JOURNAL OF ELECTRONIC TESTING: Theory and Applications 18, 63–72, 2002

State and Fault Information for Compaction-Based Test Generation

ASHISH GIANI*
Intel Corporation, Hillsboro, OR 97124, USA
asgiani@ichips.intel.com

SHUO SHENG AND MICHAEL S. HSIAO†
Department of ECE, Rutgers University, Piscataway, NJ 08854, USA
shuo@ece.rutgers.edu
mhsiao@ece.rutgers.edu

VISHWANI D. AGRAWAL
Agere Systems, Murray Hill, NJ 07974, USA
va@agere.com

Received May 16, 2001; Revised July 21, 2001

Editor: Chauchin Su

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 the new states a sequential circuit is driven into, and the other is based on the new faults that are detected 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.

Keywords: ATPG, sequential circuits, test compaction

1. Introduction

Numerous implementations of simulation-based test generators have been described in the literature [4]. Despite the use of a variety of inputs, namely, functional, random and weighted-random, the success of such programs is highly circuit-dependent. Weighted random patterns often yield better fault coverages in circuits that contain *hard-to-detect* or *random-pattern resistant faults* [2, 15]. This is because the probability of a 0 or 1 at a particular input can be biased for detecting such faults. However, no one set of weights may be suitable for all faults [1]. Besides, there is no simple way to weight the probabilities of the state variables. Another problem is the long vector sequences produced

*Formerly with Department of ECE, Rutgers University, Piscataway, NJ 08854, USA.

†Supported in part by an NSF Career Award, under contract CCR-0093042 and New Jersey Commission on Science and Technology.

by simulation-based test generators, which makes vector compaction desirable. A vector compacter is also a simulation-based tool and offers its own independent potential for test generation.

Recently, static vector restoration type of compaction [10, 16] has been used to aid test generation. A specific feature of such compaction is that the resulting compacted test set guarantees to retain the original fault coverage. Various test generation methods based on compaction have been proposed [6–9, 11–13, 17] in which static compaction of modified test sets is iteratively performed. During an iteration, the test set is first augmented by appending new vectors and then static compaction removes any unwanted vectors. This process is repeated until a satisfactory coverage or a maximum number of iterations has been reached. In one approach [11, 12], new vectors are appended to the test set by 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 appended to the test set by randomly choosing from the compacted test set obtained in the previous iteration. In another approach [7, 13, 17], spatial and temporal correlation among vectors is used to append new vectors to the test set. For example, the Hadamard transform of compacted vectors can provide the embedded spectral information, which is then used to create new vectors [8]. A recent paper [9] demonstrates the use of the Hadamard transform coefficients to devise self-test for a system-on-a-chip that has an embedded processor available for vector generation.

In the present work, we identify *useful* vectors and assign them higher probabilities of being chosen. Thus, rather than having a purely uniform distribution of vectors, we introduce some non-uniformity into the process. Useful vectors belong to two broad categories: *new-state vectors* that drive the sequential circuit into new states during successive iterations, and *new-fault vectors* that detect new faults during successive iterations. With new state vectors, we are able to visit a greater number of useful (including previously unreached) states in each iteration, which leads to the detection of more hard-to-detect faults.

We also developed a technique to speed up the test generation process. Instead of using fault sampling during compaction to reduce the execution time, previously detected faults are periodically removed from the target fault list with the corresponding compacted vector sequence saved. In other words, compaction is performed using only the remaining faults. This significantly reduces the work during each iteration of compaction.

The remainder of the paper is organized as follows. Section 2 gives an overview of the test generation procedure. Section 3 gives some definitions behind our work. Section 4 details the test generation. Section 5 describes a fault-dropping technique to speed up the ATPG process. Section 6 discusses experimental results.

2. Overview of the Test Generation Procedure

Fig. 1 presents the overall framework of the test generation procedure. The algorithm is given below:

Set parameters: Starting sequence length $L \sim 3{,}000$, and Terminating criteria – fault coverage, CPU time, maximum iterations
Initialize iteration count, $i = 0$
Generate a random vector sequence S_0 of length L
While (terminating criteria is not satisfied)
  Simulate circuit under test for fault detection by input S_i
  Obtain s_i by static compaction of S_i
  Construct vector sequence (suffix) SU_i from state and fault information
  Construct vector sequence S_{i+1} by appending SU_i to s_i
  Increment iteration count, $i = i + 1$

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.

