Asian Test Symposium, 2000, pp. 159-164

Compaction-Based Test Generation Using State and Fault Information

Ashish Giani [*], Shuo Sheng [*], Michael Hsiao [*], Vishwani D. Agrawal [**]

[*] Department of Electrical and Computer Engineering, Rutgers University, Piscataway, NJ 08854.
{ashishg, shuo, mhsiao} @ece.rutgers.edu

[**] Bell Labs, Lucent Technologies, Murray Hill, NJ 07974
va@research.bell-labs.com

Abstract

We present a new test generation procedure for sequential circuits using newly traversed state and newly detected fault information obtained between successive iterations of vector compaction. Two types of techniques are considered. One is based on which new states a sequential circuit is driven into, and the other is based on the new faults that are detected in the circuit between consecutive iterations of vector compaction. These data modify an otherwise random selection of vectors, to bias vector sequences that cause the circuit to reach new states, and cause previously undetected faults to be detected. The biased vectors, when used to extend the compacted test set, provide an intelligent selection of vectors. The extended test set is then compacted. Repeated applications of state and fault analysis, vector generation and compaction produce significantly high fault coverage using relatively small computing resources. We obtained improvements in terms of higher fault coverage, fewer vectors for the same coverage, or smaller number of iterations and time required, consistently for several benchmark circuits.

1. Introduction

Random test generation, proposed in [3], used a pseudo-random pattern generator to approximate the behavior of random patterns. As explained in [1], randomly generated patterns are useful for certain types of circuits, while they do not produce good fault coverage in other circuits. Further, they generally require a large test set size [2]. Certain lines in the circuit cannot be set to a specific logic value using purely random vectors, making faults on those lines hard to detect. These faults are called random pattern resistant faults. Weighted random patterns have been found to yield higher fault coverages in circuits that contain such types of hard to detect faults. Much work has been done on weighted random pattern generation [4,5]. In these approaches, the probability of getting a 0 or a 1 at a particular input is biased towards detecting random resistant faults. With weighted random patterns, the difficulty that arises is that no one set of weights may be suitable for all faults even in a combinational circuit. In the case of sequential circuits, the faults may need a biased internal state besides a biased input vector. So, it is more difficult to obtain a good set of weights at the primary inputs using only structural techniques. Static compaction procedures extract a set of "necessary" vectors from a given test set. Vector restoration techniques [6,7] aim to restore sufficient vectors necessary to detect all faults. Various methods to extend the compacted test set are given in [8,9], including randomly picking a vector from the compacted test set and holding it a given number of times, perturbing a few bits in a randomly chosen vector and holding it a given number of times, and copying over a sequence of vectors to generate new vectors. In these techniques, vectors are chosen randomly to extend the test set. In [10], the authors present a technique to generate tests for sequential circuits using weighted random vectors and a static compaction procedure while in [11], the correlation among test vectors was studied and this information was used to generate test vectors.

In our work, we would like to bias some "useful" vectors by making sure that they have a higher probability of being chosen. Thus, rather than having a purely uniform distribution of vectors, we introduce some non-uniformity into the process. Useful vectors are obtained using two broad categories. These are: New-State vectors that drive the sequential circuit into new states during successive iterations, and New-Fault vectors that obtain new faults during successive iterations. We show that with New State vectors, we are able to visit a greater number of useful (including previously unvisited) states in each iteration. This leads to the detection of more hard-to-detect faults in each iteration. So, we want to utilize and learn from the vectors that lead to the detection of these new states. We use this information obtained by analyzing the compacted set of vectors to **intelligently** choose useful vectors such that fewer vectors can achieve the same, or higher fault coverage in a smaller amount of time.

