A Genetic Approach to Test Application Time Reduction for Full Scan and Partial Scan Circuits

Elizabeth M. Rudnick Janak H. Patel
Center for Reliable and High-Performance Computing
University of Illinois, Urbana, IL

Abstract—Full scan and partial scan are effective design-for-testability techniques for achieving high fault coverage. However, test application time can be high if long scan chains are used. Reductions in test application time can be made if flip-flop values are not scanned in and out before and after every test vector is applied. Previous research has used deterministic fault-oriented combinational and sequential circuit test generators in generating test vectors and sequences and in deciding when to scan the flip-flops. In this work we use genetic algorithms to generate compact test sets which limit the scan operations. Results for the ISCAS89 sequential benchmark circuits show that significant reductions in test application time can be achieved, especially for partial scan circuits.

I Introduction

Typical design-for-testability (DFT) techniques involve improving the controllability and observability of internal circuit nodes. Full scan design provides complete controllability and observability at the flip-flops, and combinational circuit test generators specifically targeted at generating compact test sets have been developed to limit the test application time for full scan designs [1]. However, test application time can still be long, especially for long scan chains, since flip-flop values must be scanned in and out before and after each test vector is applied. Multiple scan chains [2], parallel scan [3, 4], and selectable-length scan [5] have also been used to reduce test application time, but these approaches require greater hardware overhead. Selective scan [6, 7] has been used in a manner similar to selectable-length scan but without the hardware overhead; in this method, only required flip-flop values are shifted in and out. This method can be combined with ordering of flip-flops in the scan chain to minimize the test application time [6], but extra routing overhead may be incurred. Partial scan design has been proposed as a means of reducing the area overhead and performance penalties of the DFT hardware. Scan chain lengths are reduced, which may result in reductions in test application time. However, sequential circuit test generators are usually required since not all flip-flops are scanned, and the resulting test sets tend to be much less compact than those generated for full scan implementations. Partial reset is effective in reducing test application time, but fault coverage may also decrease [8].

*This research was supported by the Semiconductor Research Corporation under Contract SRC 93-DP-109.

Reductions in test application time have been made for some full scan circuits by eliminating many of the scan operations [9, 10]. A hybrid approach to test generation was used in [9], in which sequences of parallel vectors were generated in addition to scan vectors. We refer to vectors applied without scan as *parallel vectors* and vectors applied with scan as *scan vectors*. The improvement in fault coverage obtainable by a sequence of parallel vectors was compared to the maximum fault coverage improvement for a set of scan vectors requiring an equivalent test application time in deciding whether to apply the parallel vectors or the scan vectors in a given time frame [9]. More significant reductions in test application time were obtained by targeting faults in reverse order of testability and by including sequences in which flip-flop values are either scanned in or out, but not both [10]. Deterministic sequential circuit test generators were used to generate test sequences in both of these hybrid test generators, while combinational circuit test generators were used to generate the scan vectors. Since combinational circuit test generators are used, the same approach cannot be applied to partial scan circuits. Modifications to an existing sequential circuit test generator for reduction of test application time in partial scan circuits were reported in [11]. Again, faults are targeted in reverse order of testability, and for each fault, attempts are made to generate normal sequences of parallel vectors, scan-initiated sequences, or scan-terminated sequences to detect the fault. If these attempts are unsuccessful, the test generator resorts to using scan vectors.

Modifying a deterministic test generator to generate scan-terminated sequences is relatively easy, but handling scan-initiated sequences is much more difficult for partial scan circuits. The test generator must do reverse time processing to find a state in which only flip-flops in the scan chain have specified values. Backtracing decisions are typically made based upon node controllability values, but these values change depending on when the scan operation occurs. In a simulation-based approach to test generation, such as the one used in our work, these problems are avoided. Processing occurs in the forward direction only, and values to be scanned into the flip-flops are evolved by a genetic algorithm (GA). Vectors in the test sequences are also evolved by the GA, and a sequential circuit fault simulator is used to select the best normal or scan-initiated sequence to apply in a given time frame. Each test sequence is specifically targeted at detecting the maximum number of faults in the smallest test application time possible. Since the objective of the GA is to maximize improvement in fault coverage for each test generated, a very compact test set results. Furthermore, the same approach can be used for both full scan and partial scan circuits.