Compaction improves the quality of the test set by retaining only a small subset of vectors that is sufficient to retain the fault coverage. This paper focuses on the use of the information available in the compacted test set in order to extend it (i.e., obtain suffix SU_i) by

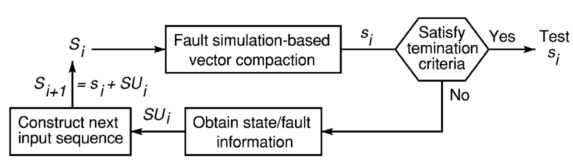

Fig. 1. Test generation framework.

replicating the vectors from the compacted test set that take the circuit to untraversed states and thus help detect new faults. Fault dropping is used as a method of enhancing simulation performance.

3. Definitions

1. *New-State Prefix (NSPR) Vectors.* A sequence of m vectors that lead to a newly visited state between consecutive iterations of compaction is called *new-state prefix (NSPR) vectors*. These vectors are obtained as follows: Identify the different states that the circuit is driven into during two successive iterations. In iteration i, the states that were reached in iteration $i-1$ are compared with those previously reached. NSPR vectors form a sequence of m vectors that leads to the new state. The value of 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 heuristic generally gives good results. A smaller value of m may fail to achieve the new state, while a larger value will make the sequence unnecessarily long. For an iteration, all new states are saved in a *state matrix*, along with pointers to vectors at which the states were reached. The state matrix is reinitialized for each iteration since the state information of past iterations is no longer relevant.
2. *New-State (NSPO) Postfix Vectors.* A sequence of n vectors that follows the vector driving the circuit into a new state in an iteration is called *new-state postfix (NSPO) vectors*. The value of n is heuristically chosen as 5. The NSPR and NSPO vectors are shown in Fig. 2(a).
3. *New-Fault (NF) Vectors.* A sequence of m vectors that leads to the detection of a new fault between consecutive iterations of compaction is called *new-fault (NF) vectors*. These vectors are obtained as follows: Identify the 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. NF vectors is a sequence of m vectors that leads to the detection of the new fault. The value of m is again chosen to be the minimum circuit depth. Each new fault and a pointer to the vector at which it is detected are stored in a *fault matrix*.
4. *Threshold-Fault (TF) Vectors.* Vectors detecting faults whose number equals or exceeds a threshold number are identified and saved in a *fault number matrix*. A sequence of m vectors prior to each of these vectors is called *threshold-fault (TF) vectors*. These vectors 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 just prior to the vector that detects more faults than the threshold value, and the vector under consideration form the TF vectors. As before, m is chosen to be the minimum sequential depth. The NF and TF vectors are shown in Fig. 2(b).

4. Test Generation Approach

Since the NSPR vectors lead the circuit into new states, they help to span the state space of the circuit. Let us consider a circuit for which we have just visited a new state as depicted in Fig. 3; exploring neighboring states

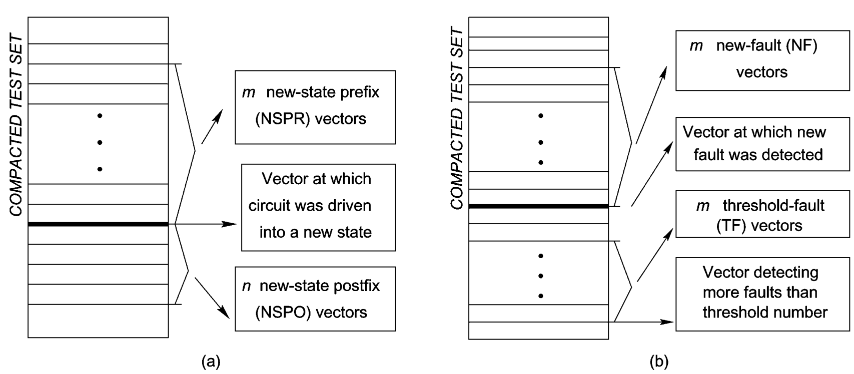

Fig. 2. New-state prefix (NSPR) and postfix (NSPO) vectors, and new-fault (NF) and threshold-fault (TF) vectors: (a) New-state prefix (NSPR) and postfix (NSPO) vectors; (b) New-fault (NF) and threshold-fault (TF) vectors.

