Combining GAs and Symbolic Methods for High Quality Tests of Sequential Circuits

Martin Keim Nicole Drechsler Bernd Becker
Institute of Computer Science, Albert-Ludwigs-University
Am Flughafen 17, 79110 Freiburg im Breisgau, Germany
email: <keim, ndrechsler, becker>@informatik.uni-freiburg.de

Abstract

A symbolic fault simulator is integrated in a Genetic Algorithm (GA) environment to perform Automatic Test Pattern Generation (ATPG) for synchronous sequential circuits. In a two phase algorithm test length and fault coverage as well are optimized. However, there are circuits with bad random testability properties, that are also hard to test using genetically optimized test patterns. Thus, deterministic aspects are included in the GA environment to improve fault coverage. Experiments demonstrate that tests with higher fault coverages and considerably shorter test sequences than in previously presented approaches are obtained.

1 Introduction

The complex and time consuming task of *Automatic Test Pattern Generation* (ATPG) for synchronous sequential circuits is of large interest in computer aided design for integrated circuits. Primarily two methodologies have been studied intensively: Deterministic algorithms, e.g. [19], and simulation based algorithms, e.g. [25]. Deterministic ATPG has the advantage that it finds a test sequence for a target fault if there exists one, or proves the fault as undetectable. However, complex calculations are necessary for the task. On the other hand side, simulation based ATPG algorithms cannot guarantee to find a test sequence, nor can they prove a fault as undetectable. But the complexity of computation is considerably reduced. The quality of the test sequence computed by these algorithms is determined by the quality of the test sequence optimization algorithms. Such an optimization paradigm is provided by *Genetic Algorithms* (GAs) [8]. GAs have been demonstrated to be an interesting and competitive alternative in ATPG, see e.g. [25, 24, 4, 3, 23, 22, 9, 10]. In general, GAs provide optimization and search strategies which are used more and more in the area of VLSI CAD [7].

In this paper we propose an ATPG algorithm based on simulation and a GA. We measure the quality of a solution by means of fault coverage and test pattern sequence length. The main focus of the GA is quality of the tests at the expense of a reasonable amount of runtime. The idea is to split the ATPG into two runs: We use a "fast but inexact" three-valued based simulation during the first run to obtain tests for easy to detect faults. Thereafter an "exact" simulation based on *ordered Binary Decision Diagrams* (BDDs) [2] is used during a second run to determine tests for the remaining "hard" faults. A probabilistic and a deterministic ATPG subroutine is implemented to find a test sequence for a target fault. Thereafter this sequence is genetically optimized to increase the coverage of other faults.

The tool supports not only the common SOT *(Single Observation Time Test Strategy)*, but also the *Multiple Observation Time Test Strategy* (MOT) [20]. The experimental results show that MOT further improves the number of detected faults in particular in comparison to symbolic SOT. Details on symbolic simulation using MOT can be found in [17]. Restricting MOT to only those outputs of the fault-free circuit that are known, there is only one partially defined sequence, that has to be compared with the output sequence of the circuit-under-test. This rMOT test strategy is able to improve fault coverage and can be used for test execution without modifying the test execution process itself.

We describe experimental results to show the efficiency of the approach in comparison to previous work. We also show the effect of using a three-valued simulation or a symbolic simulation method. We compare our method to several advanced ATPG algorithms found in literature which do not make use of a given fault free reset state or a full scan environment. In general, the test sequence length achieved with the approach proposed here is only 2/9 of that of other ATPGs for SOT, and 1/2 for MOT.

2 The ATPG Tool

To solve the ATPG problem we combined a genetic algorithm with an accurate fault simulator. In the following, we introduce the components of our ATPG tool.

2.1 Hybrid Fault Simulator

The simulator *HFSim* [16, 17] used here supports different logics for simulation: (1) a fault simulation based upon the three-valued logic, (2) a symbolic fault simulation based upon BDDs, and (3) a combined fault simulation, with a symbolic true value simulation and a three-valued based explicit fault simulation. Thus, the simulator is able to automatically adapt itself to the memory demand and execution time restrictions. It automatically selects an appropriate procedure for the next time frame. Additionally, it supports the SOT, MOT, and rMOT test strategy.

