New Techniques for Deterministic Test Pattern Generation*

ILKER HAMZAOGLU AND JANAK H. PATEL
Center for Reliable & High-Performance Computing, University of Illinois, Urbana, IL 61801, USA

Editors: M. Nicolaidis and R. Roy

Abstract. This paper presents new techniques for speeding up deterministic test pattern generation for VLSI circuits. These techniques improve the PODEM algorithm by reducing number of backtracks with a low computational cost. This is achieved by finding more necessary signal line assignments, by detecting conflicts earlier, and by avoiding unnecessary work during test generation. We have incorporated these techniques into an advanced ATPG system for combinational circuits, called ATOM. The performance results for the ISCAS85 and full scan version of the ISCAS89 benchmark circuits demonstrated the effectiveness of these techniques on the test generation performance. ATOM detected all the testable faults and proved all the redundant faults to be redundant with a small number of backtracks in a short amount of time.

1. Introduction

As the complexity of VLSI circuits and their quality requirements are increasing, the problem of test generation is becoming more important. Since the scan-based design for testability techniques are increasingly used in very large circuits, test generation for combinational circuits is getting even more important for these circuits.

The test generation problem can be viewed as a finite space search problem of finding appropriate logic assignments to the primary inputs such that the given fault is detected [1]. Since the size of the search space is exponential in the number of primary inputs and the test generation problem is proven to be NP-complete [2], it is very important to find efficient techniques to speed up the test generation process. After the branch and bound search algorithm introduced in PODEM [1], a number of techniques have been proposed in the literature to improve its performance [3–10].

The test generation problem can also be viewed as a boolean satisfiability (SAT) problem. Because it is possible to reduce it to an equivalent SAT problem using a polynomial time reduction. Recently a number of test generation systems are introduced for solving the test generation problem using fast SAT solvers [11–14].

Although there are effective ATPG algorithms, their average-case performance is decreasing because of the ever increasing complexity of today's VLSI circuits. Therefore, we propose new structure based techniques for speeding up the test pattern generation for VLSI circuits. These techniques are improved unique sensitization, improved backtrace with X-path and conflict check, multiple primary input assignments, lookback, and support region. They improve the average-case performance of the PODEM algorithm by reducing number of backtracks with a low computational cost. This is achieved by finding more necessary signal line assignments, by detecting conflicts earlier, and by avoiding unnecessary work during test generation.

To be able to assess the effectiveness of the proposed techniques, we have developed a deterministic ATPG system for combinational circuits, called ATOM, and incorporated these techniques into the test generator.

The test generator uses the single stuck at fault model and it is based on the PODEM algorithm. In addition to the techniques we have proposed, it also employs some of the test generation techniques introduced in the literature.

The experimental results on the ISCAS85 and full scan version of the ISCAS89 benchmark circuits [15, 16] demonstrated the effectiveness of these techniques on the test generation performance. ATOM detected all the testable faults and proved all the redundant faults to be redundant with very small number of backtracks in a short amount of time. Since we did not use an initial random test generation phase and we obtained similar results both with and without using fault simulation, the results demonstrated the robustness of the techniques we have developed.

The rest of the paper is organized as follows. Section 2 introduces the test generation system. Section 3 discusses the proposed test generation techniques in detail. Section 4 presents the experimental results. Finally, Section 5 presents the conclusions.

2.   The Test Generation System

We have developed an ATPG system for combinational circuits, called ATOM. ATOM is composed of a parallel-pattern fault simulator and a PODEM based deterministic test pattern generator. To be able to assess the robustness of the test generation techniques we have developed, it does not use a random test generation phase. It is designed in a object oriented style and implemented in C++.

The fault simulator is a parallel-pattern simulator that simulates 32 patterns at a time [10]. It is also a parallel-fault simulator, since it simulates the faults in a fanout free region until the fanout stem, and it only simulates the faults on the fanout stem in the rest of the circuit. It also employs the fault simulation techniques used in PROOFS fault simulator [17].

The test generator is based on the PODEM algorithm. It uses the single stuck at fault model and the five-valued logic with logic values 0, 1, X, D, $\bar{D}$. It uses the observability and controllability values based on the SCOAP [18, 19] controllability/observability measures to guide the search process. In addition to the new test generation techniques that we have developed, the test generator uses the local implications [4], static global implications [7], X-path check [1], unique sensitization [4], and dynamic unique sensitization [8] techniques proposed in the literature. We did not use

any compact test set generation techniques in the test generator.

In [8, 9, 14, 20], it is reported that using costly test generation techniques for easy to detect faults degrades the overall performance of an atpg system. We have made the same observation. Therefore, ATOM runs in two phases. In the first phase low cost test generation techniques, local implications, X-path check, unique sensitization, and support region, are used to test as much faults as possible with a maximum backtrack limit of 0, i.e. without any backtracks. Then, in the second phase more costly test generation techniques, static global implications, dynamic unique sensitization, improved unique sensitization, backtrace with X-path and conflict check, multiple primary input assignments and lookback, are used to test the remaining faults.

3.   New Test Generation Techniques

We will now describe the new techniques we propose to improve the test generation performance.