2. Overview of the Test Generation Procedure

Figure 1 presents an overall picture of the Test Generation Procedure using vector compaction. The algorithm is given below:

```
-Initialize iteration number $i$ to 0
-Generate $L$ random vectors to form a random input
  sequence $S_0$
```

```
while (the terminating condition is not satisfied)
  -Fault simulate Si on the circuit under test
  -Apply static compaction on Si to obtain a
    compacted test sequence Sic
  -Extend the compacted sequence by generating
    vectors to form a suffix Sisu
  -Increment i for the next iteration
```

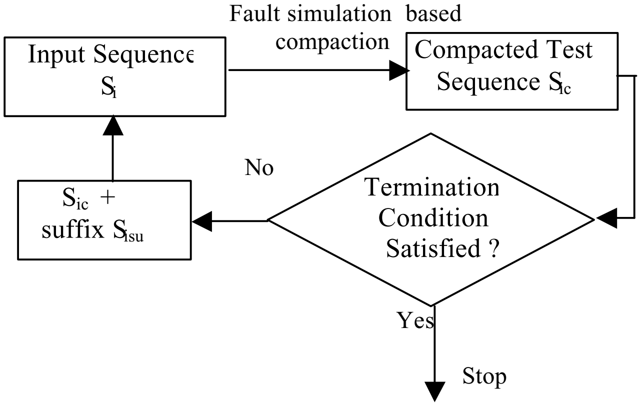

Figure 1: Overview of Test Generation

There can be three possible terminating conditions. These are: desired fault coverage is obtained, preset time for which the experiments were conducted is reached, and number of iterations of test set extension and compaction is reached. We chose the number of iterations as the terminating condition.

This paper focuses on methods of extending the compacted test set (i.e. obtaining suffix S_{isu}) by using New-State Prefix Vectors, New-State Postfix Vectors, New-Fault Vectors and Threshold-Fault Vectors.

3. Definitions

1) **New-State Prefix Vectors:** A sequence of "m" vectors that lead to a newly visited state between consecutive iterations of compaction and fault simulation is called ***New-State Prefix (NSPR) Vectors***. They are obtained as follows: Identify the different states that a circuit is driven into during two successive iterations. In iteration i, the states that were reached in iteration i-1 are compared with those that have been previously reached. NSPR vectors are the sequence of "m" vectors that lead to the new state. "m" is chosen to be the minimum circuit depth, which is the minimum number of Flip Flops on any path from a primary input to a primary output. This information is recorded in the State Matrix.

2) **New-State Postfix Vectors:** A sequence of "n" vectors that follow the vector that drives the circuit into a new state in consecutive iterations is called ***New-State Postfix (NSPO) Vectors***. "n" is chosen to be 5. This information is recorded in the State Matrix. Both NSPR and NSPO vectors are shown in Figure 2.

3) **New-Fault Vectors:** A sequence of "m" vectors that lead to the detection of a new fault between consecutive iterations of compaction and fault simulation is called ***New-Fault (N-F) Vectors***. They are obtained as follows: Identify the different faults that are detected during two successive iterations. In iteration i, the faults that were detected in iteration i-1 are compared with those that have been detected previously. N-F vectors are the sequence of "m" vectors that lead to the detection of the new fault. As before, "m" is chosen to be the minimum circuit depth. This information is recorded in the Fault Matrix.

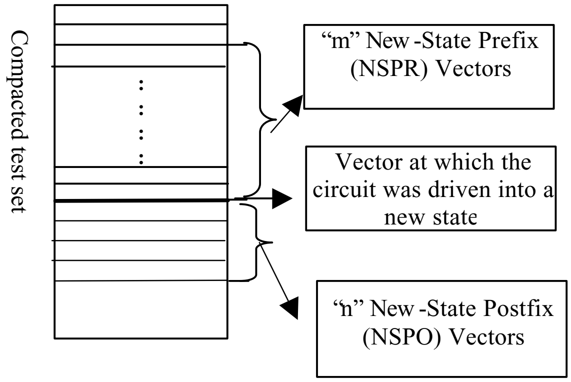
Compacted test set
"m" New-State Prefix (NSPR) Vectors
Vector at which the circuit was driven into a new state
"n" New-State Postfix (NSPO) Vectors

Figure 2: New-State Prefix and New-State Postfix Vectors

4) **Threshold-Fault Vectors:** Vectors detecting faults equal to or greater than the threshold fault number are identified, and a sequence of "m" vectors prior to each of these vectors is called ***Threshold-Fault (T-F) Vectors***. They are obtained as follows: Identify the number of faults detected by each vector. Set a threshold number of faults. The sequence of "m-1" vectors that occur prior to the vector that detects more faults than the threshold value, and the vector under consideration form the T-F vectors. As before, "m" is chosen to be the minimum sequential depth. This information is recorded in the Fault Number Matrix. Both N-F and T-F vectors are shown in Figure 3.

