International Conference on Computer-Aided Design (ICCAD), November 2000

Deterministic Test Pattern Generation Techniques for Sequential Circuits *

Ilker Hamzaoglu† and Janak H. Patel
Center for Reliable & High-Performance Computing
University of Illinois, Urbana, IL 61801

Abstract

This paper presents new test generation techniques for improving the average-case performance of the iterative logic array based deterministic sequential circuit test generation algorithms. To be able to assess the effectiveness of the proposed techniques, we have developed a new ATPG system for sequential circuits, called ATOMS, and we have incorporated these techniques into the test generator. ATOMS achieved very high fault coverages in a short amount of time for the ISCAS89 sequential benchmark circuits, demonstrating the effectiveness of these techniques on the test generation performance.

1 Introduction

Although scan based design for testability techniques can convert a sequential circuit into a combinational circuit for testing purposes, in some cases, the cost of full scan can be prohibitive in both area overhead and performance degradation. Therefore, efficient sequential circuit test generation algorithms are very important for producing high quality VLSI circuits.

The problem of test generation for sequential circuits is much more complex than the combinational circuit test generation problem [21]. Although, significant progress has been made towards solving the sequential circuit test generation problem [2], it is still a very time consuming process. Most of the sequential circuit test generation algorithms proposed in the literature use an *iterative logic array (ILA)* model of a sequential circuit for test generation purposes. These algorithms can be classified into three major categories: deterministic [3, 4, 5, 6, 7, 9, 15, 16, 17, 20, 22, 24, 31], simulation-based [13, 14, 23, 26, 28, 29], and hybrid [12, 18, 19, 27, 30]. Although these algorithms are successful in generating high quality test patterns, their test generation times are often very high.

Therefore, in this paper, we extend the deterministic test generation techniques we proposed in [10, 11] to sequential circuits and we propose new structure-based techniques for speeding up the deterministic test pattern generation for sequential circuits. The new sequential circuit test generation techniques are state space reduction and unjustifiable state identification. These techniques improve the average-case performance of the iterative logic array based deterministic sequential circuit test generation algorithms. We expect that these techniques will also improve the performance of the hybrid sequential circuit test generation systems, e.g. [18, 19], that achieve the best published test generation results by combining the deterministic and simulation-based test generation techniques.

To be able to assess the effectiveness of the proposed techniques, we have developed a new ATPG system for sequential circuits, called ATOMS, by using ATOM [10, 11] as the underlying combinational circuit test generation system, and we have incorporated these techniques into the test generator. In addition to the techniques we have proposed, the test generator also employs some of the test generation techniques introduced in the literature. The experimental results for the ISCAS89 sequential benchmark circuits showed that ATOMS achieves very high fault coverages in a short amount of time. This demonstrates the effectiveness of these techniques on the test generation performance.

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

2 Test Generation System

We have developed a new ATPG system for sequential circuits, called ATOMS, by using ATOM [10, 11] as the underlying combinational circuit test generation system. ATOMS is designed in an object-oriented style and implemented in C++.

The fault simulator uses the PROOFS fault simulation algorithm [25]. The test generator is based on the HITEC sequential circuit test generation algorithm [24], and it uses the PODEM algorithm [8] for processing each time frame. The test generator uses the single stuck-at fault model and the

*This research was supported in part by the Semiconductor Research Corporation under contract SRC 99-TJ-717 and in part by DARPA under contract DABT63-95-C-0069.

†Ilker Hamzaoglu is currently with Motorola Labs, Schaumburg, IL.

