HITEC: A TEST GENERATION PACKAGE FOR SEQUENTIAL CIRCUITS

Thomas Niermann
Sunrise Test Systems Inc.
Los Altos Ca. 94022 U.S.A.

Janak H. Patel
University of Illinois
Coordinated Science Laboratory
Urbana Illinois, 61801 U.S.A.

Abstract:

This paper presents HITEC, a sequential circuit test generation package to generate test patterns for sequential circuits, without assuming the use of scan techniques or a reset state. Several new techniques are introduced to improve the performance of test generation. A targeted D element technique is presented, which greatly increases the number of possible mandatory assignments and reduces the over-specification of state variables which can sometimes result when using a standard PODEM algorithm. A technique to use the state knowledge of previously generated vectors for state justification, without the memory overhead of a state transition diagram is presented. For faults that were aborted during the standard test generation phase, knowledge that was gained about fault propagation, by the fault simulator, is used. These techniques, when used together, produce the best published results for the ISCAS89 sequential benchmark circuits.

1. Introduction

Test pattern generation has progressed to a stage where automatic test generation is feasible on almost any combinational circuit. However, the same is not true of sequential circuit test generation. While scan based approaches can convert the sequential circuit into a combinational circuit for testing purposes, the cost of complete scan can be prohibitive in both area overhead and performance degradation. Therefore, an efficient sequential circuit test generation system is necessary which generates tests for all detectable faults and identifies all undetectable faults in the original design. The information on undetectable faults could be used to add only necessary testability overhead to make these faults testable.

This paper presents HITEC a sequential circuit test generation package which runs on sequential circuits. The main priorities of this system is to generate test sets with very high coverage, and to identify the undetectable faults. Several new techniques are introduced to improve the performance of HITEC. 1) A targeted D element technique greatly increases the number of possible mandatory assignments and reduces the over-specification of state variables, which can sometimes result when using a standard PODEM algorithm. 2) The state knowledge of previously generated vectors for state justification is used without the cost of the memory overhead of a state transition diagram. 3) For faults that were aborted during the standard test generation phase, knowledge that was gained about the fault propagation by the fault simulator is used.

This research was supported by Semiconductor Research Corporation under Contract SRC 89-109-DP.

HITEC is able to achieve a very high fault coverage, without using any user supplied high level information such as a RTL description. Not only does this avoid the extra user time of designing and verifying the high level description, but it ensures the completeness of the algorithm by always working at the gate level. By keeping the algorithm complete, HITEC will be successful on a far greater number of designs than an incomplete algorithm, and can also identify untestable faults.

2. Sequential Circuit Test Generation

Most sequential test generators conceptually transform a synchronous sequential circuit into an equivalent iterative logic array model. There are two main techniques to generate tests using the iterative array model. The first approach is to proceed backwards through time for fault propagation and state justification, thus using only reverse time processing[1, 2, 3, 4]. The second approach is to use forward time processing for line justification and fault propagation followed by reverse time processing for state justification[5, 6, 7].

HITEC consists of two phases. The first is the forward time processing phase in which the fault is activated and propagated to a primary output. The second phase is the justification of the initial state determined in the first phase. HITEC uses a decision strategy based on the implicit enumeration of PODEM [8] and uses dominators and mandatory assignments of FAN, TOPS, and SOCRATES[9, 10, 11, 12, 13, 14]. A 9 valued logic system is used to ensure completeness of the algorithm[15, 16]. Figure 1 illustrates an overview of the test generation system. In Figure 1, FTP is forward time processing and PPO is a pseudo-primary output, i.e. inputs to a D-flip-flop.

Figure 2 shows the diagram of a single time frame used in this paper. The primary inputs and outputs enter and exit the circuit on the top and the bottom, respectively. The pseudo primary inputs and outputs enter and exit the circuit on the left and the right, respectively, so that when time frames are aligned horizontally the PPOs of one time frame attach to the PPIs of the next time frame. The fault site is marked with an X. For highly sequential circuits, the number of time frames needed to detect a fault may be too large. For this reason, during the forward time processing phase we use a dynamically adjustable window size that is allowed to expand to a maximum size determined by the size of the circuit and the size of memory available. If the size of the window is at its maximum and more time frames are needed, the window is scrolled over the time frame expansion of the circuit. Figure 3 shows a circuit with a window size of four. Backtrace is performed on the four frames, making