3.1.   *Improved Unique Sensitization*

The dynamic unique sensitization technique introduced by the SOCRATES system [8] improved the effectiveness of the unique sensitization technique proposed by the FAN algorithm [4]. This technique identifies more necessary logic assignments due to fault propagation constraints that are not considered in the basic unique sensitization technique. We have further improved the dynamic unique sensitization technique to identify more necessary logic assignments. The new technique finds the necessary assignments as follows.

Suppose that the D-frontier consists of the gate g_1, and d_1 is the first static dominator of g_1 in the levelized order that has more than one fanout. Note that if g_1 itself has more than one fanout, then d_1 is same as g_1. Let us assume that d_1 fans out to the gates $f_1, f_2, \ldots, f_n$, and the necessary signal line assignments for propagating the fault from these gates, which can be identified using static unique sensitization technique, are contained in the sets $a_1, a_2, \ldots, a_n$ respectively. Then, the signal line assignments in CO = *Logic_Implications*(a_1) $\cap \cdots \cap$ *Logic_Implications*(a_n) are necessary to propagate the fault from any one of the fanout gates
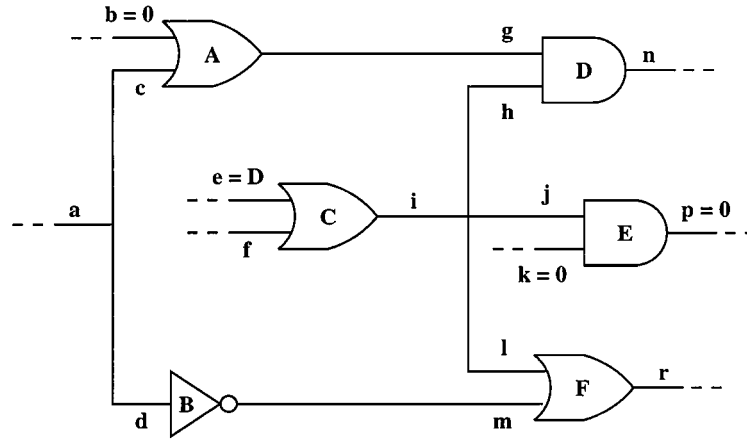
Fig. 1. Improved unique sensitization.

$f_1, f_2, \ldots, f_n$. Therefore, CO is the set of necessary logic assignments for detecting the given fault.

As the dynamic unique sensitization technique, this technique is also applicable to the situations where there is more than one gate in the D-frontier. However, when there are multiple gates in the D-frontier, finding the necessary assignments becomes costly. Therefore, we only use this technique when there is only one gate in the D-frontier. We have found that this new dynamic unique sensitization technique discovers a large number of necessary logic assignments that can not be discovered by SOCRATES. These necessary logic assignments speed up the test pattern generation by identifying conflicts earlier.

Example.   Figure 1 illustrates a situation in which this new technique discovers a necessary logic assignment that can not be discovered by SOCRATES. In this example, let us assume that the D-frontier consists of the gate C. In this situation, new technique will determine that $a = 1$ is a necessary logic assignment. Because to propagate the fault from gate D, signal g should be set to 1, and $g = 1$ implies $a = 1$. Similarly to propagate the fault from gate F, signal m should be set to 0, and $m = 0$ implies $a = 1$. Since gate E is already assigned the logic value 0, the fault can not be propagated through gate E. Therefore, signal a should be set to 1 regardless of through which gate the fault will be propagated. SOCRATES will not be able to find this necessary logic assignment, because $a = 1$ is not a dynamic dominator of gate C.

3.2.   *Improved Backtrace with X-path and Conflict Check*

After selecting a gate for fault propagation, PODEM carries out X-path check for this gate to determine whether there is a path from this gate to at least one of the primary outputs on which all lines are at X. If there is no such path then it selects another gate. This X-path check is carried out before the backtrace procedure. Although an X-path may exist before the backtrace starts, an OR-choice selected during backtrace may destroy these X-paths. We have improved the backtrace procedure to detect these situations during backtrace before committing to a primary input assignment.

Example.   Figure 2 illustrates such a situation. In this case, let us assume that the objective is to propagate the fault from gate C. Therefore signal e should be set to 1. This can be justified by either setting signal a or b to 1. However, setting signal a to 1 implies that signal i will be 1. If signal i becomes 1, then there will not be any X-paths remaining for the gate C. Thus if the backtrace procedure continues by selecting $a = 1$ choice, the resulting primary input assignment may not satisfy the initial objective, propagating the fault from gate C to a primary output. Therefore our test generator discards $a = 1$ OR-choice, and it continues to backtrace by selecting $b = 1$ OR-choice.