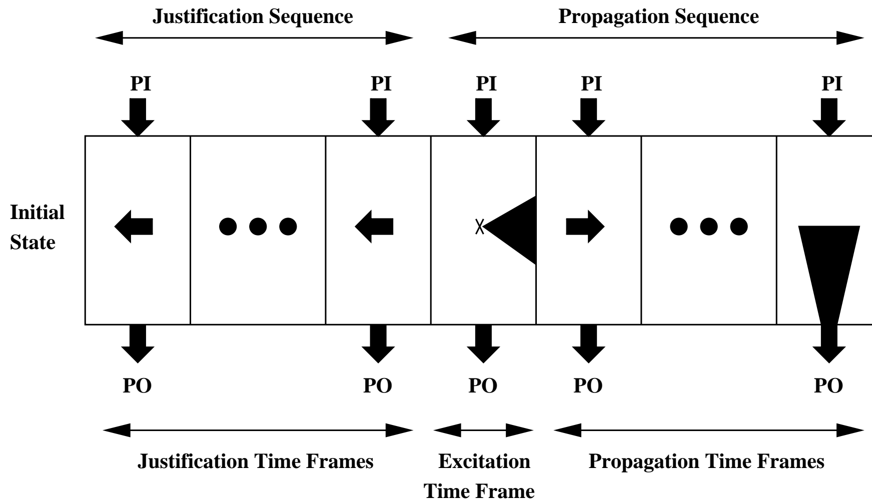

Figure 1: Sequential Circuit Test Generation Algorithm

nine-valued logic. We have incorporated the new test generation techniques that we have developed into the test generator. In addition to these techniques, the test generator also uses the state relaxation [24], unjustifiable state storage for each fault [24] and cycle detection during state justification [24] techniques proposed in the literature.

As illustrated in Figure 1, in order to generate a test sequence for a given fault, the sequential circuit test generation algorithm uses both forward time processing and reverse time processing on the iterative logic array model of the given sequential circuit. The time frame in which the fault is excited is called the *excitation time frame*, the time frames used to propagate the fault effect to a primary output are called *propagation time frames*, and the time frames used to justify the required state are called *justification time frames*. The state of the flip-flops in the excitation time frame is called the *excitation state*. The initial part of the test sequence that brings the sequential circuit from the fully unspecified initial state to the excitation state is called the *justification sequence*. The remaining part of the test sequence that excites the fault and propagates the fault effect to a primary output is called the *propagation sequence*.

3 Test Generation Techniques

We will now describe the extension of the test generation techniques that we proposed for combinational circuits to sequential circuits and the new deterministic test generation techniques that we propose for sequential circuits.

1. Improved Unique Sensitization:

Since the improved unique sensitization technique is useful for identifying necessary logic assignments due to fault effect propagation constraints, we use this technique only during the fault effect propagation phase of the sequential circuit test generation algorithm. We use the improved unique sensitization technique to identify the necessary assignments for propagating the fault effect either to a primary output or to a flip-flop within a single time frame. As in the case of combinational circuits, the improved unique sensitization technique discovers more necessary logic assignments than the previous techniques, thus speeding up the test pattern generation by identifying conflicts earlier.

2. Improved Backtrace with X-path and Conflict Check:

The backtrace operation is used in both fault effect propagation and state justification phases of the sequential circuit test generation algorithm to find appropriate primary input and flip-flop assignments that will justify the fault effect propagation and state justification objectives. Since the X-path check technique is useful only for fault effect propagation, we use the backtrace with X-path check technique only during the fault effect propagation phase. If more than one time frames are expanded during fault effect propagation, we only use this technique in the time frame where the fault effect that is selected from the D-frontier to be propagated to a primary output is present. We use the backtrace with conflict check technique in both phases of the sequential test generation algorithm. In the fault effect propagation phase, for a given OR decision point, we only compute the implications in the same time frame with the OR decision point to identify the possible conflicts within a single time frame. Since it may potentially be a costly operation, we do not compute the implications over time frames. Since state justification proceeds one time frame at a time, in the state justification phase, we also compute the implications in a single time frame.

As in the case of combinational circuits, the improved backtrace with X-path and conflict check technique helps the test generator to determine the possible conflicts before committing to a primary input or a flip-flop assignment, thus preventing the test generator from going into a nonsolution area in the search space, which might result in a significant number of backtracks.

3. Multiple Primary Input and Flip-Flop Assignments:
We use the multiple primary input assignments technique in both phases of the sequential test generation algorithm in order to assign logic values to more than one primary input or flip-flop at the end of a backtrace. In the fault effect propagation phase, we check for possible multiple primary input assignments in the propagation time frames, and we check for possible multiple primary input or flip-flop assignments in the excitation time frame. Since state justification proceeds one time frame at a time, in the state justification phase, we check for possible multiple primary input assignments in the current time frame. Similar to the combinational circuit case, multiple primary input assignments are achieved by exploiting the backtrace-stack constructed during the backtrace procedure, and it speeds up the sequential test pattern generation by reducing the number of backtraces in both fault effect propagation and state justification phases.