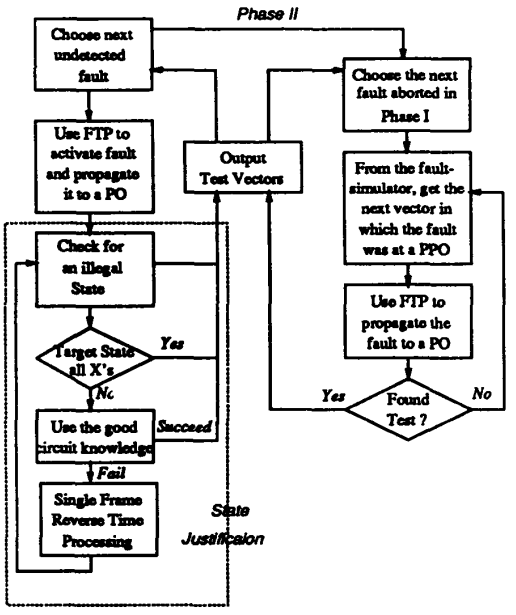

Fig. 1. System Overview

assignments on the primary inputs of the four frames or the pseudo inputs of the first frame of the window. The pseudo inputs that are assigned values is the required *initial state* which is justified by reverse time processing.

2.1. Targeted D frontier

The use of dominators to determine mandatory assignments has been shown to be very helpful in combinational circuit test generation[9, 10, 11]. A gate is a *dominator* of a line if all paths from that line to any primary output pass through that gate. Dominator gates of a line must have their inputs assigned to a noncontrolling value (i.e. 1's for AND and NAND gates and 0's for OR and NOR gates) to propagate a fault from that line to a primary output. Other mandatory assignments can be determined by backward and forward implication on those noncontrolling values. These mandatory assignments can potentially reduce the search space. One drawback to the way in which traditional algorithms handle dominators is that, if there are more then one D in the frontier, then the dominators are determined by taking the intersection of the dominators of each of the D frontier nodes, thus resulting in a small dominator set. One technique used to address the problem of a small dominator set is to choose a target output node and determine the dominators with respect to that output only[13], thus increasing the number of mandatory assignments.

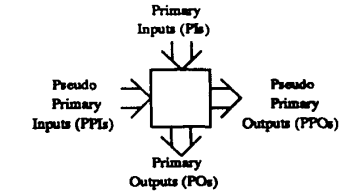

Fig. 2 A single time frame

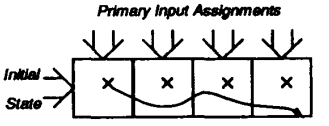

Fig. 3. Forward Time Processing

HITEC uses a targeted D frontier technique during the forward time processing phase to increase the number of mandatory assignments by always limiting the D frontier to one target D element. All other possible D frontier elements are pushed onto the decision stack, to be tried only if the chosen target D frontier element failed and the test generator backtracked to the point where the other D frontier elements were pushed onto the stack. Once a new target D element is chosen from the frontier, all its mandatory assignments are determined through the use of dominators and forward and backward implication. Thus the test generator has a set of objectives to be met so that the D element is propagated forward. The objectives are met by the PODEM backtrace algorithm. The objectives are justified in the order from the hardest to control to the easiest to control, based on the SCOAP controllability measures[17]. Once the objectives for the mandatory assignments are met, the next target D element is chosen from a list of D frontiers which are the successors of the current D frontier, by selecting the most observable element in the list.

Aside from increasing the number of mandatory assignments, the targeted D frontier technique has one other very helpful side effect. The specification on the primary inputs and the initial state is significantly reduced. If the test generator needs to backtrack to the point where a target D frontier decision was made, then all assignments made to propagate the original target D element have been undone, thus yielding the state of the test generator as if the second choice of target had been made originally. All assignments made to propagate the D along the blocked path are undone, thus putting less specification on the initial state and primary input assignments. If the PIs are less specified, then there is less chance of an incorrect PI assignment which can lead to backtracks deep into the stack. An initial state with more don't cares is inherently easier to justify for the state justification phase.

2.2. State Justification

State justification in HITEC is accomplished by reverse time processing. Three different techniques are tried to justify the state in the last expanded time frame. 1) Matching a previously visited good circuit state, 2) Single frame reverse time processing using PODEM, and 3) Determining unjustifiable states. In actual implementation, method (3) is incorporated in the methods (1) and (2), but merits separate explanation.

Using the good circuit state knowledge: Use of good circuit state knowledge in test generator was first proposed in [5, 6]. Our approach is very similar except for minor modifications. A search is done on the good circuit states reached by the set of vectors already created by the test generator. If the current state to justify is a subset of any one of those states, then there is a good chance that the vectors used to get the circuit into that state for the good circuit, will also get the circuit into that state for the faulty circuit. To check that the fault does not invalidate the state and to reduce the number of vectors to be reapplied, the vectors are applied one at a time in reverse order starting from the last vector that got the circuit into the target state. If there is a conflict with the target state, then the fault invalidates the state and the test generator proceeds to the technique of single frame reverse time processing, described below, to justify a state. If all the objectives of the target state are met in the first technique, then a test has been found and there is no need to reapply any more vectors. Note that if the state has been invalidated by the vectors, then a D has been forward propagated to a PPO, and the second phase of test generation will attempt to forward propagate this D to a primary output. Figure 4 shows an example of this technique. The table at the top shows the list of vectors previously generated and the corresponding state of the good circuit. The state of V4 has no conflicting values with the target state, so V4 is applied to the time frame. After applying V4, the target state is not fully met, so the window is expanded back one time frame and V3 is applied to that time frame. Because the target state still is not met, the window is expanded and V2 is applied. After this step a sequence of vectors has been found to set the initial state and the fault does not invalidate the sequence. Hence the test for the target fault has been found.