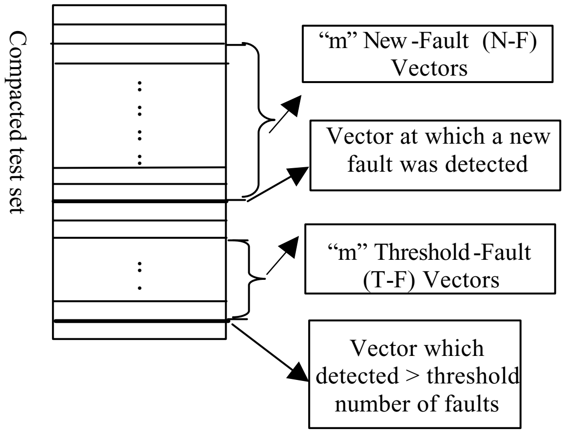
Compacted test set
"m" New-Fault (N-F) Vectors
Vector at which a new fault was detected
"m" Threshold-Fault (T-F) Vectors
Vector which detected > threshold number of faults

Figure 3: New-Fault and Threshold-Fault Vectors

4. Test Generation Approach

Since the NSPR vectors lead to the detection of new states between consecutive iterations, they help span the state space of the circuit. Let us consider a circuit for which we have just visited a new state as depicted in Figure 4; exploring neighboring states of this newly visited state may help us in detecting faults that require one of these new states as an intermediate state to excite the fault, or propagate it to a primary output. As shown, there is a sequence of vectors (V_{seq}) that takes us from a relatively well explored state subspace, to a state that has not been previously visited. There may exist other sequences that take us back from this newly visited state to the previously explored subspace. Since the newly visited state detected a hard-to-detect fault, V_{seq} is a valuable sequence. We postulate that the state subspace around this state might be useful in detecting other previously undetected hard-to-detect faults. To do so, whenever some sequence takes us back to the previously explored state space, we would need to get back to the new unexplored subspace. However, it is not easy to do so, otherwise this region would have already been explored earlier. We know that the vector sequence V_{seq} had led us to the previously unvisited state earlier and can do so again. So, we bias it with a higher priority by choosing to bias the NSPR and NSPO as important vectors.

Similarly, the N-F vectors lead to the detection of new faults in consecutive iterations. This proves that they are useful in detecting hard-to-detect faults, because they are vectors that detect faults that could not be detected in earlier iterations. The T-F vectors that detect a large number of faults have already demonstrated their usefulness, so we use them to extend the compacted test set. We believe that these vectors, when compacted, might lead to the detection of more faults in subsequent iterations.

Algorithms for our two techniques are described below:

Technique 1: Using NSPR and NSPO Vectors.

```
While (number of vectors less than cut-off)
    -Randomly pick a vector from the compacted
     test set, perturb a few bits in it and hold it a
     random number of times.
    -Copy over a sequence of five vectors from
     the compacted test set to generate new vectors.
    -Randomly pick an NSPR/NSPO vector
     from the State Matrix and copy over a
     sequence of "m" vectors prior to this vector
     and "n" vectors after this vector.
    -From the State Matrix information
     generated above, randomly pick an
     NSPR/NSPO vector, perturb a few bits in it
     and hold it a random number of times.
```