4. LookBack:
In [10, 11], we introduced the lookback search technique for the combinational circuit test generation problem. In this paper, we extend it to sequential circuits, and we use the extended lookback technique during the state justification phase of the sequential circuit test generation algorithm.

In the state justification phase, the test generator tries to find an input sequence that can bring the sequential circuit from the fully unspecified initial state to the excitation state. This is achieved by traversing the state transition graph of the sequential circuit in the backward direction starting from the excitation state. The conventional branch and bound search algorithm used for sequential circuit test generation traverses the state transition graph in a depth-first fashion in an attempt to explore it entirely until either a justification sequence is found or the excitation state is proven to be unjustifiable or a maximum backtrack limit is reached.

If the test generator reaches the maximum backtrack limit before finding a justification sequence for the excitation state, the lookback technique is used to find a justification sequence. The reason for the failure of the test generator in finding a justification sequence within the maximum backtrack limit is that there was no solution in the part of the search space, i.e., the state transition graph, that it explored. This indicates that the depth-first search technique used to traverse the state transition graph was unable to bring the test generator into a solution area in the search space. The lookback technique tries to overcome this problem by traversing the state transition graph using a combination of depth-first and breadth-first search techniques.

The lookback procedure takes three input parameters: the partial state transition graph explored by the test generator before the lookback procedure is called, a local state-backtrack limit and a global state-backtrack limit. For each state visited during the state transition graph traversal, the local state-backtrack limit determines the size of the search space that will be explored in the backward direction starting from this state using depth-first traversal. In other words, for a given state S_i and a local state-backtrack limit n, if no solution is found after traversing n predecessors of S_i using depth-first search, the test generator stops traversing the predecessors of S_i. It rather jumps back to its successor in the partial state transition graph and continues traversing the predecessors of this state using depth-first traversal until the local state-backtrack limit is reached for this state. This process is repeated for each state in the partial state transition graph until either a justification sequence is found or the limited depth first search is tried for each state or the global state-backtrack limit is reached.

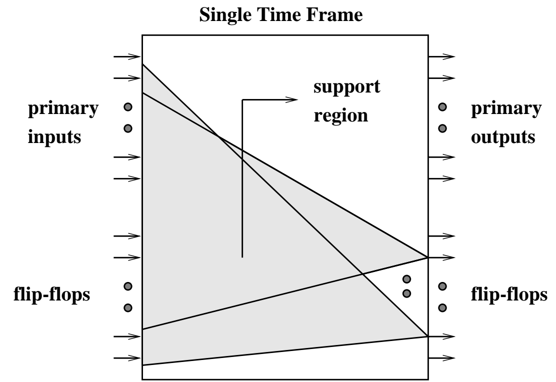

Figure 2: Support Region of a State

In summary, for each state visited during the state justification phase, the lookback technique carries out only a limited depth-first traversal in the backward direction starting from this state. Thus, it uses a combination of depth-first and breadth-first search techniques for traversing the state transition graph. Therefore, the lookback technique is effective in situations where depth-first search, used in the conventional branch and bound test generation algorithm, is unable to bring the test generator into a solution area in a reasonable amount of time.

Since lookback is not a complete search algorithm, if it fails to find a justification sequence, the test generator continues to search the rest of the search space that has not yet been explored using the conventional branch and bound search algorithm in order to either find a justification sequence or prove that the excitation state is unjustifiable.

5. Support Region:
In [10, 11], we defined the *support region* of a fault in a combinational circuit as the part of the circuit that may affect the test generation process for this fault. In this paper, we extend the definition of support region for a state of a sequential circuit, and we use the support region technique during the state justification phase of the sequential circuit test generation algorithm.

We define the *support region* of a state of a sequential circuit as the part of the circuit that may affect the state justification process for this state in a single time frame. As illustrated in Figure 2, for a given state, this region consists of the

gates in the fanin cones of all the flip-flops that have a known logic value in this state, i.e., either 0 or 1.

Since the gates outside the support region of a state do not fanout to the flip-flops that have a known logic value in this state, they do not affect the state justification process for this state in the current time frame. 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 state justification in a time frame, it is not necessary to carry out forward implications outside the support region of the state that is justified. Thus, the support region technique avoids a considerable amount of unnecessary work during state justification.