Single frame reverse time processing: If the good circuit knowledge was unsuccessful, then PODEM is applied to the single frame to attempt to justify the state. First, all the mandatory assignments are determined by backward and forward implication. The objectives to justify the mandatory assignments are then ordered from the hardest to control to the easiest to control, and then backtrace is performed on the objectives in that order. If single frame reverse time processing determines through implicit enumeration of PODEM that the current target state cannot be justified, then it is added to a list of unreachable states for this fault. This list is checked before attempting to justify any state, thus eliminating the repetition of some work. If this step is successful, then the target state becomes the state used to meet the current target state, and each of these three steps are attempted again.

Illegal states: A set of states called *illegal states*, is kept for each fault that the test generator has determined a

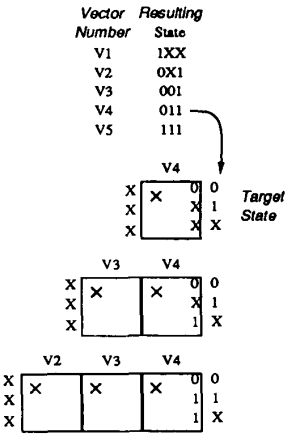

Fig. 4. Using good circuit state information

state justification process will not succeed. Such states are determined to be illegal by one of three criteria. First if a PPI has an infinite SCOAP controllability measure, then this flip-flop cannot be initialized. Thus for example, if PPO A has an infinite controllable 1 value then any state containing state variable A=1 is an unjustifiable state. The controllability measures are also computed for the faulty machine to determine any states made unjustifiable by the fault. The second criterion for illegal states is state repetition. If a state has been visited earlier during the current justification process, then the test generator found a loop in the state transition diagram and it should not continue to circle that loop. In this context illegal state does not mean unjustifiable state, it simply means that at this instant a justification attempt on this state will be wasted. The last criterion is identifying states which have been previously determined (through implicit enumeration) to be unjustifiable state, i.e. unreachable in the context of gate level test generation. A clarification on unjustifiability of states is necessary here. A state determined unjustifiable by a test generator using gate level implications may indeed be reachable from an all unknown state. However, this would require more costly unknown value propagation techniques, such as proposed in[18]. In this paper, we will only use the gate level techniques. We do not include in the illegal list the states which could not be justified due to time limit. The reason for this is that in our implementation the time limit is periodically increased during the test generation process.

2.3. Using Fault simulation knowledge

After the first pass of test generation, there may be faults which are not marked as redundant or untestable, but are not yet detected by the test set. To attempt to detect these faults, a second pass to the test generation process is used. This pass uses the knowledge obtained by the fault simulator, about which faults propagated to a PPO. For each dropped

fault the fault simulator keeps a record of the time frames in which the fault was propagated to a PPO. For each time frame in the list for a fault, the following operations are performed until a test is found. The vectors are applied in reverse order, from the time frame chosen, till the fault is activated and propagated to the PPO. The next step is to attempt to propagate the fault forward to a primary output. If more time frames are needed the time frames are expanded forward. Figure 5 illustrates this technique. The fault simulator determined that the target fault was propagated to a PPO at time frame 12. Vector 12 is then applied to one time frame and all the PPOs of that time frame are checked for a value D. Because there is no D on a PPO, vector 11 is applied to the previous time frame. Again there is no D at a PPO, so vector 10 is applied. A D is at a PPO, so these three vectors were sufficient to propagate the D to a PPO. Next, forward time processing is applied to propagate the fault to a primary output. More time frames are expanded forward as needed.

This technique helps improve the fault coverage in the harder to test circuits. This technique basically reverses the FTP and RTP phases. The state of the circuit is set to activate the fault, and then FTP is used to push the fault effect to a primary output. Because this technique is applied to faults which the standard FTP followed by RTP failed, and it reverses the order which these phases are applied, this technique provided a second very different algorithm for test generation. This algorithmic duality provides a greater probability of success on a large number of circuits.

