383 s	**810**	**27.4**	813	185 s
s832	870	1137	817	**768**	**32.5**	817	400 s	988	13.1	817	202 s
s1196	1242	435	1239	**336**	**22.8**	1239	276 s	370	14.9	1239	110 s
s1238	1355	475	1283	**338**	**28.8**	1283	295 s	415	12.6	1283	129 s
s1423	1515	150	750	**134**	**10.7**	750	270 s	150	0	750	72.1 s
s1488	1486	1170	1444	**622**	**46.8**	1444	600 s	798	31.8	1444	304 s
s1494	1506	1245	1453	**622**	**50.0**	1453	547 s	862	30.8	1453	286 s
s5378	4603	912	3238	712	21.9	3238	1723 s	**662**	**27.4**	3238	825 s
s35932	39094	496	34901	**138**	**72.2**	34901	45455 s	**138**	**72.2**	35002	15330 s
am2910	2391	2023	2189	**1569**	**22.4**	2189	3819 s	1588	21.5	2189	884 s
div16	2147	238	1679	**154**	**35.3**	1679	247 s	162	31.9	1679	142 s
Avg					**41.1**		2937 s		35.1		1013 s

Det: Number detected faults **Vec**: Test set length **Time**: Execution time **% R**: Percent reduction from original test set
Smallest test set sizes/greatest compactions highlighted in **bold**

we use multiple system calls to a complex commercial fault simulator which handles multiple clocks, complex gate types, etc. In terms of reductions in test set sizes among the various static test set compaction techniques, vector-omission based compaction [3] generally outperforms other compaction approaches at high computation costs. However, with fault-list partitioning described in this paper, the vector-omission technique may become more feasible for large circuits or large test sets. Compaction based on recurrence subsequence removal is very fast, but it produces less compact test sets. Fault-list and test-set partitioning can decrease the execution time further. Finally, both vector-restoration [6, 7] and our proposed techniques are not constrained by lack of recurrence subsequences and can produce compact test sets at less expensive costs than [3]. One significant feature about our technique is that the test-set and fault-list partitioning strategy can be applied to any of the previously proposed vector-omission [3] and vector-restoration [6, 7] methods to further reduce execution times. This is an important feature that distinguishes our technique from the previously proposed methods.

V Conclusions

We have proposed a new static test set compaction framework using fault-list/test-set partitioning and vector-re-ordering. Significant reductions in test set sizes have been obtained using our techniques. Furthermore, the partitioning technique rapidly accelerates the compaction process and can easily be used to accelerate existing static compaction algorithms without compromising on the quality of compaction. Compaction algorithms based on extensive fault simulations can particularly benefit from the partitioning technique. Our experiments show that the proposed compaction technique is viable for large circuits with large test sets.

References

[1] T. M. Niermann, R. K. Roy, J. H. Patel, and J. A. Abraham, "Test compaction for sequential circuits," *IEEE Trans. Computer-Aided Design,* vol. 11, no. 2, pp. 260-267, Feb. 1992.

[2] B. So, "Time-efficient automatic test pattern generation system," Ph.D. Thesis, EE Dept., Univ. of Wisconsin at Madison, 1994.

[3] I. Pomeranz and S. M. Reddy, "On static compaction of test sequences for synchronous sequential circuits," *Proc. Design Automation Conf.,* pp. 215-220, June 1996.

[4] M. S. Hsiao, E. M. Rudnick, and J. H. Patel, "Fast algorithms for static compaction of sequential circuit test vectors," *Proc. IEEE VLSI Test Symp.,* pp. 188-195, Apr. 1997.

[5] M. S. Hsiao and S. T. Chakradhar, "State relaxation based subsequence removal for fast static compaction in sequential circuits," *Proc. Design, Automation, and Test in Europse (DATE) Conf.,* pp. 577-582, Feb. 1998.

[6] I. Pomeranz and S. M. Reddy, "Vector restoration based static compaction of test sequences for synchronous sequential circuits," *Proc. International Conference on Computer Design*, pp. 360-365, Oct. 1997.

[7] R. Guo, I. Pomeranz, and S. M. Reddy, "Procedures for static compaction of test sequences for synchronous sequential circuits based on vector restoration," *Proc. Design, Automation, and Test in Europse (DATE) Conf.,* pp. 583-587, Feb. 1998.