II Test Generation for Test Application Time Reduction

During testing of full scan and partial scan circuits, flip-flop values are typically scanned in and out before and after each test vector is applied. The total test application time, T, is

$$T = V(F+1) + F, \tag{1}$$

where V is the number of test vectors, and F is the number of flip-flops in the scan chain. Compact test sets can be used to limit the test application time. Alternatively, partial scan can be effective in reducing test application time, if the reduction in flip-flops is not offset by a corresponding increase in the number of test vectors required. A third alternative is to limit the scan operations. If scan operations are limited, flip-flops are not scanned before every test vector is applied. Then

$$T = V_N + V_{SI}(F+1), \tag{2}$$

where V_N is the number of vectors applied without using scan, and V_{SI} is the number of vectors applied which are preceded by a scan operation. A test generator used to generate tests which limit scan operations may not be able to achieve 100% coverage of testable faults. In this case, test generation can be done for any untested faults assuming all vectors use scan and further assuming that the state is scanned out after each vector is applied. Then Equations (1) and (2) can be combined to get the total test application time:

$$T = V_N + V_{SI}(F+1) + V_{SIO}(F+1) + F. \tag{3}$$

Here, V_{SIO} is the number of extra vectors applied which are directly preceded and followed by scan operations.

In this work, we use a GA-based test generator to generate test sequences. Both scan-initiated sequences and normal sequences are generated. In a *scan-initiated sequence*, the first vector is scanned but all successive vectors are parallel vectors. Sequences of parallel vectors are called *normal sequences*. The ratio of the number of faults detected to test application time is compared for the scan-initiated and normal sequences, and the sequence with the highest ratio is added to the test set for a given time frame.

Flip-flop values may or may not be scanned in before a test sequence is applied, and they may or may not be scanned out after a test sequence is applied. Thus a sequence may be normal, scan-initiated, scan-terminated, or both scan-initiated and scan-terminated. Faults whose effects propagate to the primary outputs (POs) are detected, but faults whose effects propagate to flip-flops in the scan chain after the last vector in the sequence is applied are also detected if the sequence is scan-terminated. Ideally, we would like to evolve sequences of each of the four types and then choose the best sequence. In evaluating the fitness of a sequence which is not scan-terminated, we would include only faults detected at the POs. Sequences which are not scan-initiated would not be considered if the previous sequence selected was scan-terminated, and faults detected at the flip-flops in a scanout operation at the end of the previous sequence would not be included in the fitness of the current scan-initiated sequence. However, the number of test sequences evaluated can be cut in half if the decisions on scan operations are restricted to whether a scan operation will occur before the current test sequence is applied. A test sequence becomes scan-terminated only if the next sequence is a scan-initiated sequence. The scanout operation of one sequence is overlapped with the scanin operation of the next sequence; therefore, no additional test application time is incurred due to the scanout operation.

For a given time frame, we generate both scan-initiated and normal sequences. In computing the fitness of a candidate sequence, we assume that the sequence will be scan-terminated. This fitness function encourages evolution of sequences which can propagate fault effects to flip-flops in the scan chain. If the next sequence added to the test set is scan-initiated, the faults will be detected in the scan operation. Otherwise a better sequence can be found which does not require a scan operation, most likely because the fault effects at the flip-flops can be propagated to the POs by the sequence. In addition, if the current sequence is scan-initiated, then faults detected at the flip-flops in the scan operation are also included in the fitness function. Thus, the fitness computed is an *approximation* of the actual quality of a test sequence. This approximation may result in a small increase in test application time, but the execution will be speeded up significantly. The sequence having the best ratio of number of faults detected to test application time is selected and added to the test set.