The cost of computing the support region of a given state is very small, because we compute the fanin cone of each flip-flop as a preprocessing step before test generation. During test generation, we compute the support region of a state by simply computing the union of the fanin cones of all the flip-flops that have a known logic value in this state.

6. State Space Reduction:

Iterative logic array based deterministic sequential circuit test generation algorithms work on an iterative logic array model of a sequential circuit for test generation purposes. This model represents a sequential circuit as an array of identical time frames. Each time frame represents the state of the sequential circuit at a different point in time. Since a given fault is present in every time frame, deterministic sequential circuit test generation algorithms use nine-valued logic and they search the state space of $X/X, 0/X, 1/X, X/0, X/1, 0/1, 1/0, 0/0, 1/1$ logic value assignments for each flip-flop in every time frame.

We propose a new technique, based on the following theorem, for reducing the size of the search space that needs to be explored by a deterministic sequential circuit test generator. This technique reduces the search space by avoiding to consider the $0/1$ or $1/0$ logic value assignments for any flip-flop in the excitation time frame.

Theorem 1 *If a fault in a sequential circuit is testable, there exists a test sequence that detects this fault without assigning opposite good and faulty logic values, i.e.,* $0/1$ *or* $1/0$*, to any flip-flop in the excitation time frame.*

Proof. If the assignment of $0/1$ or $1/0$ logic value to a flip-flop in the excitation time frame is required to generate a test sequence for a fault in a sequential circuit, then this fault is combinationally redundant, because $0/1$ or $1/0$ logic value cannot be applied to a primary input for testing a combinational circuit. Since a combinationally redundant fault is sequentially untestable, there exists no test sequence that can detect this fault. Therefore, if a fault in a sequential circuit is testable, there exists a test sequence that detects this fault without assigning opposite good and faulty logic values, i.e., $0/1$ or $1/0$, to any flip-flop in the excitation time frame. ■

Theorem 1 indicates that during sequential circuit test generation it is not necessary to consider the $0/1$ or $1/0$ logic value pairs for any flip-flop in the excitation time frame. In other words, to generate a test sequence for a testable fault or to prove that a fault is untestable, it is sufficient to search the state space of $X/X, 0/X, 1/X, X/0, X/1, 0/0, 1/1$ logic value assignments for each flip-flop in the excitation time frame, because if a fault is testable, there is a solution for the test generation problem in this region of the state space. On the other hand, if the fault is untestable, there is no solution for the test generation problem in this region of the state space. Thus, the state space reduction technique avoids considerable amount of unnecessary search during test generation by reducing the state space that needs to be explored for a given fault.

7. Unjustifiable State Identification:

A state S_i of a sequential circuit is said to be *unjustifiable* if there exists no input sequence that can bring the sequential circuit from the fully unspecified initial state to state S_i. If, on the other hand, there is such an input sequence, S_i is said to be *justifiable*. The identification of unjustifiable states is very useful for sequential circuit test generation, since the knowledge of unjustifiable states prevents the test generator from going into a nonsolution area in the search space [16, 24].

We propose a new low-cost unjustifiable state identification technique based on the following observation. If an unjustifiable state S_u can be reached from a state S_k, then S_k is an unjustifiable state. Based on this observation, the new technique discovers unjustifiable states as follows. Given an unjustifiable state S_u, it traverses the state transition graph of the sequential circuit in the backward direction starting from S_u. The state transition graph can be traversed by using either the depth-first or breadth-first search technique. All the states visited during this traversal are declared to be unjustifiable. However, since the number of states from which S_u is reachable can be very large, it may not be possible to traverse all of these states due to execution time and storage space limitations. Therefore, the traversal can be restricted to visit only a predetermined number of states.

This is a low-cost technique for identifying unjustifiable states which may otherwise be very difficult to prove as unjustifiable. Since this technique is useful for discovering new unjustifiable states using existing unjustifiable states, it can be used for the industrial circuits for which a number of functionally unjustifiable states are known before test generation. The knowledge of these additional unjustifiable states speeds up the sequential circuit test pattern generation process by avoiding unnecessary search during the state justification phase.