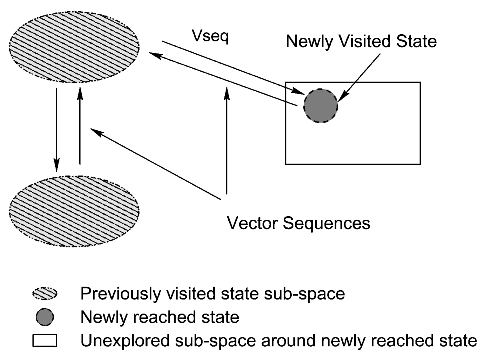

Fig. 3. Reasons for biasing NSPR and NSPO vectors.

of this newly visited state helps in detecting faults that require one of these 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 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, otherwise this region would have already been explored. We know that the vector sequence V_{seq} led the circuit into a previously unreached state and can do so again. So, we give higher priorities to the NSPR and NSPO vectors. In iteration i, we compare the states reached in iteration $i-1$ with earlier ones and the newly reached states of iteration $i-1$ are considered harder to reach as they were not reached in earlier iterations. The vector sequences that take the circuit to new states are the ones we would like to repeat in the hope that we can visit their neighbors too.

Similarly, NF vectors lead to the detection of new faults. This proves that they are useful in detecting hard-to-detect faults. TF 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 these two techniques are given in Figs. 4 and 5.

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 reported in another paper [11]. The cut-off value is initially set to 2,000 vectors. If two consecutive sequences do not detect any

```
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, randomly pick an NSPR/NSPO
        vector, perturb a few bits in it and hold it a random number of times.
```