2.2 GA Aspects

Our ATPG algorithm makes use of two main GA phases. In *Phase 1*, the algorithm starts optimizing a test sequence of fixed length, beginning in a given starting state, e.g. the unknown initial state. This test sequence is optimized with respect to fault coverage and length of the test sequence. In *Phase 2* tests are generated for the remaining undetected faults, i.e. the test sequence from *Phase 1* is iteratively enlarged by test vectors.

An element (*individual*) of a GA's population is a binary string, that represents a test sequence of n time frames. The maximum length of the individuals is fixed, i.e. each test sequence in a population has an upper bound in length. Initially, these binary strings are randomly chosen. The objective function rate the number of undetected faults as a first and the length of the test sequence as a second optimization criterion. Thus, the value of the objective function has to be minimized. This objective function consists basically of the fault simulatior *HFSim* described in the last section. The performance of the GA with respect to runtime is improved by using *tournament selection* [6] which reduces the number of evaluations tremendously. (For comparison we used *roulette wheel selection* [6] in another series of experiments).

Genetic operators modify the individuals in the populations. Here, we make use of "standard" operators [14, 13], like mutations or crossover, and several problem specific operators (*horizontal crossover* [5], *vertical crossover* [5], and *free vertical crossover*). The size of the population in *Phase 1* is set to $|\mathcal{P}| = 32$ and in *Phase 2* it is set to $|\mathcal{P}| = 16$. If *Phase 2* runs in the three-valued mode the population size is enlarged to $|\mathcal{P}| = 32$, since the three-valued simulation runs much faster. *Phase 2* is started if *Phase 1* has had no improvement during the last 100 generations. Each recombination operator is performed with a probability of 20% and the mutation rate is set to 5%. The magnitude of probabilities for the operators showed to be a good choice in several other applications.

2.3 Deterministic and Probabilistic Aspects

To improve fault coverage for hard to test faults, it is necessary to include some deterministic ATPG into the GA-environment. Our solution is a BDD-based forward-propagation algorithm as an optional routine of *Phase 2*. Starting from the reached state, a fault detection BDD is constructed, i.e. a BDD for a circuit where the according primary outputs of the fault-free and of the faulty circuit are XORed together and these XOR-outputs are ORed together; the primary inputs are shared. There are two sets of present state variables: one for the fault free and one for the faulty circuit. A fault is detectable according to SOT, iff there is an assignment to the primary input variables such that for all present state variables the fault detection BDD evaluates to 1. This found test pattern sequence (i.e. the 1-path in the quantified BDD) defines the initial values of the individuals of the current GA-population. Since usually only a few primary input variables occur on this 1-path only the according positions on the bit-string representing an individual are initially preset. The value of all other non occurring variables is randomly determined.

To compute a test sequence according to MOT, the same fault detection BDD must be build. However, the test of detection is different. The fault detection BDD of the current time frame is paired with the fault detection BDDs of all former time frames. Thus, a fault is detectable, iff there is a primary input variable assignment, such that the output sequence produced by the circuit-under-test cannot be determined by any initial state of the fault free circuit [17].