PODEM uses only the controllability measures to select an OR-choice at an OR decision point during backtrace. However, during backtrace sometimes an OR-choice conflicts with the choices selected earlier
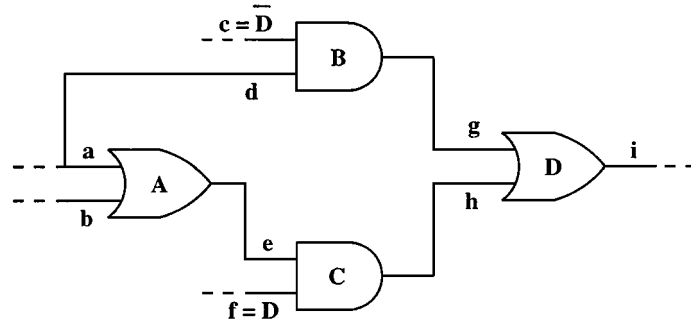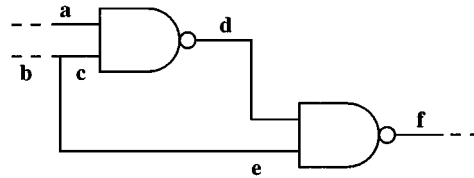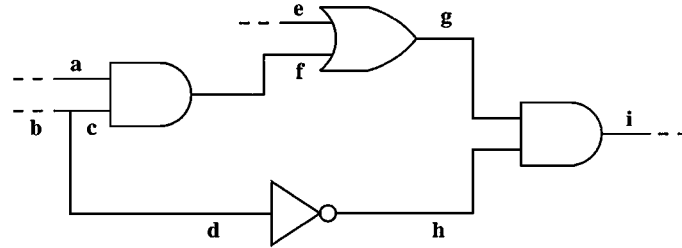
Fig. 2.   Backtrace with X-Path check.

(a) Direct Conflict

(b) Indirect Conflict

Fig. 3.   Backtrace with conflict check.

in the backtrace. If this conflict is not considered, then the resulting primary input assignment may not satisfy the initial objective. Therefore, we have improved the backtrace procedure with an extensive conflict check to determine the possible conflicts before committing to a primary input assignment. This avoids the test generator from going into a nonsolution area in the search space which might result in a significant number of backtracks.

Example.   Figure 3(a) illustrates a situation in which a later OR-choice directly conflicts with an earlier AND-choice. In this example, let us assume that $f = 0$ is the current objective of the backtrace procedure. Justifying this objective requires setting the signals d and e to 1. If the backtrace procedure chooses the AND-choice $d = 1$, then this can be justified by either setting signal

a or signal c to 0. The OR-choice $c = 0$ clearly conflicts with the $e = 1$ AND-choice. In this example, the improved backtrace procedure will determine this conflict, and it will continue to backtrace by choosing the $a = 0$ OR-choice.

The above example illustrates the basic improvement. However, there are more complicated cases in which a later OR-choice does not directly conflict with the earlier choices, rather the logic implications of this OR-choice conflicts with the earlier choices.

Example.   Figure 3(b) illustrates such a situation. In this example, let us assume that $i = 1$ is the current objective of the backtrace procedure. Justifying this objective requires setting the signals g and h to 1. If the backtrace procedure chooses the AND-choice $g = 1$,

then this can be justified by either setting signal e or signal f to 1. Although $f = 1$ does not directly conflict with the earlier choices, setting signal f to 1 implies that $h = 0$ which conflicts with the $h = 1$ AND-choice. The improved backtrace algorithm will determine this conflict, and it will continue to backtrace by choosing $e = 1$ OR-choice.

The following observation about detecting the conflicts during a backtrace simplifies the improved backtrace procedure. To determine the possible conflicts during a backtrace, it is sufficient to check only the OR-choices. It is not necessary to check whether the choices at an AND decision point will cause a conflict. Since the previous OR-choice will imply each one of these AND-choices, this will already be checked when we consider the implications of the previous OR-choice.

Therefore, the improved backtrace procedure works as follows. During the backtrace a backtrace-stack is built to keep track of the earlier OR-choices. At each OR decision point, we push all the choices and the logic implications of the selected choice to the stack marking the selected choice as tried. At each OR decision point the backtrace procedure checks whether the OR-choice destroys all the X-paths of the initial objective and whether it directly or indirectly conflicts with the choices selected earlier in the backtrace. If an OR-choice causes a conflict, one of the untried choices is selected next. If all the choices at an OR decision point cause a conflict, the backtrace procedure backtracks until the last OR-choice. This way it tries all possible OR-choices until it backtraces to a primary input without any conflicts or there is no untried OR-choice left in which case the test generator backtracks.

For some circuits, this technique may cause a large number of backtracks during a backtrace degrading the overall performance of the test generator. To avoid this situation, we use a maximum backtrack limit of 20 in our test generator. If this limit is exceeded during a backtrace, this technique is no longer used for this circuit, i.e. the backtrace procedure is executed without any conflict checks. Among the ISCAS circuits, this maximum backtrack limit is exceeded only during the test generation for c6288 when the test generator is executed without fault simulation.

3.3. Multiple Primary Input Assignments

Multiple backtrace technique introduced by the FAN algorithm concurrently backtraces more than one objective and it may assign logic values to more than one primary input at the end of a backtrace. We have also improved the single backtrace technique of PODEM to assign logic values to more than one primary input at the end of a backtrace. As opposed to multiple backtrace technique, our technique does not require backtracing multiple objectives. This new technique speeds up the test pattern generation by reducing the number of backtraces.

Multiple primary input assignments is achieved by exploiting the backtrace-stack constructed during the single backtrace procedure. The backtrace-stack contains the OR-choices selected during backtrace and their logic implications. We have improved the test pattern generator to check the backtrace-stack after a backtrace to determine whether any of the OR-choices selected during backtrace implies a logic value for a primary input which is not assigned a logic value by the backtrace procedure and which is not already in the primary input decision tree. If such a logic implication exist, we then put the corresponding primary input into the decision tree by assigning the implied logic value.