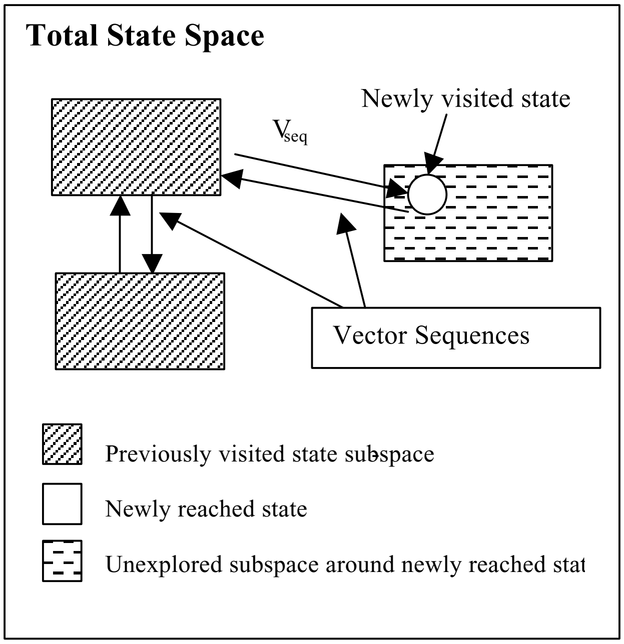

Figure 4: Reasons for biasing NSPR and NSPO vectors

Technique 2: Using N-F and T-F Vectors.

```
While (number of vectors less than cut-off)
    -Randomly pick a vector from the compacted
     test set, perturb a few bits in it and hold it a
     random number of times.
    -Copy over a sequence of five vectors from
     the compacted test set to generate new vectors.
    -From the Fault Matrix, randomly pick an
     N-F vector, perturb a few bits in it and hold
     it a random number of times.
    -Randomly pick an N-F vector from the
     Fault Matrix, and copy over a sequence of
     "m" vectors prior to this vector.
    -Randomly pick a T-F vector from the Fault
     Number Matrix, perturb a few bits in it and
     hold it a random number of times.
    -Randomly pick a T-F vector from the Fault
     Number Matrix, and copy over a sequence
     of "m" vectors prior to this vector.
```

We choose the cut-off value, the number of bits perturbed in a vector, and the number of times a vector is held in a similar manner as [8]. The cut-off value is initially set to 2,000 vectors. If two consecutive sequences do not detect any new faults, the cut-off is increased to the next higher value. Possible values that can be assigned to cut-off are 5,000, 10,000, 20,000 and 40,000. The cut-off is never increased beyond 40,000 because this would result in an excessive time for fault simulation while the compaction process occurs. Zero, one, two or three bits can be perturbed with equal probability in a vector. A vector can be held for 2^i time units, with a probability of $2^{(i+1)}$ for $0 \leq i \leq 7$, and 2^{-8} for $i = 8$.