Since the fault detection BDD is constructed using the current state of the circuits that is largely initialized, the BDD-forward simulation described above works fine for five up to several ten time frames. Finally, the BDD-sizes will exceed a given size limit. In order to symbolically simulate a much larger number of time frames another algorithm is implemented. It bases on signal probabilities and fault detection probabilities, and the fact that both probabilities can be computed exactly and easily using BDDs [15]: A single fault detection BDD is constructed assuming the unknown initial state. The 'forward simulation algorithm' works now as follows: (1) Compute the signal probability of all next state lines of the former time frame. (2) Assign this value to the present state variables. (3) Compute the fault detection probability. (4) Increment the time frame and continue with step 1 for a given number of iterations. Since the BDDs are iteratively evaluated and no BDD-operations are performed, the algorithm works very efficiently. Changing the 1-probability value of the primary input variables changes the fault detection probability. Thus, optimizing the fault detection probability means manipulating the vector of input probabilities. This optimization task is done by an instance of the described GA. However, the algorithm makes an inherent error: Within each time frame all probabilities are exactly computed, but no signal dependencies across time frames are considered. Nevertheless, for two reasons the approximation is sufficient: (1) Experiments [11] showed that the relative error between the approximation and the real value is (very) small in most cases. (2) The computed input probability vector is finally used to determine an initial population of *Phase 2*. This means, it is sufficient that the input probability vector gives a good *hint* in a promising direction.

2.4 The ATPG Tool Composition

Now all components of our ATPG tool have been presented. In the following we show the options how to combine these components to an ATPG tool in order to achieve high quality test pattern for sequential circuits. At first, the tool has the possibility to select a three-valued greedy reset heuristic from [12]. If selected, the (partial) reset sequence forms the first part of the test sequence. In the first run, the GA determines a test sequence using the three-valued mode of *HFSim* only. In the second run, this sequence is firstly symbolically simulated by one of the three symbolic modes of *HFSim* (SOT, rMOT, MOT). Furthermore, the GA is started using the symbolic modes to improve the test sequence. Note, that each next step in our ATPG starts in the state reached by applying the test sequence found so far to the circuit-under-test. Depending on the user input, *Phase 2* of the second run selects either no ATPG subroutine (GA_{HFSim}, genetic only), or the probabilistic subroutine (GA_{HFSim} probabilities), or the deterministic subroutine (GA_{HFSim} deterministic).

3 Experimental Results

To show the efficiency of the approach we applied the ATPG algorithm to several *ISCAS 89* benchmark circuits. We compare the results to those of other GA based methods ([23, 22, 9]) and to a *Best of* selection of ATPG algorithms ([19, 1, 21, 10]). Table 1 shows the fault coverages (`FC`) and test length ($|$`Y`$|$). After the circuit name, the number of primary inputs and the number of memory elements are given. Furthermore, the best results from our tool are presented.

Experimental results concerning SOT are given in Table 2. The columns 2–11 show the results obtained by the restriction to a pure three-valued simulation. The labels `to` (`ro`) denote experiments using the tournament selection (roulette wheel selection). Tha label `re` inidicate the usage of the greedy reset sequence heuristic. (Note, that all results are achieved with *only one* parameter setting of the GA.) However, for almost all circuits we achieve better results in fault coverage and/or test sequence length than [23].

If the GA no longer finds an improvement using the three-valued mode of *HFSim*, the test sequence is handed to the GA-ATPG using symbolic evaluation and the same selection method (tournament or roulette wheel) as before. Now *Phase 1* starts in the (symbolic) state achieved after the simulation of Y. In the first iteration of *Phase 2* in the symbolic case, the elements of the test pattern are chosen randomly with a 1-probability of 50%. Since there are faults, that are hard to detect with such a sequence, *Phase 2* is iterated with different 1-probabilities: We let *Phase 2* run eleven times, from 0% to 100% 1-probability in 10% steps. The two other series of experiments are as follows: GA_{HFSim} using the fault detection probability optimization routine, and GA_{HFSim} using the deterministic test sequence computation routine. The experimental results are given on the right of Table 2. The column labelled `Symb. Sim.` shows the fault coverage by simply simulating the three-