3.4. LookBack

When a conflict occurs in the test generation process, PODEM backtracks to the last primary input decision. The dependency-directed backtracking technique introduced in [21] and [9] improved the PODEM algorithm to nonchronologically backtrack to an earlier primary input decision. When a conflict occurs in the test generation process, dependency-directed backtracking technique carries out an extensive analysis to determine the cause of the conflict and it backtracks to the last primary input decision that is responsible from the conflict. However, the conflict analysis is a quite costly process.

We have improved the PODEM algorithm to nonchronologically backtrack to an earlier primary input decision using a new search technique called LookBack. LookBack is a probing technique which is triggered after the maximum backtrack limit of the test generator is reached. In this case, instead of continuing with the branch and bound search algorithm, we save the current search state, i.e. the primary input decision tree, and use the lookback technique. LookBack is effective in the situations where an earlier primary input decision causes the test generator to go into a large nonsolution area. It tries to avoid spending too much time in this nonsolution area and tries to quickly

```
LOOKBACK(DT: decision-tree, LookBack-Backtrack_Limit: int)
{
  While DT is not empty
  {
      current_pi = last primary input in DT
      Complement the logic value of current_pi
      no_of_backtracks = 0
      DT' = DT      // save the DT
      While (no_of_backtracks < LookBack-Backtrack_Limit)
      {
          Perform branch and bound search starting from DT
          If test vector is found then return SUCCESS
          If conflict then
          {
            no_of_backtrack = no_of_backtracks + 1
            backtrack to the previous primary input decision in DT
          }
      }
      DT = DT'      // restore the DT
      backtrack to the previous primary input decision in DT
  }
  return FAILURE
}
```

Fig. 4. LookBack algorithm.

reach to a solution area without carrying out any conflict analysis.

As it is shown in Fig. 4, LookBack procedure takes two parameters, the current decision tree and a lookback-backtrack limit. In our test generator, we used a lookback-backtrack limit of 1. Starting from the last primary input decision in the decision tree lookback procedure complements the logic value of the primary input and runs the PODEM algorithm until the given lookback-backtrack limit is reached. If no solution is found within this lookback-backtrack limit, it backtracks to the previous primary input decision in the original decision tree. This way instead of spending a large amount of time in the nonsolution area, it quickly backtracks to the earlier primary input decision that caused the test generator to enter into the nonsolution area. Since lookback is not a complete algorithm, if it fails to find a test vector, then the test generator continues with the branch and bound search algorithm starting from the original decision tree.

Example. Figure 5 illustrates a situation where the lookback technique helps the test generator to quickly get rid of the nonsolution area and detect the fault. In this figure, the unshaded part of the triangles represent the nonsolution areas that are completely searched by lookback whereas the shaded part of the triangles re-
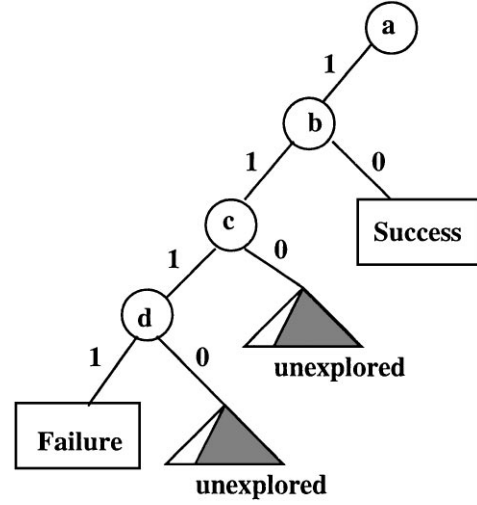

Fig. 5. LookBack example.

present the nonsolution areas that are not explored at all by lookback, one of the square boxes represents a conflict, i.e. failure, and the other one represents a solution. Let us assume that the test generator reached to its maximum backtrack limit, and the current decision tree consists of the decisions $a = 1$, $b = 1$, $c = 1$ and $d = 1$. In this example, the branch and bound algorithm may take a large number of backtracks to come back to the decision tree node b and find the solution. Since lookback only explores a small portion of each triangle, the size of which is determined by the lookback-backtrack limit given to the lookback procedure, it quickly backtracks to the decision tree node b and finds the solution.

3.5. Support Region

We define the *support region* for a fault in a given circuit as the part of the circuit that may affect the test generation process for this fault. As illustrated in Fig. 6, for a given faulty gate this region consists of the gates in the fanin cones of the primary outputs that are in the fanout cone of this faulty gate. Since the gates outside the support region do not fanout to the same primary outputs with the faulty gate, they do not effect the test generation process for this faulty gate. Therefore, even though a logic value of a gate in the support region may imply a logic value for a gate outside the support region, during test generation it is not necessary to carry out forward implications outside the support region of the target fault. This technique avoids considerable amount of unnecessary work during the test generation.
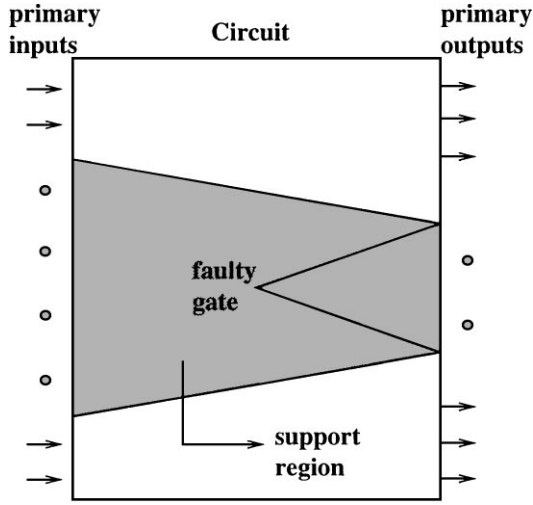
Fig. 6. Support region example.