4 Experimental Results

ATOMS was tested on the ISCAS89 sequential benchmark circuits on a 200 MHz Pentium Pro PC with 128MB RAM running Linux 2.0.0 using GNU CC version 2.8.0. In all the experiments, a maximum backtrack limit of 75 and a maximum time frame expansion limit of 200 are used in the

Circuit	ATOMS				HITEC			
	Det	Unt	Vect	Time	Det	Unt	Vect	Time
s298	265	30	312	5.5s	265	26	322	16.2m
s344	329	11	127	24.9s	324	11	115	8.1m
s349	335	13	118	23.9s	332	13	128	7.7m
s382	364	18	5169	2.9m	301	9	1463	1.5h
s386	314	70	328	3.2s	314	70	286	7.3s
s400	384	27	3868	3.2m	341	17	1845	1.2h
s444	424	29	4433	5.5m	373	25	1761	1.5h
s510	0	564	0	1.0s	–	–	–	–
s526	451	18	6552	29.0m	316	23	436	5.8h
s641	404	63	184	2.0s	404	63	209	4.8s
s713	476	105	172	2.4s	476	105	173	6.7s
s820	814	36	997	19.9s	813	37	1115	3.5m
s832	818	52	998	19.4s	817	53	1137	5.8m
s953	89	990	13	6.2s	–	–	–	–
s1196	1239	3	373	2.6s	1239	3	435	5.5s
s1238	1283	72	409	3.2s	1283	72	475	8.2s
s1423	1095	10	570	40.6m	723	14	150	13.9h
s1488	1444	42	1101	1.1m	1444	41	1170	16.5m
s1494	1453	53	1162	36.7s	1453	52	1245	9.6m
s5378	3379	148	839	34.2m	3231	217	912	18.4h
s35932	34,908	3984	272	1.8h	34,901	3984	496	4.7h

Table 1: Comparison of Sequential ATPG Results

fault effect propagation phase. Since the state justification is a computationally expensive process, in the state justification phase, we used a smaller backtrack limit and a smaller time frame expansion limit for the larger circuits, and we did not use the lookback technique for the s35932 benchmark circuit. In particular, a maximum backtrack limit of 3000 and a maximum time frame expansion limit of 2000 are used for the circuits with less than 750 gates, whereas a maximum backtrack limit of 30 and a maximum time frame expansion limit of 1000 are used for the circuits with more than 750 gates. Since we were not aware of any functionally unjustifiable states for the ISCAS89 circuits, we did not use the unjustifiable state identification technique in these experiments.

We compared the performance of ATOMS with HITEC [24], one of the best deterministic sequential ATPG system reported in the literature. Table 1 presents the comparison. The columns in the table present the number of faults detected, the number of faults proven to be untestable, the number of test vectors generated and the total test generation time, respectively. The execution times for ATOMS include the time for computing the dominators and the static global implications, the fault simulation time and the input/output time. The "–" sign in the table indicates that the corresponding result for this circuit is not reported. The performance results for ATOMS were obtained on a 200 MHz Pentium Pro PC with 128MB RAM running Linux 2.0.0 using GNU CC version 2.8.0. The performance results for HITEC were obtained on an HP 9000 J200 workstation with 256 MB RAM. Although, it is not possible to exactly compare the test generation times, the performances of these machines on SpecInt95 benchmarks can be used to approximately compare the test generation times: 8.09 for 200 MHz Pentium Pro and 4.98 for HP 9000 J200.

The results in the table show that ATOMS achieved higher fault coverages in a much smaller amount of time than HITEC for all the circuits. This demonstrates the effectiveness of the new test generation techniques on improving the average-case performance of the iterative logic array based deterministic sequential circuit test generation algorithms. We expect that these techniques will also improve the performance of the hybrid sequential circuit test generation systems, e.g. [18, 19], that achieve the best published test generation results by combining the deterministic and simulation-based test generation techniques.

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 columns in the table present the total number of detected faults, the total number of faults proven to be untestable, the total number of aborted faults, and the total test generation time in seconds for all the ISCAS89 sequential benchmark circuits. The first row in the table presents the performance results when all the test generation techniques are used. The other rows present the performance results when all the test generation techniques except the ones indicated in the first column are used.