III Applying GAs to Test Generation

The simple GA, as described by Goldberg [12], contains a population of *strings*, or individuals. Each string is an encoding of a solution to the problem at hand. Each individual has an associated *fitness*, which gives an indication of the quality of the corresponding solution and thus depends on the application. The evolutionary processes of *selection, crossover,* and *mutation* are used to generate an entirely new population from the existing population. This process is repeated for m generations. To generate a new population from the existing one, two individuals are selected, with selection biased toward more highly fit individuals. The two individuals are crossed to create two entirely new individuals, and each character in a new string is mutated with some small mutation probability p. The two new individuals are then placed in the new population, and this process continues until the new generation is entirely filled. At this point, the previous generation can be discarded. In our work, we use tournament selection without replacement, uniform crossover, and 8 generations of nonoverlapping populations, each containing 32 strings. In *tournament selection without replacement,* two individuals are randomly chosen and removed from the population, and the best is selected; the two individuals are not replaced into the original population until all other individuals have also been removed. In *uniform crossover,* characters from the two parents are swapped with probability 1/2 at each string position in generating the two offspring. A crossover probability of 1 is used; i.e., the two parents are always crossed in generating the two offspring. Also, a mutation probability of 1/64 is used. These conditions gave the best results in our previous work in terms of fault coverage and execution times, and results of comparisons with other schemes are given in [13].

Because selection is biased toward more highly fit individuals, the average fitness is expected to increase from one generation to the next. However the best individual may appear in any generation, so we save the best individual found.

In applying GAs to test generation, successive vectors in a test sequence are placed in adjacent positions along a string in the population. The string is then an encoding of several successive test vectors to be applied to the primary inputs (PIs) of a circuit. We use a binary coding in which the 2 characters, 1 and 0, represent the values to be applied to the various PIs. For scan-initiated sequences, the state of the flip-flops to be scanned in is prefixed to each string, as illustrated in Figure 1. Each character in the string now corresponds to a 1 or 0 value

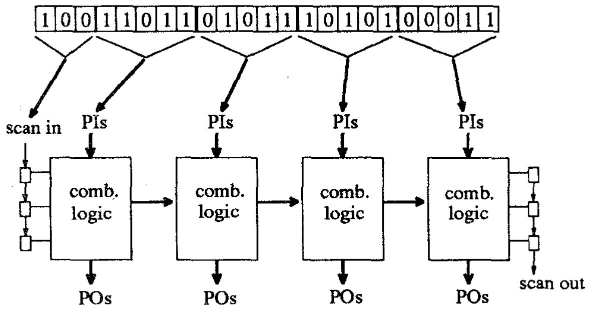

Figure 1: Generation of scan-initiated test sequences

to be scanned into a flip-flop or applied to a PI. Since the state evolves along with the test vectors, and selection is biased toward better individuals, a scan-initiated sequence detecting additional faults is expected to be generated. Scan operations are useful for both controllability and observability enhancement; therefore, additional faults which can be detected at the flip-flops if they are scanned out are included in the fitness function computation. Furthermore, these faults are included in the fault coverage if the next sequence is a scan-initiated sequence, since a scanout operation is overlapped with the scanin operation, as discussed previously.

The generation of scan-initiated sequences for partial scan circuits is the same as that for full scan circuits, except that only a subset of the flip-flops are scanned. Flip-flops not in the scan chain are simply loaded with the normal next state logic values from the previous time frame. Alternatively, *destructive scan* could be assumed in which flip-flops not in the scan chain are placed in an unknown state. In determining if additional faults are detected at the flip-flops in a scanout operation, only flip-flops in the scan chain are checked.

IV Test Generator Implementation

The GA-based test generation framework for scan circuits is illustrated in Figure 2. The GA generates populations of scan-initiated or normal sequences and selects the best test sequence to apply in the current time frame. The fitness of each candidate test is evaluated using the PROOFS sequential circuit fault simulator [14], and the fault simulator is also used to update the state of the circuit and to drop detected faults once a test is generated. The GA is reinitialized with random sequences for each attempt at improving the fault coverage.