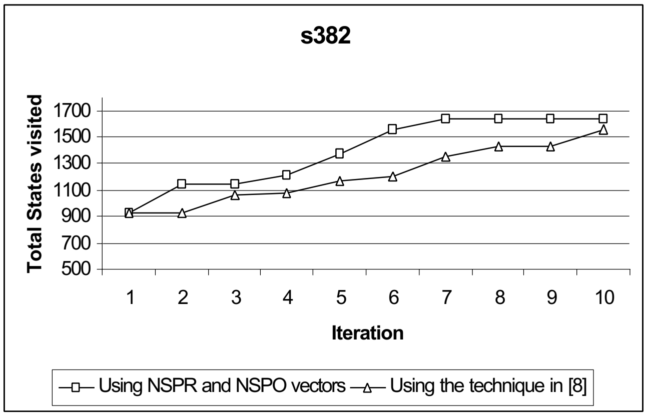

Figure 5: Total States visited for s382

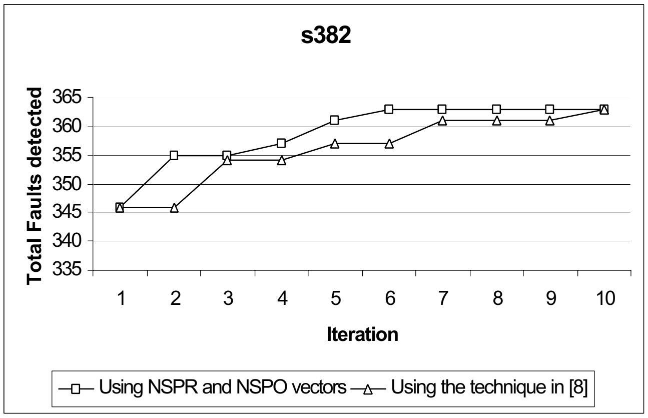

Figure 6: Total Faults detected for s382

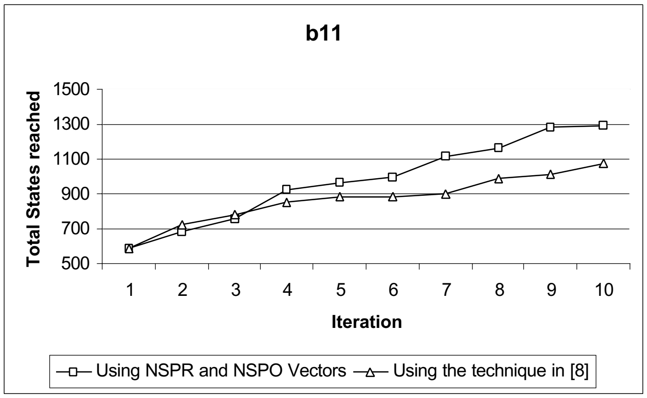

Figure 7: Total States visited for b11

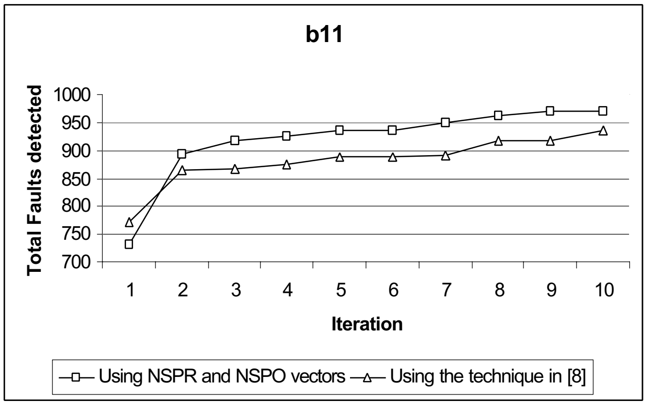

Figure 8: Total Faults detected for b11

5. Results

All experiments were conducted on an Ultra SPARC 10 with 256 MB of RAM for some ISCAS 89 and ITC 99 benchmark circuits.

The fault simulator used in our experiments targets all the faults in the circuit in each iteration. Other previously reported techniques target 128 [8] and 256 [9] randomly selected faults in the initial iterations, and gradually increase the sample size until all the undetected faults are targeted in later iterations. This leads toa disparity in the execution time taken for fault simulation during compaction. To remove this disparity and obtain a suitable platform for comparison, we implemented the algorithm suggested by [8] and used our fault simulator and compactor. As mentionedearlier, we fixed an upper bound of 125 iterations as the terminating condition.

Before going into the detailed results of our experiments, Figures 5, 6, 7 and 8 show the key idea of how the useful new states obtained from biasing NSPR and NSPO vectors can help in test generation. Figure 5 gives the total number of states visited in s382 during the first ten iterations for our technique using NSPR and NSPO vectors, versus the number of states reached for our implementation of the technique in [8]. Figure 6 shows the total number of faults detected in the s382 in the corresponding iterations for both techniques. For example, by iteration 2, our NSPR/NSPO technique has visited 1147 states, while the original implementation [8] visited 926 states. These 221 additional states helped to detect 9 additional hard faults in that iteration, as indicated in Figure 6. Consistently, our technique visits a greater number of new states and detects 363 faults by the sixth iteration, while our implementation of the technique in [8] required 10 iterations to reach the same number of faults. Thus, we are able to detect faults faster.

Likewise, Figure 7 gives the total number of states visited in the b11 during the first ten iterations for our technique using NSPR and NSPO vectors, versus the number of states reached for our implementation of the technique in [8]. Figure 8 shows the total number of faults detected in the b11 in the corresponding iterations for both techniques. As seen in these two figures, our technique visits a greater number of new states and detects more faults in a lower number of iterations when compared to our implementation of the technique in [8].

Going into detailed results, the results for the technique in [9] are shown in Table 1, while the results for our implementation of the technique in [8] are shown in Table 2. In Table 1, column 1 gives the circuit name, while columns 2, 3 and 4 give the number of faults detected, test set size, and the time taken to detect the indicated number of faults respectively. For example, for s382, 364 faults were detected with a test set size of 572. The time required was 0.46 minutes. In Table 2, columns 1 to 4 represent the same parameters as Table 1. Column 5 gives the number of iterations required to achieve the fault coverage. For example, for s400, 382 faults were detected with a test set size of 617 vectors. 6 minutes and 9 iterations were required.