Fig. 4. 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.
    From the fault matrix, randomly pick an NF vector, perturb a few bits in it and hold
        it a random number of times.
    Randomly pick an NF vector from the fault matrix, and copy over a sequence of m vectors
        prior to this vector.
    Randomly pick a TF vector from the fault number matrix, perturb a few bits in it and
        hold it a random number of times.
    Randomly pick a TF vector from the fault number matrix, and copy over a sequence
        of m vectors prior to this vector.
```

Fig. 5. Technique 2: Using NF and TF vectors.

new fault, 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 during the compaction process. 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 $i = 8$. Zero, one, two or three bits of a vector can be perturbed with equal probability. Perturbing a few bits is similar to mutation in genetic algorithms [14]. While this is a random perturbation, the perturbation probability is kept very low to avoid randomizing the vector. As with genetic algorithms, starting with a different random seed will result in different starting population, but the final results are strikingly similar. Similarly, while choosing *n new-state postfix* (NSPO) vectors, we ran experiments with $n = 5$, 10 and 15. While there were slight differences in the numbers of faults detected per iteration, there was no difference in the final fault coverage reached. We settled for $n = 5$, because this reduced the time taken for fault simulation and compaction.

5. Speeding up Test Generation by Fault Dropping

One way to reduce the computation costs of simulation-based test generation is to reduce the number of faults simulated at any time. When fault sampling is used during compaction [11, 12], a sample of 128 or 256 randomly chosen faults is used. As the sampled faults become detected, more faults are added to the set being simulated, until all faults have been considered. Instead of using fault sampling, we developed a *fault dropping* technique to speed up test generation. In fault dropping, detected faults are periodically removed from the target fault list, reducing the time taken by simulation. Fig. 6 illustrates the difference in the two concepts. In the fault sampling approach, test generation starts with a smaller number of target faults, and gradually add new faults to the target list. This is illustrated by the rising curve. On the contrary, the fault-dropping technique starts with the entire fault list, and removes the detected faults periodically to reduce the target fault list. This is illustrated by the falling curve. When faults are removed, the test sequence that detected those faults must be saved. Finally, these saved sequences are combined and an overall compaction is performed to produce the final test set. In smaller circuits that require only a few iterations to obtain the desired fault coverages, faults are removed every 1 or 2 iterations. For larger circuits, we remove faults less frequently, executing 4 or 5 iterations between removal of faults.

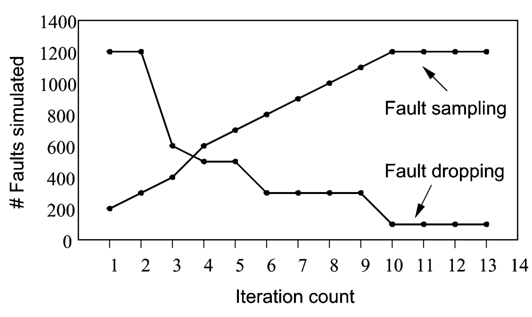

Fig. 6. Fault sampling versus fault dropping.

6. Experimental Results

All experiments were conducted on an Ultra Sparc 10 with 256 MB of RAM for several ISCAS'89 [3] and ITC'99 [5] benchmark circuits. The fault simulator used in our experiments targets all single stuck-at faults of the circuit in each iteration. Previously reported techniques target 128 [11] and 256 [12] randomly selected faults in the initial iterations, and gradually increase the sample size until all undetected faults are targeted in later iterations. This leads to a disparity in the execution time taken for fault simulation during compaction. To remove this disparity and obtain a suitable reference for comparison, we implemented the algorithm suggested by [11] and used our fault simulator and compacter. As mentioned earlier, we fixed an upper bound of 125 iterations as the terminating condition. We do not assume reset state to be available for ISCAS'89 circuits. For ITC'99 circuits, a reset signal is available, but is not explicitly used during test generation.