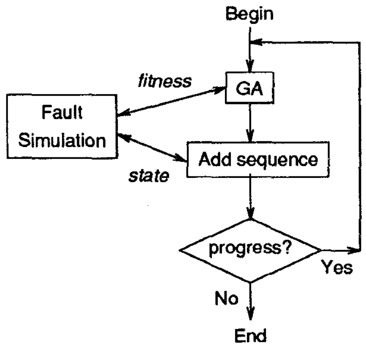

Figure 2: GA-based test generation for scan circuits

The overall objective of test generation is detection of all testable faults in the smallest test application time possible. In trying to achieve this objective, we attempt to find a test sequence which has a large ratio of number of faults detected to test application time. A normal sequence of moderate length typically has a lower test application time than a shorter scan-initiated sequence. However, the scan-initiated sequence has the potential to detect more faults. Therefore, we use a GA to generate both scan-initiated and normal sequences for the current time frame. We typically use a normal sequence length, L_N, of half the sequential depth of the circuit. The length of the scan-initiated sequence, L_S, is two in this work by default. In addition we also generate a test sequence of length one, since the computation cost is relatively small, and the smaller sequence may reduce the test application time. To select the sequence to be applied at the current time frame, the test generator compares the fitness of each of the three tests, namely the scan-initiated sequences of lengths 1 and 2 and the normal sequence of length L_N, and selects the best among these three if it results in an improvement in fault coverage.

In this work, we use 200-fault samples in evaluating the fitness of each candidate test, although the full fault list could be used to obtain a more accurate result, at the cost of increased execution time. The fitness values of scan-initiated and normal sequences are calculated as follows:

$$fitness(S) = \frac{C + \#\ faults\ scanned\ out}{L_S + 2F} \tag{4}$$

$$fitness(N) = \frac{C + \#\ faults\ scanned\ out}{L_N + F}, \tag{5}$$

where L_S and L_N are the lengths of the scan-initiated and normal sequences, F is the number of flip-flops, $L_S + 2F$ and $L_N + F$ are measures of the test application time, and C is a measure of the fault coverage:

$$\begin{aligned} C &= \#\ faults\ detected \\ &+ \frac{\#\ faults\ propagated\ to\ flip\ flops}{(\#\ faults\ simulated)(\#\ flip\ flops)}. \end{aligned} \tag{6}$$

Since detected faults are dropped, only undetected faults are targeted. The fraction of fault effects propagated to flip-flops is included in the fitness function, since a fault whose effects propagate to the flip-flops in one time frame is very likely to be detected in the next time frame. This metric is useful in guiding the GA to generate sequences which detect a greater number of faults. The number of faults detected at the flip-flops if the sequence is scan-terminated is included separately. If the next sequence is scan-initiated, these faults will be detected. Otherwise, they are still more likely to be detected by the next sequence.

Alternatively, faults detected at the POs could be separated from faults which would be detected at the flip-flops if the sequence is scan-terminated. The cost of the faults detected in the scanout operation is then higher in terms of test application time than faults detected at the POs, and the fitness functions are modified as follows:

$$fitness(S) = \frac{C}{L_S + F} + \frac{\#\ faults\ scanned\ out}{L_S + 2F} \quad (7)$$

$$fitness(N) = \frac{C}{L_N} + \frac{\#\ faults\ scanned\ out}{L_N + F}. \quad (8)$$

Test generation terminates after four consecutive attempts at improving the fault coverage fail. The GA-based test generator does not always achieve 100% fault coverage. For the remaining faults we assume that each vector is scan-initiated and scan-terminated and use the deterministic test generator HITEC [15] after the GA-based test generator terminates.

V Results

A scan capability was added to the existing GA-based sequential circuit test generator in 400 additional lines of C++ code. Tests were generated for full scan implementations of the ISCAS89 sequential benchmark circuits [16] on a SUN SPARCstation II with 64 MB memory. Circuit descriptions and test generation results are shown in Table 1. The structural sequential depth is the minimum number of flip-flops in a path between the PIs and the furthest gate. Each result is the average from ten runs, with a different random seed used for each run. Standard deviations are shown in parentheses. The exception is s35932, for which results of a single run are reported due to the long execution time. The number of testable faults was determined using the HITEC sequential circuit test generator [15]. The number of faults detected by the GA-based sequential circuit test generator with the scan capability is shown, along with the number of test cycles required to apply the generated tests, as calculated using Equation (2). Additional scan vectors were generated using HITEC to cover all remaining testable faults; the number of extra scan vectors generated and the total number of test cycles required, as calculated using Equation (3), are shown. Execution times are also shown, and these times include all phases of test generation. The total number of test cycles required for COMPACTEST (**COMP**) [1] and TARF [10] are shown for comparison. The best results are highlighted. Note that COMPACTEST and TARF results were not reported for some of the circuits. Only two circuits had higher test application times than those for TARF [10], which used deterministic, fault-oriented test generators: s1196 and s1238. Execution times were not reported for TARF. The genetic approach gave lower test application times than COMPACTEST for all circuits except s5378 and s35932. Execution times for COMPACTEST can be expected to be considerably faster than those for the genetic approach.