Although this technique requires the computation of the support regions, the cost of this computation can be reduced by computing this information only once for a fanout free region in a demand-driven way. Since the support region for all the faults in a fanout free region is the same, this information can be computed once for a fanout free region, and it can be used for all the faults in the same fanout free region. Moreover, since we do not use this technique in the fault simulator, it is not necessary to compute the support regions for the faults that are detected by fault simulation. Therefore, we compute the support regions only for the faults that are targeted by the test generator.

4. Experimental Results

ATOM is tested on the ISCAS85 and full scan version of the ISCAS89 benchmark circuits on a 200 MHz Pentium Pro PC with 128MB RAM running Linux 2.0.0 using GNU CC version 2.8.0. Table 1 presents the performance results. In all these experiments a maximum backtrack limit of 6 is used, and the lookback technique is triggered only after 6 backtracks.

The columns Det, Red, Abt, Vect present the number of faults detected, the number of faults proven to be redundant, number of aborted faults and the number of test vectors generated respectively. We did not present the the number of faults detected, the number of faults proven to be redundant and the number of test vectors generated for the without fault simulation case. Because the number of faults detected and the number

of faults proven to be redundant are same for both with and without fault simulation cases, and the number of test vectors for the without fault simulation case is same as the number of detected faults, i.e. one test vector is generated for each fault. The results show that ATOM detected all testable faults and proved all redundant faults to be redundant. Since we used a parallel-pattern fault simulator, and we didn't use any compact test set generation techniques, the test set sizes generated by ATOM are relatively large. In this paper, we do not address the compact test set generation problem.

The TB column in the table presents the total number of backtracks done during the test generation for the corresponding circuit. The MDB column shows the maximum number of backtracks required to detect a fault and the MRB column shows the maximum number of backtracks required to prove that a fault is redundant. The MT1B column shows the number of faults that required more than one backtrack to be detected or to be proven as redundant. The results show that ATOM detected all testable faults and proved all redundant faults to be redundant with a very small number of backtracks.

The Time column for the with fault simulation case presents the total test generation time in seconds for the corresponding circuit, including the time for computing the dominators and the static global implications, the fault simulation time and all the input/output times. The Time column for the without fault simulation case also presents the total test generation time in seconds, but without including the time for writing the generated test vectors to a file. Since for each of the largest three ISCAS89 circuits the size of the test vector file is around 60MB, there is a considerable input/output time for these circuits. Since this input/output time is the same independent of the test generation algorithm used and it is determined by the performance of the input/output subsystem of the machine used, we did not include it in the total time presented for the without fault simulation case. The results show that ATOM detected all testable faults and proved all redundant faults to be redundant in a short amount of time without aborting any faults.

In summary, these performance results show that the new test generation techniques we have developed are quite successful in detecting testable faults and proving redundant faults to be redundant with a small number of backtracks in a short amount of time both with and without using fault simulation. This demonstrates the effectiveness and robustness of these new test generation techniques.

Table 1. ATPG results.