name	*PI*	*SI*	Best of [19, 1, 21, 10]		[23]		[22]		[9]		Best results three-valued		Best results symbolic	
			FC	$\|Y\|$	FC	$\|Y\|$	FC	$\|Y\|$	FC	$\|Y\|$	FC	$\|Y\|$	FC	$\|Y\|$
s298	3	14	86.04	87	85.94	161	86.04	415	86.04	221	86.04	79	**87.66**	**90**
s344	9	15	96.20	37	96.20	95	95.91	169	96.20	75	96.20	56	**97.37**	**56**
s349	9	15	95.71	137	95.71	95	95.71	188	95.71	104	95.71	57	**96.86**	**57**
s386	7	6	**81.77**	**121**	76.88	154	**81.77**	359	**81.77**	241	80.47	128	**81.77**	160
s400	3	21	90.14	2424	85.70	280	81.22	704	90.14	2196	90.09	443	**92.22**	**655**
s444	3	21	89.45	1945	85.59	275	80.38	880	89.45	1046	88.19	372	**90.08**	**529**
s510	19	6	0.00	–	0.00	–	0.00	–	0.00	–	0.00	–	**100.00**	**245**
s526n	3	21	81.80	2642	75.08	281	67.75	873	81.62	2109	81.92	820	**83.36**	**860**
s641	35	19	86.51	63	86.51	139	86.51	292	86.51	140	86.51	64	**87.37**	**64**
s713	35	19	81.93	176	81.93	128	81.93	294	81.93	185	81.93	74	**82.62**	**74**
s820	18	5	95.76	424	60.76	146	95.76	1108	95.76	1243	68.35	106	**95.88**	**412**
s832	18	5	94.02	701	61.95	150	94.02	1064	94.02	1109	69.89	114	**94.14**	**400**
s953	16	29	8.87	20	–	–	–	–	–	–	8.34	16	**99.07**	**196**
s1196	14	18	**99.76**	244	99.19	347	**99.76**	377	**99.76**	948	99.60	233	**99.76**	**236**
s1238	14	18	**94.69**	247	94.02	383	**94.69**	409	**94.69**	880	94.61	231	**94.69**	**233**
s1488	8	6	97.17	317	93.67	243	97.17	1369	97.17	1191	97.17	367	**97.31**	**367**
s1494	8	6	96.48	540	94.02	245	96.48	1224	96.48	985	96.48	305	**96.61**	**305**
s5378	35	179	**79.06**	**11571**	68.98	511	70.35	683	76.17	7270	72.21	180	74.99	303
s35932	35	1728	**89.78**	257	89.55	197	89.17	425	89.77	825	**89.78**	**163**	**89.78**	**163**

Table 1. Experimental results found in literature compared to our best result.

name	GA_{HFSim} three valued: to		ro		to,re		ro,re		Symb. Sim.	GA_{HFSim} symbolic: Genetic only		Probabilistic		Deterministic	
	FC	$\|Y\|$	FC	$\|Y\|$	FC	$\|Y\|$	FC	$\|Y\|$	FC	FC	$\|Y\|$	FC	$\|Y\|$	FC	$\|Y\|$
s298	86.04	128	86.04	79	86.04	135	86.04	95	87.01	87.66	91	87.66	94	87.66	90
s344	96.20	61	96.12	51	96.20	72	96.20	56	97.37	–	–	–	–	–	–
s349	95.71	61	95.71	65	95.71	64	95.71	57	96.86	–	–	–	–	–	–
s386	79.65	99	78.91	96	80.47	128	78.91	125	80.47	80.73	142	81.25	172	81.77	160
s400	56.13	82	81.12	209	57.78	73	90.09	443	90.09	90.33	444	92.22	655	92.22	691
s444	60.55	47	60.76	42	88.19	372	60.55	51	88.19	88.40	373	90.08	529	90.08	557
s510	0.00	–	0.00	–	0.00	–	0.00	–	–	100.00	252	99.82	341	100.00	245
s526n	80.65	639	81.92	820	81.74	812	81.74	717	83.18	–	–	83.36	860	83.36	879
s641	86.51	99	86.51	67	86.51	95	86.51	64	87.37	–	–	–	–	–	–
s713	81.93	81	81.93	79	81.93	99	81.93	74	82.62	–	–	–	–	–	–
s820	58.12	121	68.35	106	67.76	145	64.71	143	68.47	68.59	131	85.06	727	95.88	412
s832	57.01	118	69.89	114	60.23	128	67.24	86	70.00	70.11	139	82.76	720	94.14	400
s953	8.34	17	8.34	19	8.34	16	8.34	17	26.04	99.07	253	99.07	294	99.07	196
s1196	99.52	346	99.60	243	99.28	335	99.60	233	99.60	–	–	–	–	99.76	236
s1238	94.46	458	94.46	342	94.17	318	94.61	231	94.61	–	–	–	–	94.69	233
s1488	97.17	367	96.97	280	96.90	335	97.11	301	97.31	–	–	–	–	–	–
s1494	96.28	332	72.21	305	96.15	267	96.35	299	96.61	–	–	–	–	–	–
s5378	70.15	222	72.50	180	70.19	274	70.11	203	72.50	72.52	182	*	*	74.99	303
s35932	89.78	163	89.78	163	89.78	170	89.78	191	89.78	–	–	*	*	–	–