In generating the test sets, the test generator forces a scan operation before any test vectors are applied to initialize the flip-flops. This feature has been found to give better results than the alternative of allowing a normal sequence at the beginning of a test set. Also, the fitness functions defined in Equations (4) and (5) are used. Normal test sequence lengths of half the sequential depth were used for all circuits except s5378 and s35932; for these circuits, test sequence lengths of one-quarter of the sequential depth were used to reduce the execution time. Scan-initiated test sequence lengths of 1 and 2 were used. Experiments were also conducted using various alternative approaches; results for these experiments are shown in Table 2. Again, each result is the average from ten runs, with a different random seed used for each run; standard deviations are shown in parentheses. For circuits s1196 and s1238, using the fitness functions in Equations (7) and (8) reduced the test application time by 29%, but for most circuits the fitness functions in Equations (4) and (5) gave better results. Using scan-initiated sequence lengths of 1 and 3 vectors reduced the test application time by 10% for circuit s1423 but resulted in increases in test application time for several other circuits.

Improvements in test application time for partial scan circuits are shown in Table 3. A normal sequence length equal to the sequential depth was used for all circuits except s5378 and s35932, i.e., a value twice that used for full scan circuits. For s5378 and s35932, test sequence lengths of one-quarter of the sequential depth were used. Longer sequences are needed to achieve high fault coverages in partial scan circuits, since not all flip-flops are directly controllable and observable. The number of flip-flops selected to be placed in the scan chain is shown in the table for each circuit. The OPUS partial scan selection tool [17] was used to select a minimal set of flip-flops to break all cycles in the circuit graph for all circuits. For circuits s526, s1423, and s5378, additional flip-flops were selected using controllability and observability measures; 30%, 50%, and 40% of the flip-flops were selected, respectively, for the three circuits. The number of faults detected, the HITEC execution time, and the number of faults found to be untestable by HITEC are shown for the resulting partial scan circuits. Also shown is the number of test cycles required to apply the test vectors generated, assuming the flip-flop values are scanned in and out before and after each test vector is applied. Results for the GA-based test generator are shown next, including the total number of test cycles required and test generation times. Results for all circuits except s35932 are averaged over ten runs, with a different random seed used for each run; standard deviations are shown in parentheses. After the GA-based test generator was run, additional tests were generated by HITEC, if necessary, to achieve the fault coverage reported in the table. The number of test cycles reported includes the extra cycles required to apply these additional scan vectors, and the test generation time includes the extra HITEC execution time. For circuit s1423, 19 more faults were detected, on average, with the genetic approach as compared to the deterministic approach. Significant improvements in test application time were achieved for all circuits. Different partial scan flip-flops were used in TARP [11]; therefore, the results cannot be compared directly, but significant improvements were also observed for the deterministic approach. Results have also been

Table 1: Results for Full Scan Circuits

Cir-cuit	Flip-Flops	Seq Depth	Faults		Test Cycles	Extra Vec	Exec Time	Total Cycles		
			Testable	Detected				GA	COMP	TARF
s298	14	8	308	308(0)	233(14)	0	2.92m	**233(14)**	404	376
s344	15	6	342	342(0)	123(24)	0	1.95m	**123(24)**	139	166
s349	15	6	348	348(0)	123(12)	0	2.30m	**123(12)**	239	125
s382	21	11	399	399(0)	478(50)	0	4.84m	**478(50)**	593	680
s386	6	5	384	384(0)	303(12)	0	5.10m	**303(12)**	496	376
s400	21	11	420	420(0)	470(25)	0	5.61m	**470(25)**	-	-
s444	21	11	460	460(0)	439(17)	0	5.49m	**439(17)**	615	788
s526	21	11	554	552.5(1.8)	949(81)	1.5	14.1m	**995(55)**	1231	1551
s641	19	6	467	459.4(1.1)	269(32)	7.5	5.79m	**438(27)**	-	-
s713	19	6	543	534.7(0.9)	285(35)	8.2	9.00m	**466(46)**	-	-
s820	5	4	850	846.9(2.1)	422(34)	3.0	18.1m	**445(28)**	641	617
s832	5	4	856	853.3(1.3)	417(15)	2.4	18.0m	**437(17)**	623	589
s1196	18	4	1242	1236.9(2.5)	742(65)	5.0	38.7m	854(46)	2640	**465**
s1238	18	4	1286	1281.5(1.9)	813(69)	4.5	46.0m	917(51)	2773	**448**
s1423	74	10	1501	1497.1(1.4)	2036(191)	3.2	53.7m	**2450(184)**	2624	3222
s1488	6	5	1486	1485.8(0.4)	396(16)	0.2	22.5m	**396(15)**	846	641
s1494	6	5	1494	1493.8(0.4)	394(19)	0.2	23.6m	**399(16)**	811	582
s5378	179	36	4563	4551.9(4.3)	33,276(820)	8.4	7.56h	34,967(540)	**19,979**	-
s35932	1728	35	35,110	35,106	31,285	3	117.9h	38,200	**24,205**	-