Figs. 7 and 8 show the key idea of how the useful new states obtained from NSPR and NSPO vectors can help in test generation. Fig. 7(a) gives the total number of states visited in s382 during the first ten iterations for the technique using NSPR and NSPO vectors, versus the number of states reached for our implementation of a previous technique [11]. Fig. 7(b) shows the total number of faults detected in s382 in corresponding iterations for both techniques. For example, by iteration 2, the NSPR/NSPO technique has visited 1147 states, while the other technique [11] visited 926 states. These 221 additional states helped to detect 9 additional hard faults in that iteration, as indicated

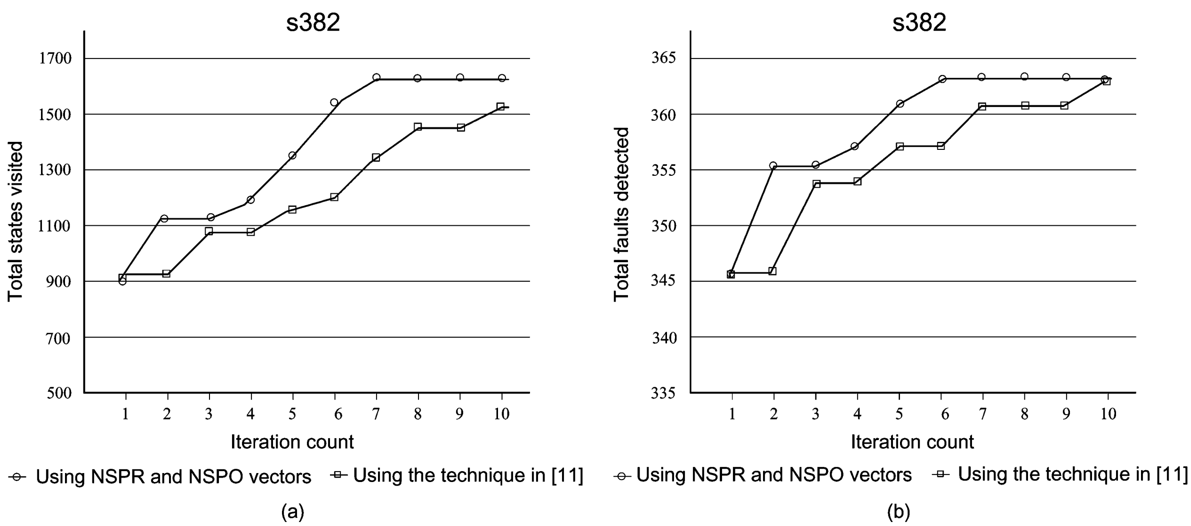

Fig. 7. Test generation for s382: (a) Total states visited for s382; (b) Total faults detected for s382.

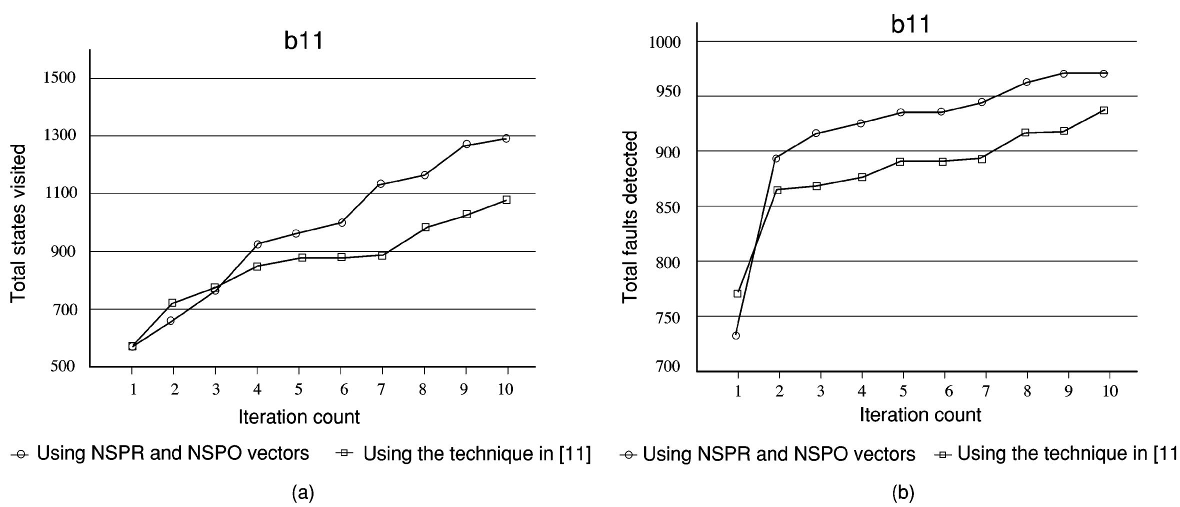

Fig. 8. Test generation for b11. (a) Total states visited for b11; (b) Total faults detected for b11.

in Fig. 7(b). Consistently, our technique visited a greater number of new states and detected 363 faults by the sixth iteration, while the other technique [11] required 10 iterations to detect the same number of faults.

Fig. 8(a) gives the total number of states visited in b11 during the first ten iterations for the technique using NSPR and NSPO vectors, versus the number of states reached for the implemented previous technique [11]. Fig. 8(b) shows the total number of faults detected in b11 by iterations in both techniques. As seen in these two figures, our technique visits a greater number of new states and detects more faults using fewer iterations when compared to our implementation of the previous technique [11].

The results for another published technique, Proptest [12], are shown in Table 1, and the results for our implementation of the previous technique just described [11] are shown in Table 2. In Table 1, column 1 gives the circuit name, and columns 2–4 give the number of faults detected, test set size, and the time taken by an Ultra Sparc 10, respectively. For example,

Table 1. Test generation results reported in [12].

Circuit	Faults detected	Vectors generated	CPU minutes
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

Table 2. Algorithm of [11] implemented without fault sampling.

Circuit	Faults detected	Vectors generated	CPU minutes	Iterations performed
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	1672	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

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 through 4 represent the same parameters as in 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 in six minutes, requiring 9 iterations.

The results for our techniques are shown in Table 3. We have run several experiments on each benchmark circuit with different random seeds (making perturbation/holding different among trials), but the results are similar. Changing the seed does not affect the overall number of detected faults, but it marginally changes the length of the final test set, and can take one or two iterations more, increasing the total time by a small percentage. The reported results are the best ones. 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 of Guo et al. [11], columns 6, 7, 08 and 9 give these parameters for the technique using NSPR and NSPO vectors to extend the compacted test set, while columns 10–13 give these parameters for the technique using NF and TF vectors to extend the compacted test set.

Comparing the results of Tables 1 and 3, we see that for most circuits, our techniques produce smaller test set sizes to achieve the same fault coverage. For example, in the technique using NSPR and NSPO vectors, the number of vectors required is about 13% lower for s526, about 16% lower for s1488, about 9% lower for s1494, and about 5% lower for b11 when compared to Proptest [12]. Comparing columns 3 and 7 of Table 3, the number of vectors required by this technique is about 34% lower for b01 and about 57% lower for b04. We are also able to detect two more faults in s400 and 393 more faults in b21, and cover 51 more faults by about 35% fewer vectors for b12 in comparison to the method of Guo et al. [11].