Circuit	ATOM with fault simulation									ATOM without fault simulation					
	Det	Red	Abt	Vect	TB	MDB	MRB	MT1B	Time	Abt	TB	MDB	MRB	MT1B	Time
c432	520	4	0	110	0	0	0	0	0.1	0	0	0	0	0	0.3
c499	750	8	0	127	0	0	0	0	0.3	0	0	0	0	0	1.1
c880	942	0	0	133	0	0	0	0	0.1	0	0	0	0	0	0.3
c1355	1566	8	0	192	3	1	0	0	2.0	0	66	3	0	12	12.7
c1908	1870	9	0	210	3	2	0	1	0.8	0	7	2	0	2	2.8
c2670	2630	117	0	242	4	0	4	1	1.1	0	4	0	4	1	3.9
c3540	3291	137	0	264	0	0	0	0	1.9	0	7	3	0	2	9.9
c5315	5291	59	0	216	0	0	0	0	1.2	0	28	17	0	3	6.7
c6288	7710	34	0	64	0	0	0	0	1.3	0	12478	32	0	1371	86.3
c7552	7419	131	0	393	7	1	0	0	4.6	0	83	4	0	5	23.6
s208	217	0	0	65	0	0	0	0	0.0	0	0	0	0	0	0.0
s298	308	0	0	52	0	0	0	0	0.0	0	0	0	0	0	0.1
s344	342	0	0	62	0	0	0	0	0.0	0	0	0	0	0	0.1
s349	348	2	0	65	0	0	0	0	0.0	0	0	0	0	0	0.1
s382	399	0	0	72	0	0	0	0	0.0	0	0	0	0	0	0.1
s386	384	0	0	109	0	0	0	0	0.1	0	0	0	0	0	0.1
s400	418	6	0	71	0	0	0	0	0.0	0	0	0	0	0	0.1
s420	455	0	0	98	0	0	0	0	0.1	0	0	0	0	0	0.1
s444	460	14	0	77	0	0	0	0	0.0	0	0	0	0	0	0.1
s510	564	0	0	90	0	0	0	0	0.1	0	0	0	0	0	0.2
s526	554	1	0	107	0	0	0	0	0.1	0	0	0	0	0	0.1
s526n	553	0	0	106	0	0	0	0	0.1	0	0	0	0	0	0.1
s641	467	0	0	99	0	0	0	0	0.1	0	0	0	0	0	0.2
s713	543	38	0	100	0	0	0	0	0.1	0	0	0	0	0	0.2
s820	850	0	0	190	0	0	0	0	0.1	0	0	0	0	0	0.3
s832	856	14	0	200	0	0	0	0	0.2	0	0	0	0	0	0.3
s838	857	0	0	183	0	0	0	0	0.2	0	0	0	0	0	0.5
s953	1079	0	0	138	0	0	0	0	0.2	0	0	0	0	0	0.6
s1196	1242	0	0	227	0	0	0	0	0.4	0	0	0	0	0	0.9
s1238	1286	69	0	240	9	1	6	1	0.6	0	12	2	6	2	1.2
s1423	1501	14	0	135	0	0	0	0	0.2	0	0	0	0	0	0.6
s1488	1486	0	0	196	0	0	0	0	0.2	0	0	0	0	0	0.9
s1494	1494	12	0	191	0	0	0	0	0.3	0	0	0	0	0	0.9
s5378	4563	40	0	358	0	0	0	0	0.8	0	0	0	0	0	2.6
s9234	6475	452	0	660	37	0	2	12	7.0	0	43	2	2	14	13.8
s13207	9664	151	0	709	18	0	6	5	6.6	0	51	12	6	9	20.8
s15850	11336	389	0	643	0	0	0	0	7.0	0	0	0	0	0	23.9
s35932	35110	3984	0	129	0	0	0	0	6.4	0	128	1	0	0	74.5
s38417	31015	165	0	1458	0	0	0	0	15.9	0	0	0	0	0	46.6
s38584	34797	1506	0	989	0	0	0	0	14.1	0	0	0	0	0	40.9
Total	*181612*	*7364*	*0*	*9770*	*81*	*2*	*6*	*20*	*74.3*	*0*	*12907*	*32*	*6*	*1421*	*378.5*

Table 2.   Impact of each technique on test generation performance.

ATOM	With fault simulation			Without fault simulation		
	Abt	TB	Time	Abt	TB	Time
(w/All techniques)	0	81	74.3	0	12907	378.5
(w/o Improved unique sensitization)	13	1010	79.8	14	13992	369.0
(w/o Improved backtrace)	7	747	77.8	7	19601	400.8
(w/o Multiple PI assignments)	0	80	78.8	0	12992	415.7
(w/o LookBack)	0	81	74.3	909	7949	360.1
(w/o Support region)	0	81	203.2	0	12907	1874.7
(w/o Two phase test generation)	0	130	206.0	1	14481	1132.0

In order to assess the individual impact of each test generation technique on the overall ATPG performance, we carried out a more detailed experimental analysis and presented the results in Table 2. The Abt, TB and Time columns present the total number of aborted faults, the total number of backtracks and the total test generation time in seconds for all the ISCAS85 and ISCAS89 benchmark circuits. The first row in the table presents the performance results when all the test generation techniques proposed in this paper are used.

The other rows present the performance results when all the test generation techniques except the one indicated in the first column are used. The improved unique sensitization and improved backtrace contribute to both detecting hard-to-detect faults and reducing test generation time. Multiple primary input assignments contributes to reducing test generation time which is especially clear for without fault simulation case. During test generation, LookBack is triggered only after the targeted fault is aborted, and this only happened for the circuits c5315, c6288 and s13207 when fault simulation is not used. For these circuits 1, 906 and 2 faults are aborted respectively when lookback is not used. Support region and two phase test generation significantly reduce the test generation time both with and without fault simulation cases.

In summary, the results presented in Table 2 show the effectiveness of each technique in reducing both the number of backtracks and the test generation time and in detecting hard-to-detect faults that may otherwise require large number of backtracks and large execution times to be detected.

In order to accurately compare the performance of ATOM with all the previously published test generation systems, these programs should be executed on the same machine using the same benchmark circuits. Since, the programs for most of the previously published test generation algorithms are not publicly available, accurately reimplementing all of these algorithms is not feasible, and we don't have access to many of the machines on which the previous results are reported, it is very difficult to perform such a comparison. Since, Stephan et al. [14] already carried out such a through and accurate comparison of the performance of their test generation system, TEGUS which uses a satisfiability based test generation algorithm, with all the previously published results, we compared the performance of ATOM with TEGUS. Thus, we implicitly compared the performance of ATOM with all the previously published results.