Table 2: Alternative Approaches for Full Scan Circuits

Circuit	Total Cycles				Execution Time (min)			
	A	B	C	D	A	B	C	D
s298	246(23)	231(11)	241(19)	239(21)	5.48	2.72	3.24	2.64
s344	147(18)	129(14)	122(20)	99(18)	4.18	1.85	2.19	1.69
s349	165(16)	116(16)	120(13)	103(12)	4.42	2.21	2.52	2.13
s382	499(23)	476(29)	528(32)	490(29)	6.94	4.75	4.87	4.48
s386	301(11)	304(12)	330(10)	323(10)	5.45	5.42	5.57	5.20
s400	504(26)	445(27)	512(30)	497(32)	8.21	5.36	5.39	5.21
s444	497(30)	470(25)	506(30)	474(30)	11.1	6.45	5.88	5.57
s526	989(43)	987(67)	1043(46)	1092(46)	23.5	12.8	13.6	13.9
s641	475(31)	468(24)	467(38)	514(38)	9.50	6.14	6.82	6.25
s713	462(47)	462(32)	474(50)	499(33)	12.5	9.62	10.6	9.56
s820	443(21)	467(22)	474(27)	483(20)	19.6	18.6	19.8	18.1
s832	426(29)	444(20)	470(17)	477(15)	20.4	18.7	19.8	18.6
s1196	841(52)	920(49)	604(38)	858(33)	38.5	38.7	39.2	37.0
s1238	917(74)	972(94)	646(30)	904(55)	44.9	48.5	46.2	45.7
s1423	2331(134)	2209(170)	2346(181)	2623(218)	110.4	57.9	60.9	55.7
s1488	411(22)	405(30)	413(25)	390(22)	39.4	22.2	23.2	21.1
s1494	381(23)	407(21)	401(16)	397(16)	37.3	23.8	23.8	22.3

A: do not force scan before first vector
B: use scan-initiated sequence lengths of 1 and 3 vectors
C: use fitness functions in Equations (7) and (8)
D: use normal sequence length equal to sequential depth

Table 3: Results for Partial Scan Circuits

Circuit	Flip-Flops	Total Faults	Faults Det	HITEC			GA	
				Unt	Cycles	Time	Cycles	Time
s298	1	308	292	15	1517	1.21m	1039(345)	6.98m
s344	5	342	340	2	581	6.37s	163(21)	4.72m
s349	5	350	346	3	671	24.4s	125(20)	4.13m
s382	9	399	394	5	1579	5.27s	1003(126)	7.24m
s386	5	384	384	0	1007	3.55s	424(16)	5.51m
s400	9	426	415	11	1689	5.65s	998(113)	7.88m
s444	9	474	455	19	1699	9.11s	1009(81)	10.2m
s526	6	555	544	11	14,216	6.30m	10,430(2451)	21.7m
s641	7	467	442	25	1031	17.4s	193(19)	7.83m
s713	7	581	514	67	1015	48.4s	199(28)	10.9m
s820	4	850	850	0	1789	11.7s	547(45)	21.0m
s832	4	870	860	10	1949	13.5s	746(70)	20.2m
s1196	0	1242	1239	3	460	22.7s	246(14)	50.1m
s1238	0	1355	1283	72	469	34.6s	253(10)	56.8m
s1423	37	1515	1462	13	13,907	2.70h	2387(686)	1.32h
s1488	5	1486	1486	0	1679	21.8s	415(33)	24.5m
s1494	5	1506	1498	8	1739	22.6s	399(19)	31.4m
s5378	72	4603	4474	122	58,326	34.8m	21,042(888)	6.90h
s35932	306	39,094	35,110	3984	156,262	5.74h	114,364	35.0h