For the technique using NF and TF vectors to extend the compacted test set, the number of vectors required is about 13% lower for s1494 when compared to Proptest [12]. Comparing columns 3 and 11 of Table 3, the number of vectors required by this technique is about 31% lower for b04, and about 25% lower for b08. We are also able to detect one more fault in s400 and 180 additional faults using 751 fewer vectors for b21 in comparison to the method of Guo et al. [12]. A possible explanation of why Proptest [12] requires fewer vectors than our techniques for some circuits is that we used a *linear reverse order restoration algorithm* [11], but restored two vectors at a time, while Proptest [12] uses a *radix reverse order restoration algorithm*.

Compared to the number of iterations in our implementation of the technique of Guo et al. [11] (Table 3,

Table 3. Test generation with previous and new methods.

	Previous method [11] without fault sampling				New methods: NSPR and NSPO vectors				New methods: NF and TF vectors			
Circuit	Det	Vec	Time	Iter	Det	Vec	Time	Iter	Det	Vec	Time	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 = Number of faults Detected; Vec = Number of vectors generated; Iter = Number of iterations required; Time = CPU minutes for Ultra Sparc 10.

column 5), those for the NSPR and NSPO technique (Table 3, column 9) are about 38% lower for s382, about 68% lower for s526, about 59% lower for s1423 and about 80% lower for s1488 and s1494. Similarly, we obtain major improvements in the number of iterations required using NF and TF vectors (Table 3, column 13). Thus, we not only obtain savings in the test set size, but also use fewer iterations and less time to obtain the same fault coverage.

Table 4 shows the time taken when the technique of fault dropping is applied during fault simulation based compaction, compared to when all faults in the circuit are simulated during every iteration. Column 1 gives the circuit name. Columns 2 and 3, respectively, give the fault coverage and time taken by Proptest [12], which used fault sampling. Columns 4 and 7 give the fault coverage for the techniques using NSPR/NSPO vectors and NF/TF vectors, respectively. Columns 5 and 8 give the time taken for test pattern generation when faults are dropped for the techniques using NSPR/NSPO vectors and NF/TF vectors, respectively. Columns 6 and 9 give the time taken for test pattern generation when all faults are considered as target faults in each iteration for the techniques using NSPR/NSPO vectors and NF/TF vectors, respectively.

We observe that the time taken for test generation generally reduces by a factor of 3 to 4 and sometimes even reduces by a factor of 10. These times are now comparable to the time taken by Proptest [12]. Moreover, for some hard-to-test circuits like s5378 and b12, both NSPR/NSPO vector based test generation and NF/TF vector based test generation take shorter times when compared to Proptest [12]. This highlights the effectiveness of the technique of fault dropping in speeding up the compaction time, since it results in an overall speedup in the test generation time. For circuits like b21, our techniques take more time, but we detect a significant number of additional faults.

7. Conclusion

We have proposed a new and efficient technique for compaction-based test generation where the information about state reachability and fault detectability is

Table 4. Fault coverage and CPU time for test generation with and without fault dropping.

			New methods					
			NSPR and NSPO vectors			NF and TF vectors		
	Proptest with fault sampling [12]			CPU minutes			CPU minutes	
Circuit	Faults detected	CPU min.	Faults detected	Fault drop.	No fault dropping	Faults detected	Fault drop.	No fault dropping
s382	364	0.46	364	2.02	12.0	364	1.53	4.0
s400	382	0.65	384	1.25	7.0	383	0.83	3.0
s526	454	2.64	454	5.75	21.0	454	5.50	18.0
s713	476	0.13	476	0.5	0.50	476	0.5	1.0
s1196	1239	0.4	1239	1.09	15.0	1239	1.25	7.0
s1238	1283	0.44	1283	1.75	13.0	1283	1.12	6.0
s1423	1416	8.84	1416	15.08	60.0	1416	14.59	47.0