Table 3 presents the comparison. The comparison is based on all the ISCAS85 and the full scan versions of all the ISCAS89 benchmark circuits. The experimental

Table 3.   Comparison of ATPG results.

Circuits	Number of faults targeted		With fault simulation Time/fault (ms)		Without fault simulation Time/fault (ms)	
	TEGUS	ATOM	TEGUS	ATOM	TEGUS	ATOM
ISCAS85	26815	32496	0.62	0.41	6.47	4.54
ISCAS89	128218	156480	0.39	0.39	1.51	1.48
ISCAS85+ISCAS89	155033	188976	0.43	0.39	2.37	2.0

results for both programs are obtained on the same machine, a 200 MHz Pentium Pro PC with 128MB RAM running Linux 2.0.0. Since, for both TEGUS and ATOM, the number of aborted faults for all the circuits both with and without using fault simulation was zero, we did not present the number of aborted faults in the table. In the table, the Number of Faults Targeted column presents the number of faults explicitly targeted by the corresponding test generation system. Since, in addition to fault equivalence, TEGUS also uses fault dominance relation, which is not used in ATOM, for fault collapsing, for a given circuit the number of faults explicitly targeted by TEGUS is smaller than that of ATOM. In the table, the Time/Fault column presents the average test generation time per fault in milliseconds. The execution times reported for both ATOM and TEGUS for without fault simulation case do not include the time for writing the generated test vectors to a file. Since the two test generators target different number of faults, we used time/fault measure to compare the performance of ATOM with TEGUS instead of overall test generation time. Time/fault value is obtained by dividing the total test generation time by the number of faults in the collapsed fault list.

Both systems demonstrated their robustness by detecting all the testable faults and proving all the redundant faults to be redundant both with and without using fault simulation. However, the experimental results show that ATOM is faster than TEGUS both with and without using fault simulation.

Stephan et al. [14] showed that TEGUS is faster than the previously published ATPG systems when all the faults are explicitly targeted, i.e. when fault simulation is not used. They also showed that TEGUS is faster than all but two of the previously published ATPG systems when random test generation and fault simulation is used. It is shown that in the latter case the ATPG systems reported in [22] and [10] are 10% faster than TEGUS. Since ATOM is 16% faster than TEGUS when no fault simulation is used and it is 10% faster than TEGUS when fault simulation is used, based on the results reported in [14] we can conclude that when both test generation and fault simulation performances are considered ATOM is one of the three fastest ATPG systems reported in the literature, whereas when only the test generation performance is considered ATOM is the fastest ATPG system reported in the literature.

5.   Conclusions

In this paper, we have introduced new efficient and robust structure based techniques for speeding up the deterministic test pattern generation for VLSI circuits. These techniques improve the average-case performance of the PODEM algorithm by reducing number of backtracks with a low computational cost. This is achieved by finding more necessary signal line assignments, by detecting conflicts earlier, and by avoiding unnecessary work during test generation.

To be able to assess the effectiveness of the proposed techniques, we have developed a deterministic ATPG system for combinational circuits, called ATOM, and incorporated these techniques into the test generator. The performance results for the ISCAS85 and full scan version of the ISCAS89 benchmark circuits demonstrated the effectiveness of these techniques on the test generation performance. ATOM detected all the testable faults and proved all the redundant faults to be redundant with very small number of backtracks in a short amount of time. Since these results are achieved without using an initial random test generation phase, and similar results are obtained both with and without fault simulation, the results demonstrated the robustness of the techniques we have developed.

In this paper, we evaluated the effectiveness of these new techniques for the combinational circuit test generation problem. The problem of test generation for sequential circuits is much more difficult than the combinational circuit test generation problem [23, 24]. In the sequential circuit test generation, each target fault must be excited, the fault effect must be propagated to a primary output, and the required state must be justified. Since, this process usually requires considering multiple time frames, it is much more time consuming than combinational circuit test generation. As a future research, we are planning to apply these new test generation techniques to the sequential circuit test pattern generation problem. Since combinational circuit test generation techniques are used for processing each time frame as part of a sequential circuit test generation algorithm, we believe that these techniques will also improve the performance of the structure based sequential circuit test generation algorithms.

References

1. P. Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits," *IEEE Trans. on Computers*, pp. 21–222, March 1981.
2. H. Fujiwara and S. Toida, "The Complexity of Fault Detection Problems for Combinational Logic Circuits," *IEEE Trans. on Computers*, pp. 555–560, June 1982.
3. M. Abramovici, J.J. Kulikowski, P.R. Menon, and D.T. Miller, "SMART and FAST: Test Generation for VLSI Scan-Design Circuits," *IEEE Design & Test of Computers*, pp. 43–54, Aug. 1986.

4. H. Fujiwara and T. Shimono, "On the Acceleration of Test Generation Algorithms," *IEEE Trans. on Computers*, pp. 1137–1144, Dec. 1983.

5. T. Kirkland and M.R. Mercer, "A Topological Search Algorithm for ATPG," *Proc. of the Design Automation Conf.*, June 1987, pp. 502–508.

6. W. Kunz and D. Pradhan, "Recursive Learning: An Attractive Alternative to the Decision Tree for Test Generation in Digital Circuits," *Proc. of the Int. Test Conf.*, Sept. 1992, pp. 816–825.