Table 2. Results using three-valued evaluation (left) and symbolic simulation (right).

valued test sequence symbolically. A '$-$' denotes that the according method obtains no further improvement in comparison to the symbolic simulation. A '*' means that the according computation has been terminated due to too complex BDDs (800000 bdd-nodes maximum have been allowed) or too many faults to consider.

Assume the FC and test length of the deterministic GA_{HFSim} to be 100.00%. Table 3 (left) shows the ratio of these values to those of other algorithms. Since not all researchers report results for *s953* two columns have been made. For both columns the results for *s510* have not been considered since this circuit is untestable using basic three-valued ATPG. Obviously, our GA-ATPG computes higher fault coverages than all other presented ATPGs, even if the advantage is not much. However, our test sequence length is only a fraction of theirs: It is about only 2/9 !

Experiments concerning MOT are given in Table 4. It also shows that the change of the test method from SOT to rMOT further improves the fault coverage and often shortens the test sequence in comparison to symbolic SOT. Comparing to MIX [18], that only supports rMOT, our (r)MOT-GA_{HFSim} computes mostly the same fault coverage. However, our test sequence length is only as half as long (Table 3, right).

The execution time of the symbolic part of the ATPG algorithm is, in general, only some minutes due to the following facts: Most faults have already been found and the correct circuit and many faulty circuits are initialized. Therefore, the memory demand of the symbolic simulation is small due to small BDDs. For example for circuit *s5378*: Three valued GA_{HFSim} using tournament selection takes an execution time of 6h 50 minutes, whereas the symbolic GA_{HFSim} SOT (MOT) part needs time less than additional 11 (54) minutes on a Sparc Ultra 1. However, the cumulative three-valued simulation time is high because we do perform a single-fault single-pattern simulation until now. Thus, a parallel version is focus of our current work.

4 Conclusions

In this paper we presented a *Genetic Algorithm* (GA) for *Automatic Test Pattern Generation* (ATPG) for sequential circuits. We outlined the advantages of using hybrid symbolic fault simulation concerning the quality of the fault coverages. The experiments have demonstrated that our ATPG tool can generate high quality test patterns in both, fault coverage and test length: The test length is reduced to 2/9 (1/2 for (r)MOT) with respect to that of comparable ATPGs without neglecting the fault coverage. This quality is mainly based on a high quality GA. Applying symbolic methods, in particular rMOT, does further improve the quality of the test sequence without any drawbacks in test execution.

Acknowledgment

The authors would like to thank Dr. Rolf Drechsler for his participation in a former version of this work.