s1488	1444	1.93	1444	2.54	15.0	1444	2.46	9.0
s1494	1453	2.20	1453	3.02	6.0	1453	2.88	8.0
s5378	3643	35.55	3643	28.01	155.0	3643	32.81	148.0
b01	–	–	133	0.10	0.1	133	0.05	0.2
b04	–	–	1168	1.03	3.0	1168	0.67	5.0
b08	–	–	463	1.49	2.0	463	1.34	3.0
b11	1004	1.59	1004	2.08	20.0	1004	1.83	10.0
b12	1470	27.49	1521	26.75	93.0	1516	25.58	55.0
b21	18023	135.72	18416	495.21	3662.0	18203	473.03	3651.0

incorporated. The results show that high fault coverages are 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 NF/TF technique. The number of vectors required to achieve a high level of fault coverage was smaller than other techniques. The CPU time was lower as well.

References

1. V.D. Agrawal, "When to Use Random Testing," *IEEE Trans. Computers*, vol. C-27, no. 11, pp. 1054–1055, Nov. 1978.
2. M.F. Alshaibi and C.R. Kime, "Fixed-Biased Pseudo-Random Built In Self Test for Random Pattern Resistant Circuits," in *Proc. International Test Conf.*, 1994, pp. 929–938.
3. F. Brglez, D. Bryan, and K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuits," in *Proc. Int. Symposium on Circuits and Systems (ISCAS)*, May 1989, pp. 1929–1934.
4. M.L. Bushnell and V.D. Agrawal, *Essentials of Electronic Testing for Digital, Memory & Mixed-Signal VLSI Circuits*. Boston: Kluwer Academic Publishers, 2000.
5. S. Davidson et al., "ITC'99 Benchmark Circuits—Preliminary Results," in *Proc. Int. Test Conf.*, 1999, pp. 1125. Also see websit www.cerc.utexas.edu/itc99-benchmarks/bench.html.
6. A. Giani, S. Sheng, M.S. Hsiao, and V.D. Agrawal, "Compaction-Based Test Generation Using State and Fault Information," in *Proc. 9th Asian Test Symp.*, 2000, pp. 159–164.
7. A. Giani, S. Sheng, M.S. Hsiao, and V.D. Agrawal, "Correlation-Based Test Generation for Sequential Circuits," in *Proc. 9th IEEE North Atlantic Test Workshop*, 2000, pp. 76–83.
8. A. Giani, S. Sheng, M.S. Hsiao, and V.D. Agrawal, "Efficient Spectral Techniques for Sequential ATPG," in *Proc. Design, Automation, and Test in Europe (DATE)*, 2001, pp. 204–208.
9. A. Giani, S. Sheng, M.S. Hsiao, and V.D. Agrawal, "Novel Spectral Methods for Built- In Self-Test in a System-on-a-Chip Environment," in *Proc. 19th IEEE VLSI Test Symp.*, 2001, pp. 163–168.
10. 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. Design, Automation, and Test in Europe (DATE)*, 1998, pp. 583–587.
11. R. Guo, I. Pomeranz, and S.M. Reddy, "A Fault Simulation Based Test Pattern Generator for Synchronous Sequential Circuits," in *Proc. 17th IEEE VLSI Test Symp.*, 1999, pp. 260–267.
12. R. Guo, I. Pomeranz, and S.M. Reddy, "PROPTEST: A Property Based Test Pattern Generator for Sequential Circuits Using Test

Compaction," in *Proc. 36th Design Automation Conf.*, 1999, pp. 653–659.
13. A. Jain, V.D. Agrawal, and M.S. Hsiao, "On Generating Tests for Sequential Circuits Using Static Compaction," in *6th IEEE Int. Test Synthesis Workshop*, March 1999.
14. P. Mazumder and E.M. Rudnick, *Genetic Algorithms for VLSI Design, Layout, and Test Automation*. Upper Saddle River, New Jersey: Prentice-Hall, 1999.
15. F. Muradali, T. Nishida, and T. Shimizu, "Structure and Technique for Pseudo Random-Based Testing of Sequential Circuits," *J. Electronic Testing: Theory and Applications*, vol. 6, no. 1, pp. 107–115, Feb. 1995.
16. I. Pomeranz and S.M. Reddy, "Vector Restoration Based Static Compaction of Test Sequences for Synchronous Sequential Circuits," in *Proc. Int. Conf. Computer Design*, 1997, pp. 360–365.
17. S. Sheng, A. Jain, M.S. Hsiao, and V.D. Agrawal, "Correlation Analysis of Compacted Test Vectors and the Use of Correlated Vectors for Test Generation," in *7th IEEE Int. Test Synthesis Workshop*, March 2000.

Ashish Giani received his B.E. degree in Electronics Engineering from the University of Mumbai, India in 1998 and his M.S. degree in Computer Engineering in 2000 from Rutgers, The State University of New Jersey. He is currently working as a Design Engineer at Intel Corporation, Hillsboro, OR.