7. M.H. Schulz, E. Trischler, and T.M. Sarfert, "SOCRATES: A Highly Efficient Automatic Test Pattern Generation System," *IEEE Trans. on Computer-Aided Design*, pp. 126–137, Jan. 1988.

8. M.H. Schulz and E. Auth, "Improved Deterministic Test Pattern Generation with Applications to Redundancy Identification," *IEEE Trans. on Computer-Aided Design*, pp. 811–816, July 1989.

9. J.P.M. Silva and K.A. Sakallah, "Dynamic Search-Space Pruning Techniques in Path Sensitization," *Proc. of the Design Automation Conf.*, June 1994, pp. 705–711.

10. J.A. Waicukauski, P.A. Shupe, D.J. Giramma, and A. Matin, "ATPG for Ultra-Large Structured Designs," *Proc. of the Int. Test Conf.*, Aug. 1990, pp. 44–51.

11. S.T. Chakradhar, V.D. Agrawal, and S.G. Rothweiler, "A Transitive Closure Algorithm for Test Generation," *IEEE Trans. on Computer-Aided Design*, pp. 1015–1028, July 1993.

12. T. Larrabee, "Test Pattern Generation Using Boolean Satisfiability," *IEEE Trans. on Computer-Aided Design*, pp. 4–15, Jan. 1992.

13. J.P.M. Silva and K.A. Sakallah, "Robust Search Algorithms for Test Pattern Generation," *Proc. of the Fault-Tolerant Computing Symp.*, June 1997, pp. 152–161.

14. P. Stephan, R.K. Brayton and A.L. Sagiovanni-Vincentelli, "Combinational Test Generation Using Satisfiability," *IEEE Trans. on Computer-Aided Design*, pp. 1167–1176, Sept. 1996.

15. F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Designs and a Special Translator in Fortran," *Proc. of the Int. Symp. on Circuits and Systems*, June 1985.

16. F. Brglez, D. Bryan, and K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuits," *Proc. of the Int. Symp. on Circuits and Systems*, May 1989, pp. 1929–1934.

17. T.M. Niermann, W. Cheng, and J.H. Patel, "PROOFS: A Fast, Memory-Efficient Sequential Circuit Fault Simulator," *IEEE Trans. on Computer-Aided Design*, pp. 198–207, Feb. 1992.

18. S.J. Chandra and J.H. Patel, "Experimental Evaluation of Testability Measures for Test Generation," *IEEE Trans. on Computer-Aided Design*, pp. 93–97, Jan. 1989.

19. L.H. Goldstein and E.L. Thigpen, "SCOAP: Sandia Controllability/Observability Analysis Program," *Proc. of the Design Automation Conf.*, 1980, pp. 190–196.

20. S. Kundu, L.M. Huisman, I. Nair, V. Iyengar, and L.N. Reddy, "A Small Test Generator for Large Designs," *Proc. of the Int. Test Conf.*, Sept. 1992, pp. 30–40.

21. R.P. Kunda, P. Narain, J.A. Abraham, and B.D. Rathi, "Speed Up of Test Generation Using High-Level Primitives," *Proc. of the Design Automation Conf.*, June 1990, pp. 594–599.

22. Y. Matsunaga and M. Fujita, "A Fast Test Pattern Generator for Large Scale Circuits," *Proc. of the Synth. Simulation Meeting Int. Interchange*, April 1992, pp. 263–271.

23. K.T. Cheng, "Gate-Level Test Generation for Sequential Circuits," *ACM Trans. on Design Automation*, pp. 405–442, Oct. 1996.

24. T.M. Niermann and J.H. Patel, "HITEC: A Test Generation Package for Sequential Circuits," *Proc. of the European Design Automation Conf.*, Feb. 1991, pp. 214–218.

Ilker Hamzaoglu was born in Trabzon, Turkey. He received the B.S. and M.S. degrees in Computer Engineering from Bogazici University, Istanbul, Turkey. He received the NATO Science Fellowship Award from the Turkish Scientific and Technical Research Council. He is currently with the University of Illinois at Urbana-Champaign, where he is a Ph.D. student at the Computer Science Department and a Research Assistant at the Center for Reliable and High Performance Computing. He received the W.J. Poppelbaum Memorial Award from the Computer Science Department. His research interests include computer-aided design and testing of digital systems, computer architecture and parallel processing.

Janak H. Patel is with the University of Illinois at Urbana-Champaign, where he is currently a Co-Director of Center for Reliable and High Performance Computing and a Professor of Electrical and Computer Engineering and Computer Science, and a Research Professor with the Coordinated Science Laboratory. He is a co-founder of Sunrise Test Systems, an ATG and testability software company. He has also provided consulting and tutorial services to the industry on design and test. He is currently engaged in teaching, research, and consulting in the areas of automatic test generation, design for testability, fault simulation and diagnosis.

Dr. Patel was born in Bhavnagar, India. He received the B.Sc. degree in Physics from Gujrat University, India. He also received the B.Tech. degree from Indian Institute of Technology, Madras, India, and M.S. and Ph.D. degrees from Stanford University, Stanford, CA, all in electrical engineering. He is a Fellow of IEEE and a recipient of the 1998 the IEEE Piore Award. He formerly served on the editorial board of the Journal of Electronic Testing: Theory and Applications.