SOT				
	Without s953		With s953	
Method	FC	$\|Y\|$	FC	$\|Y\|$
Det. GA_{HFSim}	100.00	100.00	100.00	100.00
Prob. GA_{HFSim}	98.35	108.72	98.45	110.26
Symb. GA_{HFSim}	96.19	76.72	96.42	78.68
Symb. Sim.	96.08	75.13	91.87	72.63
Best three valued	95.64	75.13	90.37	72.63
[9]	99.40	411.49	–	–
[22]	96.90	214.64	–	–
[23]	92.82	75.89	–	–
Best of [19, 1, 21, 10]	99.60	434.57	94.13	418.71

rMOT Without s510		
Method	FC	$\|Y\|$
Det. GA_{HFSim}	100.00	100.00
Symb. GA_{HFSim}	94.51	73.03
Symb. Sim.	90.14	68.10
Best three valued	89.88	68.10
[18]	100.32	199.07

Table 3. Comparing the experimental results. SOT on the left and MOT on the right.

name	MIX		Best three valued		rMOT Sim.	GA_{HFSim} rMOT				MOT Sim.	GA_{HFSim} MOT			
						Genetic only		Deterministic			Genetic only		Deterministic	
	FC	$\|Y\|$	FC	$\|Y\|$	FC	FC	$\|Y\|$	FC	$\|Y\|$	FC	FC	$\|Y\|$	FC	$\|Y\|$
s298	88.64	206	86.04	79	88.31	88.64	86	88.64	90	88.31	88.64	86	88.64	90
s344	97.95	76	96.20	56	97.95	–	–	–	–	97.95	–	–	–	–
s349	97.43	104	95.71	57	97.43	–	–	–	–	97.43	–	–	–	–
s386	81.77	226	80.47	128	80.47	80.73	142	81.77	160	80.47	80.73	142	81.77	160
s400	93.40	1170	90.09	443	92.22	92.45	444	93.63	706	92.22	92.45	444	93.41	706
s444	91.77	870	88.19	372	90.51	90.72	373	92.41	929	90.51	90.72	373	92.41	929
s510	100.00	587	0.00	–	–	100.00	337	*	*	–	100.00	309	*	*
s526n	83.24	1545	81.92	820	84.09	–	–	84.27	879	84.09	–	–	84.27	879
s641	87.37	131	86.51	64	87.37	–	–	–	–	87.37	–	–	–	–
s713	82.62	130	81.93	74	82.62	–	–	–	–	82.62	–	–	–	–
s820	95.88	838	68.35	106	68.47	68.59	131	95.88	412	68.47	68.59	131	95.88	412
s832	94.14	881	69.89	114	70.00	70.11	139	94.14	400	70.00	70.11	139	94.14	400
s953	99.07	563	8.34	16	27.90	99.07	217	99.07	188	27.90	99.07	212	99.07	188
s1196	99.76	306	99.60	233	99.60	–	–	99.76	236	99.60	–	–	99.76	236
s1238	94.69	347	94.61	231	94.61	–	–	96.69	233	64.61	–	–	96.69	233
s1488	97.31	918	97.17	367	97.31	–	–	–	–	97.31	–	–	–	–
s1494	96.61	759	96.48	305	96.61	–	–	–	–	96.61	–	–	–	–
s5378	79.14	1766	72.21	180	72.54	72.56	182	75.21	273	72.54	72.56	182	75.21	273
s35932	89.81	296	89.78	163	89.81	–	–	–	–	89.81	–	–	–	–

Table 4. Experimental results using different symbolic evaluation methods (rMOT and MOT).

References

[1] M. Abramovici, K.B. Rajan, and D.T. Miller. Freeze: A new approach for testing sequential circuits. In *Design Automation Conf.*, pages 22–25, 1992.

[2] R.E. Bryant. Graph - based algorithms for Boolean function manipulation. *IEEE Trans. on Comp.*, 35(8):677–691, 1986.

[3] F. Corno, P. Prinetto, M. Rebaudengo, and M.S. Reorda. Comparing topological, symbolic and GA-based ATPGs: an experimental approach. In *Int'l Test Conf.*, pages 39–47, 1996.

[4] F. Corno, P. Prinetto, M. Rebaudengo, and M.S. Reorda. GATTO: A genetic algorithm for automatic test pattern generation for large synchronous sequential circuits. *IEEE Trans. on CAD*, 15(8):991–1000, 1996.