presented for selective scan with a two-phase approach to test generation, in which normal sequences are first generated for easy-to-detect faults, and then scan vectors are generated for hard-to-detect faults [7]; again results cannot be compared directly, since different partial scan flip-flops were used and the fault coverages were lower.

VI Conclusions

High test application time can be a problem for full scan and partial scan circuits. Significant reductions in test application time can be made if flip-flop values are not scanned in and out for every test vector. We use a genetic approach to generating compact test sets which limit the scan operations. Results for the ISCAS89 benchmark circuits show that reductions in test application time can be obtained for most small full scan circuits. For the largest full scan circuits, combinational circuit test generators are available which produce test sets so compact that the genetic approach cannot compete. However, combinational circuit test generators cannot be used for partial scan circuits, and the genetic approach provides significant improvements over the traditional approach. Further improvements can be obtained by combining this method with a multiple-scan-chain approach.

References

[1] I. Pomeranz, L. N. Reddy, and S. M. Reddy, "COMPACTEST: A method to generate compact test sets for combinational circuits," *Proc. Int. Test Conf.*, pp. 194-203, 1991.

[2] S. Narayanan, R. Gupta, and M. Breuer, "Configuring multiple scan chains for minimum test time," *Proc. Int. Conf. Computer-Aided Design*, pp. 4-8, 1992.

[3] S. M. Reddy and R. Dandapani, "Scan design using standard flip-flops," *IEEE Design & Test*, pp. 52-54, February 1987.

[4] B. Vinnakota and N. K. Jha, "Synthesis of sequential circuits for parallel scan," *Proc. European Conf. Design Automation*, pp. 366-370, 1992.

[5] S. P. Morley and R. A. Marlett, "Selectable length partial scan: A method to reduce vector length," *Proc. Int. Test Conf.*, pp. 385-392, 1991.

[6] R. Gupta and M. A. Breuer, "Ordering storage elements in a single scan chain," *Proc. Int. Conf. Computer-Aided Design*, pp. 408-411, 1991.

[7] W. -J. Lai, C.- P. Kung, and C.- S. Lin, "Test time reduction in scan designed circuits," *Proc. European Conf. Design Automation*, pp. 489-493, 1993.

[8] B. Mathew and D. G. Saab, "Partial reset: An alternative DFT approach," *VLSI Design*, vol. 1, no. 4, pp. 299-311, 1994.

[9] D. K. Pradhan and J. Saxena, "A design for testability scheme to reduce test application time in full scan," *Proc. VLSI Test Symp.*, pp. 55-60, 1992.

[10] S. Y. Lee and K. K. Saluja, "An algorithm to reduce test application time in full scan designs," *Proc. Int. Conf. Computer-Aided Design*, pp. 17-20, 1992.

[11] S. Y. Lee and K. K. Saluja, "Sequential test generation with reduced test clocks for partial scan designs," *Proc. VLSI Test Symp.*, pp. 220-225, 1994.

[12] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning,* Addison-Wesley, 1989.

[13] E. M. Rudnick, J. H. Patel, G. S. Greenstein, and T. M. Niermann, "Sequential circuit test generation in a genetic algorithm framework," *Proc. Design Automation Conf.*, pp. 698-704, 1994.

[14] T. M. Niermann, W. -T. Cheng, and J. H. Patel, "PROOFS: A fast, memory-efficient sequential circuit fault simulator," *IEEE Trans. Computer-Aided Design*, pp. 198-207, February 1992.

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

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

[17] V. Chickermane and J. H. Patel, "An optimization based approach to the partial scan design problem," *Proc. Int. Test Conf.*, pp. 377-386, 1990.