The results show that the improved unique sensitization, improved backtrace and multiple primary input assignments techniques are quite effective in both detecting hard-to-detect faults and reducing test generation time. Since the lookback technique is used after the test generator aborts the state justification process for a given fault, it spends additional effort to generate a test sequence for this fault. Therefore, in order to do a fair comparison, when the lookback technique is not used we increased the justification backtrack limit to 3 times and 6 times the normal justification backtrack limit. The results show that without using the lookback technique the test generator is unable to detect the same number of faults in

ATOMS	Det	Unt	Abt	Time (s)
(w/ All Techniques)	50,268	6338	1844	13,769
(w/o Improved Unique Sensitization, Improved Backtrace, and Multiple PI Assignments)	50,177	6338	1935	20,366
(w/o LookBack, 3 times Backtrack Limit)	49,896	6350	2204	10,749
(w/o LookBack, 6 times Backtrack Limit)	49,918	6375	2157	13,011
(w/o Support Region)	50,268	6338	1844	20,270

Table 2: Impact of Each Technique on Sequential Circuit Test Generation Performance

the same amount of test generation time. Finally, the results show that the support region technique significantly reduces the test generation time.

5 Conclusions

In this paper, we presented new deterministic test pattern generation techniques for sequential circuits. These techniques improve the average-case performance of the iterative logic array based deterministic sequential circuit test generation algorithms. To be able to assess the effectiveness of the proposed techniques, we have developed a new ATPG system for sequential circuits, called ATOMS, and we have incorporated these techniques into the test generator. ATOMS achieved very high fault coverages in a short amount of time for the ISCAS89 sequential benchmark circuits, demonstrating the effectiveness of these techniques on the test generation performance.

References

[1] F. Brglez, D. Bryan, and K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuits," *Proc. Int. Symp. on Circuits and Systems*, pp. 1929-1934, May 1989.

[2] K. T. Cheng, "Gate-Level Test Generation for Sequential Circuits," *ACM Trans. on Design Automation*, vol. 1, no. 4, pp. 405-442, October 1996.

[3] W. T. Cheng, "The BACK Algorithm for Sequential Test Generation," *Proc. Int. Conf. on Computer Design*, pp. 66-69, October 1988.

[4] W. T. Cheng and T. J. Chakraborty, "Gentest: An Automatic Test Generation System for Sequential Circuits," *IEEE Computer*, pp. 43-49, April 1989.

[5] H. Cho, G. D. Hachtel, and F. Somenzi, "Redundancy Identification/Removal and Test Generation for Sequential Circuits Using Implicit State Enumeration," *IEEE Trans. on Computer-Aided Design*, vol. 12, no. 7, pp. 935-945, July 1993.

[6] A. Dargelas, C. Gauthron, and Y. Bertrand, "Mosaic: A Multiple-Strategy Oriented Sequential ATPG for Integrated Circuits," *Proc. European Design and Test Conf.*, pp. 29-36, March 1997.

[7] A. Ghosh, S. Devadas, and A. R. Newton, "Test Generation and Verification for Highly Sequential Circuits," *IEEE Trans. on Computer-Aided Design*, vol. 10, no. 5, pp. 652-667, May 1991.

[8] P. Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits," *IEEE Trans. on Computers*, vol. C-30, no. 3, pp. 21-222, March 1981.

[9] N. Gouders and R. Kaibel, "Test Generation Techniques for Sequential Circuits," *Proc. IEEE VLSI Test Symp.*, pp. 221-226, April 1991.

[10] I. Hamzaoglu and J. H. Patel, "New Techniques for Deterministic Test Pattern Generation," *Proc. IEEE VLSI Test Symp.*, pp. 446-452, April 1998.

[11] I. Hamzaoglu and J. H. Patel, "New Techniques for Deterministic Test Pattern Generation," *Journal of Electronic Testing: Theory and Applications*, vol. 15, no. 1/2, pp. 63-73, October 1999.

[12] M. S. Hsiao, E. M. Rudnick, and J. H. Patel, "Alternating Strategies for Sequential Circuit ATPG," *Proc. European Design and Test Conf.*, pp. 368-374, March 1996.

[13] M. S. Hsiao, E. M. Rudnick, and J. H. Patel, "Sequential Circuit Test Generation Using Dynamic State Traversal," *Proc. European Design and Test Conf.*, pp. 22-28, February 1997.