[5] F. Corno, P. Prinetto, M. Rebaudengo, M.S. Reorda, and R. Mosca. Advanced techniques for GA-based sequential ATPG. In *European Design & Test Conf.*, pages 375–379, 1996.

[6] L. Davis. *Handbook of Genetic Algorithms*. van Nostrand Reinhold, New York, 1991.

[7] R. Drechsler. Evolutionary algorithms for computer aided design of integrated circuits. In *Int'l Symposium on IC Technologies, Systems and Applications*, pages 302–311, 1997.

[8] J.H. Holland. *Adaption in Natural and Artifical Systems*. The University of Michigan Press, Ann Arbor, MI, 1975.

[9] M.S. Hsiao, E.M. Rudnick, and J.H. Patel. Alternating strategies for sequential circuit ATPG. In *European Design & Test Conf.*, pages 368–374, 1996.

[10] M.S. Hsiao, E.M. Rudnick, and J.H. Patel. Sequential circuit test generation using dynamic state traversal. In *European Design & Test Conf.*, pages 22–28, 1997.

[11] M. Keim and B. Becker. Nearly exact signal probabilities for synchronous sequential circuits – an experimental analysis. Technical Report 106/98, Albert-Ludwigs-University, Freiburg, Jun. 1998.

[12] M. Keim, B. Becker, and B. Stenner. On the (non-) resetability of synchronous sequential circuits. In *VLSI Test Symp.*, pages 240–245, 1996.

[13] M. Keim, N. Göckel, R. Drechsler, and B. Becker. Combining GAs and symbolic methods for high quality tests of sequential circuits. Technical Report 105/98, Albert-Ludwigs-University, Freiburg, May 1998.

[14] M. Keim, N. Göckel, R. Drechsler, and B. Becker. Test generation for (sequential) multi-valued logic networks based on genetic algorithm. In *Int'l Symp. on multi-valued Logic*, 1998.

[15] R. Krieger. PLATO: A tool for computation of exact signal probabilities. In *VLSI Design Conf.*, pages 65–68, 1993.

[16] R. Krieger, B. Becker, and M. Keim. A hybrid fault simulator for synchronous sequential circuits. In *Int'l Test Conf.*, pages 614–623, 1994.

[17] R. Krieger, B. Becker, and M. Keim. Symbolic fault simulation for sequential circuits and the multiple observation time test strategy. In *Design Automation Conf.*, pages 339–344, 1995.

[18] X. Lin, I. Pomeranz, and S.M. Reddy. MIX: A test generation system for synchronous sequential circuits. In *VLSI Design*, pages 456–463, 1998.

[19] T.M. Niermann and J.H. Patel. HITEC: A test generation package for sequential circuits. In *European Conf. on Design Automation*, pages 214–218, 1991.

[20] I. Pomeranz and S.M. Reddy. The multiple observation time test strategy. *IEEE Trans. on Comp.*, pages 627–637, May 1992.

[21] I. Pomeranz and S.M. Reddy. On static compaction of test sequences for synchronous sequential circuits. In *Design Automation Conf.*, pages 215–220, 1996.

[22] E. Rudnick and J.H. Patel. Combining deterministic and genetic approaches for sequential circuit test generation. In *Design Automation Conf.*, pages 183–188, 1995.

[23] E.M. Rudnick, J.H. Patel, G.S. Greenstein, and T.M. Niermann. Sequential circuit test pattern generation in a genetic algorithm framework. In *Design Automation Conf.*, pages 698–704, 1994.

[24] D.G. Saab, Y.G. Saab, and J.A. Abraham. Iterative [simulation-based genetics+deterministic techniques] =complete ATPG. In *Int'l Conf. on CAD*, pages 40–43, 1994.

[25] D.S. Saab, Y.G. Saab, and J.A. Abraham. Cris: A test cultivation program for sequential vlsi circuits. In *Int'l Conf. on CAD*, pages 216–219, 1992.