Table 2: Compaction results for STRATEGATE test sets

Ckt	Total Faults	Original		No-Partition				Partition			
		Vec	Det	Vec	% R	Det	Time	Vec	% R	Det	Time
s298	308	194	265	**132**	**32.0**	265	74.3 s	**132**	**32.0**	265	63.3 s
s344	342	86	329	**48**	**44.2**	329	39.3 s	58	32.6	329	24.4 s
s382	399	1486	364	**601**	**59.6**	364	241 s	**601**	**59.6**	364	92.8 s
s400	426	2424	383	**1033**	**57.4**	383	330 s	**1033**	**57.4**	383	133 s
s444	474	1945	424	**716**	**63.2**	424	307 s	**716**	**63.2**	424	133 s
s526	555	2642	454	**1631**	**38.3**	454	470 s	**1631**	**38.3**	454	281 s
s641	467	166	404	**136**	**18.1**	404	122 s	139	16.3	404	77.4 s
s713	581	176	476	122	30.7	476	109 s	**121**	**31.3**	476	77.5 s
s820	850	590	814	**525**	**11.0**	814	370 s	559	5.3	814	140 s
s832	870	701	818	**564**	**19.5**	818	357 s	607	13.4	818	160 s
s1196	1242	574	1239	**277**	**51.7**	1239	243 s	329	42.7	1239	121 s
s1238	1355	625	1282	**324**	**48.2**	1282	273 s	339	45.8	1282	136 s
s1423	1515	3943	1414	1273	67.7	1414	1950 s	**1239**	**68.6**	1414	541 s
s1488	1486	593	1444	**481**	**18.9**	1444	688 s	523	11.8	1444	254 s
s1494	1506	540	1453	**472**	**12.6**	1453	736 s	494	8.5	1453	258 s
s5378	4603	11481	3639	1347	88.3	3639	3185 s	**1340**	**88.4**	3639	2337 s
s35932	39094	257	35100	**133**	**48.2**	35100	35486 s	139	45.9	35100	9359 s
am2910	2391	2509	2198	**556**	**77.8**	2198	1091 s	635	74.7	2198	396 s
mult16	1708	1696	1665	201	88.1	1665	371 s	**184**	**89.2**	1665	167 s
div16	2147	1098	1815	504	54.1	1815	1073 s	**483**	**56.0**	1815	372 s
Avg					**48.9**		2501 s		46.4		796 s

Det: Number detected faults **Vec**: Test set length **Time**: Execution time **% R**: Percent reduction from original test set
Smallest test set sizes/greatest compactions highlighted in **bold**

Table 3: Results for production circuits

Ckt	Gates	FFs	Total Faults	Original		Partition		
				Vec	Det	Vec	% R	Det
circuit1	7428	137	8411	2228	6780	1532	31.2	6781
circuit2	6387	130	6976	3190	5992	2059	35.5	5997
circuit3	11784	487	18701	3963	17182	3559	10.2	17287
circuit4	24784	995	36280	3807	34456	2411	36.7	34457

Gates: Number of gates **FFs**: Number of flip-flops **Det**: Number of faults detected
Vec: Test set length **% R**: Percent reduction from original test set size

[8] S.K. Bommu, S.T. Chakradhar, and K.B. Doreswamy, "Static test sequence compaction based on segment reordering and fast vector restoration," *Proc. Intl. Test Conf.*, 1998.

[9] S.K. Bommu, S.T. Chakradhar, and K.B. Doreswamy, "Static compaction using overlapped restoration and segment pruning," *Proc. Intl. Conf. CAD*, 1998.

[10] S.K. Bommu, S.T. Chakradhar, and K.B. Doreswamy, "Vector restoration using accelerated validation and refinement," *Proc. Asian Test Symp.*, 1998.

[11] A. Raghunathan and S. T. Chakradhar, "Acceleration techniques for dynamic vector compaction," *Proc. Intl. Conf. Computer-Aided Design,* pp. 310-317, 1995.

[12] S. T. Chakradhar and A. Raghunathan, "Bottleneck removal algorithm for dynamic compaction and test cycles reduction", *Proc. European Design Automation Conf.,* pp. 98-104, September 1995.

[13] S. T. Chakradhar and A. Raghunathan, "Bottleneck removal algorithm for dynamic compaction in sequential circuits," *IEEE Trans. on Computer-Aided Design,* vol. 16, no. 10, pp. 1157-1172, Oct., 1997.

[14] E. M. Rudnick and Janak H. Patel "Simulation-based techniques for dynamic test sequence compaction," *Proc. Intl. Conf. Computer-Aided Design,* pp. 67-73, 1996.

[15] T. M. Niermann and J. H. Patel, "HITEC: A test generation package for sequential circuits," *Proc. European Conf. Design Automation (EDAC)*, pp. 214-218, 1991.

[16] T. M. Niermann and J. H. Patel, "Method for automatically generating test vectors for digital integrated circuits," *U.S. Patent No.* 5,377,197, December 1994.

[17] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," *Int. Symposium on Circuits and Systems*, pp. 1929-1934, 1989.

[18] T. H. Cormen, C. E. Leiserson, R. L. Rivest, *Introduction to Algorithms*. Cambridge, MA: The MIT Press, 1990.

[19] M. S. Hsiao, E. M. Rudnick, and J. H. Patel, "Automatic test generation using genetically-engineered distinguishing sequences," *Proc. VLSI Test Symp.,* pp. 216-223, 1996.

[20] M. S. Hsiao, E. M. Rudnick, and J. H. Patel, "Sequential circuit test generation using dynamic state traversal," *Proc. European Design and Test Conf.,* pp. 22-28, 1997.