Shuo Sheng received his B.S. in Automatic Control Engineering with minor in Applied Mathematics from Huazhong University of Science and Technology, Wuhan, P.R. China in 1995. He obtained his M.S. in Pattern Recognition and Intellegent Control from Tsinghua University, Beijing, P.R. China in 1998. Since Sept., 1998, he has been a Ph.D. student at the ECE Department, Rutgers University, Piscataway, NJ. Currently, he is conducting his dissertation at Virginia Tech. His research interest includes the application of Signal Processing methods to Automatic Test Generation and Verification for Sequential Circuits. He is a student member of IEEE computer society and signal processing society since 1998.

Michael S. Hsiao received the B.S. in computer engineering with highest honors from the University of Illinois at Urbana-Champaign in 1992, and M.S. and Ph.D in electrical engineering in 1993 and 1997, respectively, from the same university. During academic year 1992, he was a teaching assistant in the Department of Electrical and Computer Engineering at the University of Illinois. Between 1993 and 1997, he was a research assistant at the Center of Reliable and High-Performance Computing, University of Illinois. During his studies, he was recipient of the Digital Equipment Corp. Fellowship and McDonnell Douglas Scholarship. He was a visiting scientist at NEC USA in Princeton, NJ, during the summer of 1997. Between 1997 and 2001, Dr. Hsiao was an Assistant Professor in the Department of Electrical and Computer Engineering at Rutgers, the State University of New Jersey. Since 2001, he has been an Associate Professor in the Bradley Department of Electrical and Computer Engineering at Virginia Tech. Dr. Hsiao is a recipient of the National Science Foundation CAREER Award. He has published more than 50 refereed journal and conference papers and is serving on the program committee for IEEE International Conference on VLSI Design, IEEE International Test Synthesis Workshop and IEEE North Atlantic Test Workshop. His current research focuses on VLSI testing, design verification, diagnosis, and low power design.

Vishwani D. Agrawal is a Distinguished Member of Technical Staff in the Processor Architectures and Compilers Research Department at Agere Systems, Murray Hill, New Jersey, and a Visiting Professor of Electrical and Computer Engineering at Rutgers University, New Brunswick, New Jersey. He obtained his BSc degree from the University of Allahabad, Allahabad, India, in 1960, BE degree from the University of Roorkee, Roorkee, India, in 1964, ME degree from the Indian Institute of Science, Bangalore, India, in 1966, and PhD degree in electrical engineering from the University if Illinois, Urbana-Champaign, in 1971. His current interests are VLSI testing, synthesis for testability, and parallel algorithms. He has published over 250 papers, has coauthored five books and holds thirteen U.S. patents. His text-book, *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits* (Kluwer Academic Publishers, 2000), co-authored with M.L. Bushnell, has just been published. He is the founder and Editor-in-Chief of the *Journal of Electronic Testing: Theory and Applications* and a past Editor-in-Chief of the *IEEE Design & Test Computers* magazine. He is the Consulting Editor for the *Frontiers in Electronic Testing* book series of Kluwer Academic Publishers, Boston. He is the co-founder of the *International Conference on VLSI Design*, held annually in India. He has served on numerous conference committees and has been a frequently invited speaker. Most recently, he was the keynote speaker at the *Ninth Asian Test Symposium*, held in Taiwan in December 2000. During 1989 and 1990, he served on the Board of Governors of the IEEE Computer Society, and in 1994, chaired the Fellow Selection Committee of that Society. He has received six *Best Paper Awards*. In 1993, he received the *Distinguished Alumnus Award* of the University of Illinois at Urbana-Champaign, and in 1998, the *Harry H. Goode Memorial Award* of the IEEE Computer Society for "innovative contributions to the field of electronic testing." Dr. Agrawal is a fellow of the IETE (India), a member of the ACM and a fellow of the IEEE. Intensely interested in education, he serves on the advisory boards of the ECE Departments at the University of Illinois and NJIT.