The results for our techniques are shown in Table 3. Column 1 gives the circuit name, while columns 2, 3, 4 and 5 give the number of faults detected, test set size, the time taken and number of iterations required to detect the indicated number of faults respectively for the technique in [8], columns 6, 7, 8 and 9 give these parameters for the technique using NSPR and NSPO vectors to extend the compacted test set, while columns 10, 11, 12 and 13 give these parameters for the technique using N-F and T-F vectors to extend the compacted test set.

Comparing the results obtained in Tables 1 and 3, we can see that for most circuits, our techniques require a smaller test set size to achieve the same high fault coverage. For example, in the technique using NSPR and NSPO vectors, the number of vectors required is about 13% less for the s526, about 16% less for the s1488, about 9% less for the s1494, and about 5% less for the b11 when compared to the method used in [9]. Comparing columns 3 and 7 of Table 3, the number of vectors required by this technique is about 34% less for the b01and about 57% less for the b04. We are also able to detect two more faults in the s400 and 393 more faults in the b21, and obtain 51 more faults and use about 35% less vectors for the b12 in comparison with the method used in [9].

In the technique using N-F and T-F vectors to extend the compacted test set, the number of vectors required is about 13% less for the s1494 when compared to the method used in [9]. Comparing columns 3 and 11 of Table 3, the number of vectors required by this technique is about 31% less for the b04, and about 25% less for the b08. We are also able to detect one more fault in the s400 and 180 additional faults using 751 less vectors for b21 in comparison with the method used in [9].

A possible explanation of why the technique in [9] requires fewer vectors than our techniques for certain circuits is that we used a Linear Reverse Order Restoration Algorithm mentioned in [8], but restored two vectors at a time, while the technique in [9] uses a Radix Reverse Order Restoration Algorithm.

Comparing the iterations taken for our implementation of the technique in [8] (Table 3, column 5), with the iterations required for our NSPR and NSPO technique (Table 3, column 9), we observe that the number of iterations required to achieve the fault coverage is about 38% less for the s382, about 68% less for the s526, about 59% less for the s1423 and about 80% less for the s1488 and s1494. Similarly, we obtain major improvements in the number of iterations required using N-F and T-F vectors (Table 3, column 13). Thus, we not only obtain savings in the test set size, but also need fewer iterations and less time to obtain the same high fault coverage.

Table 1: Results reported in [9]

PREVIOUS METHOD [9]			
Circuit	**Det.**	**Vec.**	**Time (min.)**
s382	364	572	0.46
s400	382	677	0.65
s526	454	1557	2.64
s713	476	104	0.13
s1196	1239	224	0.4
s1238	1283	235	0.44
s1423	1416	1049	8.84
s1488	1444	426	1.93
s1494	1453	454	2.2
s5378	3643	672	35.55
b11	1004	419	1.59
b12	1470	3697	27.49
b21	18023	12651	135.72

Det. = Faults Detected
Vec. = # Vectors in Test Set

Table 2: Implementation of the original algorithm in [8] without fault samples

	PREVIOUS METHOD [8]			
Circuit	**Det.**	**Vec.**	**Time (min.)**	**Iter.**
s382	364	641	26	21
s400	382	617	6	9
s526	454	1854	51	47
s713	476	205	4	10
s1196	1239	241	13	47
s1238	1283	334	18	46
s1423	1416	1387	171	87
s1488	1444	549	59	79
s1494	1453	508	89	73
s5378	3643	672	840	117
b01	133	91	0.5	5
b04	1168	416	14	18
b08	463	422	9	19
b11	1004	496	22	33
b12	1470	3697	292	66
b21	18023	14946	6910	51

Det. = Faults Detected Vec. = # Vectors in Test Se
Iter. = # Iterations required

Table 3: Results for: A) Algorithm in [8] without fault samples, B) NSPR and NSPO Vectors and C) N-F and T-F Vectors