[14] M. S. Hsiao, E. M. Rudnick, and J. H. Patel, "Application of Genetically-Engineered Finite-State-Machine Sequences to Sequential Circuit ATPG," *IEEE Trans. on Computer-Aided Design*, vol. 17, no. 3, pp. 239-254, March 1998.

[15] T. P. Kelsey and K. K. Saluja, "Fast Test Generation for Sequential Circuits," *Proc. Int. Conf. on Computer-Aided Design*, pp. 354-357, November 1989.

[16] M. H. Konijnenburg, J. Th. van der Linden, and A. J. van de Goor, "Illegal State Space Identification for Sequential Circuit Test Generation," *Proc. Design, Automation, and Test in Europe Conf.*, pp. 741-746, March 1999.

[17] D. H. Lee and S. M. Reddy, "A New Test Generation Method for Sequential Circuits," *Proc. Int. Conf. on Computer-Aided Design*, pp. 446-449, November 1991.

[18] X. Lin, I. Pomeranz, and S. M. Reddy, "MIX: A Test Generation System for Synchronous Sequential Circuits," *Proc. Int. Conf. on VLSI Design*, pp. 456-463, January 1998.

[19] X. Lin, I. Pomeranz, and S. M. Reddy, "Techniques for Improving the Efficiency of Sequential Circuit Test Generation," *Proc. Int. Conf. on Computer-Aided Design*, pp. 147-151, November 1999.

[20] H. T. Ma, S. Devadas, A. R. Newton, and A. Sangiovanni-Vincentelli, "Test Generation for Sequential Circuits," *IEEE Trans. on Computer-Aided Design*, vol. 7, no. 10, pp. 1081-1093, October 1988.

[21] T. E. Marchok, Aiman El-Maleh, W. Maly and J. Rajski, "Complexity of Sequential ATPG," *Proc. European Design and Test Conf.*, pp. 252-261, February 1995.

[22] R. Marlett, "An Effective Test Generation System for Sequential Circuits," *Proc. Design Automation Conf.*, pp. 250-256, June 1986.

[23] P. Mazumder and E. M. Rudnick. *Genetic Algorithms for VLSI Design, Layout and Test Automation*. Prentice Hall, New Jersey, 1999.

[24] T. M. Niermann and J. H. Patel, "HITEC: A Test Generation Package for Sequential Circuits," *Proc. European Design Automation Conf.*, pp. 214-218, February 1991.

[25] T. M. Niermann, W. Cheng, and J. H. Patel, "PROOFS: A Fast, Memory-Efficient Sequential Circuit Fault Simulator," *IEEE Trans. on Computer-Aided Design*, vol. 11, no. 2, pp. 198-207, February 1992.

[26] P. Prinetto, M. Rebaudengo, and M. Sonza Reorda, "An Automatic Test Pattern Generator for Large Sequential Circuits Based on Genetic Algorithms," *Proc. Int. Test Conf.*, pp. 240-249, October 1994.

[27] E. M. Rudnick and J. H. Patel, "Combining Deterministic and Genetic Approaches for Sequential Circuit Test Generation," *Proc. Design Automation Conf.*, pp. 183-188, June 1995.

[28] E. M. Rudnick, J. H. Patel, G. S. Greenstein, and T. M. Niermann, "A Genetic Algorithm Framework for Test Generation," *IEEE Trans. on Computer-Aided Design*, vol. 16, no. 9, pp. 1034-1044, September 1997.

[29] D. G. Saab, Y. G. Saab, and J. A. Abraham, "CRIS: A Test Cultivation Program for Sequential VLSI Circuits," *Proc. Int. Conf. on Computer-Aided Design*, pp. 216-219, November 1992.

[30] D. G. Saab, Y. G. Saab, and J. A. Abraham, "Iterative Simulation-Based Genetics + Deterministic Techniques = Complete ATPG," *Proc. Int. Conf. on Computer-Aided Design*, pp. 40-43, November 1994.

[31] M. H. Schulz and E. Auth, "Essential: An Efficient Self-Learning Test Pattern Generation Algorithm for Sequential Circuits," *Proc. Int. Test Conf.*, pp. 28-37, August 1989.