	A) [8] without fault samples				B) Using NSPR and NSPO vectors				C) Using N-F and T-F Vectors			
Circuit	Det.	Vec.	Time (min.)	Iter.	Det.	Vec.	Time (min.)	Iter.	Det.	Vec.	Time (min.)	Iter.
s382	364	641	26	21	364	897	12	13	364	764	4	7
s400	382	617	6	9	384	807	7	7	383	766	3	9
s526	454	1854	51	47	454	1349	21	15	454	1548	18	14
s713	476	205	4	10	476	85	0.5	4	476	79	1	4
s1196	1239	241	13	47	1239	229	15	48	1239	245	7	28
s1238	1283	334	18	46	1283	245	13	43	1283	290	6	25
s1423	1416	1387	171	87	1416	1209	60	36	1416	1249	47	31
s1488	1444	549	59	79	1444	360	15	16	1444	430	9	16
s1494	1453	508	89	73	1453	417	6	15	1453	393	8	16
s5378	3643	672	840	117	3643	1412	155	28	3643	889	148	32
b01	133	91	0.5	5	133	60	0.1	5	133	82	0.2	3
b04	1168	416	14	18	1168	180	3	10	1168	289	5	15
b08	463	422	9	19	463	390	2	11	463	316	3	14
b11	1004	496	22	33	1004	399	20	28	1004	461	10	18
b12	1470	3697	292	66	1521	2418	93	31	1516	2400	55	25
b21	18023	14946	6910	51	18416	16139	3662	22	18203	11900	3651	35

Det. = Faults Detected Vec.= # Vectors in Test Set. Iter. = # Iterations required

6. Conclusions

We have proposed a new and efficient technique for extending the compacted test set, using information based on new states reached during succesive iterations, and new faults detected between consecutive iterations. Our results show that, as in previous methods, high fault coverage was achieved. In some cases, we detected more faults than have been previously reported. A significant improvement in fault coverage for the b12 and b21 was observed using both the NSPR/NSPO technique and the N-F/T-F technique. At the same time, the number of vectors required to achieve the high level of fault coverage was fewer than other techniques required, while the number of iterations needed, and the time taken was also lower.

7. References

[1] V. D. Agrawal, “When to Use Random Testing”, IEEE Trans. On Computers, vol C-27, No. 11, pp. 1054-1055, November 1978.

[2] M. Abramovici, M. A. Breuer and A. D. Friedman, "Digital System Testing and Testable Design", New York, NY: Computer Science Press, 1990.

[3] M. A. Breuer, "A Random and an Algorithmic Technique for Fault Detection Test Generation for Sequential Circuits", IEEE Trans. On Computers, vol C-20, No. 11, pp. 1364-1370, November 1971.

[4] M. F. Alshaibi and C. R. Kime, "Fixed-biased pseudo-random built in self test for random pattern resistant circuits", in Proc. Intl. Test Conf., Oct. 1994, pp. 929-938.

[5] F. Muradali, T. Nishada, and T. Shimizu, "Structure and technique for pseudo random-based testing of sequential circuits", in Journal of Electronic Testing: Theory and Applications (JETTA), Feb. 1995, pp. 6:107-115.

[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 Europe (DATE) Conf., pp. 583-587, Feb. 1998.

[8] R. Guo, I. Pomeranz, and S. M. Reddy, “A fault simulation based test pattern generator for synchronous sequential circuits”, 17th IEEE VLSI Test Symposium (VTS), pp. 260-267, April 1999.

[9] R. Guo, S. M. Reddy and I. Pomeranz, “PROPTEST: A Property Based Test Pattern Generator for Sequential Circuits Using Test Compaction”, Proc. Design Automation Conference (DAC), pp. 653-659, June 1999.

[10] A. Jain, V.D. Agrawal, M. S. Hsiao, "On generating tests for sequential circuits using static compaction", ITSW 1999, March 1999.

[11] A. Giani, S. Sheng, M. S. Hsiao, and V. D. Agrawal, "Correlation-Based Test Generation for Sequential Circuits", Proc. 9th IEEE North Atlantic Test Workshop (NATW), pp. 76-83, May 2000.