3. Results

HITEC has been interfaced to the PROOFS fault simulator[19] and has been run on several of the ISCAS sequential benchmark circuits on a SparcStation 1 with 16MB memory. Table 1 shows the attributes of the benchmark circuits used to evaluate the test generator. All these circuits are from the ISCAS89 benchmark circuits without the use of any scan or reset state[20]. Table 2 shows the test generation results of HITEC, along with the fault simulation results. HITEC uses a variable time limit due to large variation in complexity of sequential circuits. Initially the time limit is set to 2 seconds per fault which then is increased ten fold for the aborted faults. This increase in time limit is repeated until a desired test generation efficiency is reached or a total user specified time limit is reached. The test generation efficiency is defined as the ratio of detected faults plus proven untestable faults over the total number of faults. The untestable faults include combinationally and sequentially redundant faults. In addition, those faults that require unjustifiable states are also included. Unjustifiable states were defined earlier in the paper in the context of gate level implications. It should be noted that the faults declared untestable due to unjustifiable states are not necessarily redundant. In Table 2, extremely low fault coverage in some circuits is due to the fact that the standardized benchmark description does not include any resettable flip-flops. In some circuits, the absence of reset lines on flip-flops made the circuits competely uncontrollable, and hence the low fault coverage.

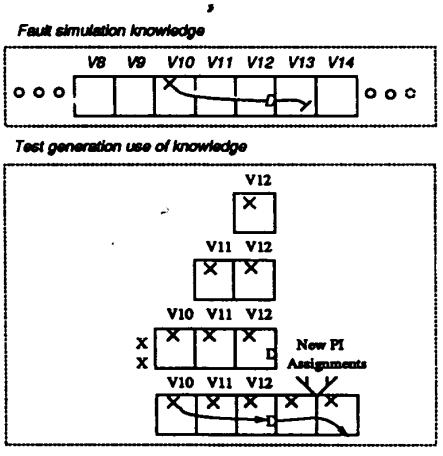

Fig. 5. Using fault simulation knowledge

4. Conclusion

This paper has presented several new techniques to improve the performance of sequential circuit test generators. Among the new concepts presented in this paper are the use of fault simulation knowledge to increase test coverage during a second phase of test generation and a targeted D element technique for D propagation. The concepts presented in the paper were implemented and tested on the ISCAS sequential benchmark circuits. These techniques, when used together, produce the best published results for the ISCAS89 sequential benchmark circuits.

Table 1. Circuit Statistics

Circuit	Gates	D flip flops	Primary Inputs	Primary Outputs	Faults
s208	96	8	11	2	215
s344	160	15	9	11	342
s349	161	15	9	11	350
s386	159	6	7	7	384
s420	196	16	19	2	430
s510	211	6	19	7	564
s641	379	19	35	24	467
s713	393	19	35	23	581
s820	289	5	18	19	850
s832	287	5	18	19	870
s953	395	29	16	23	1079
s1196	529	118	14	14	1242
s1238	508	18	14	14	1355
s1488	653	6	8	19	1486
s1494	647	6	8	19	1506
s9234	5597	228	19	22	3934
s15850	9772	597	14	87	11513
s35932	17793	1728	35	320	39094

Table 2 HITEC Test Generation Results

Circuit	Test Gen Time(sec)	Fault Sim Time(sec)	Vectors	Detected Faults	Untestable Faults	Fault Coverage	Tgen Efficiency
s208	24	5	184	137	78	63.7	100
s344	4,775	10	142	328	11	95.9	99.1
s349	3,129	6	137	335	13	95.7	99.4
s386	62	20	311	314	70	81.7	100
s420	2,701	15	218	179	251	41.6	100
s510	2	4	0	0	564	0	100
s641	737	40	242	404	62	86.5	99.7
s713	84	40	253	476	105	81.9	100
s820	1,374	56	1,254	812	32	95.4	99.1
s832	2,079	191	1,316	815	47	93.6	99.0
s953	921	16	41	89	990	8.2	100
s1196	41	12	480	1239	3	99.7	100
s1238	183	18	535	1283	72	94.6	100
s1488	12,943	405	1,294	1444	42	97.0	100
s1494	6,874	107	1,407	1453	53	96.4	100
s9234	63	62	24	18	3916	0.2	100
s15850	1,580	124	32	86	11403	0.7	99.7
s35932	55,434	3,629	1,000	34697	3984	88.8	99.0

5. References

[1] W.T. Cheng, "The back algorithm for sequential test generation," *Proc. 1988 IEEE Int. Conf. on Computer Design*, pp. 66-69, Oct. 1988.

[2] M.H. Schulz and E. Auth, "Essential: an efficient self-learning test pattern generation algorithm for sequential circuits," *Proc. 1989 Int. Test Conf.*, pp. 28-37, Sept. 1989.

[3] Ralph Marlett, "EBT: a comprehensive test generation technique for highly sequential circuits," *Proc. 15th Design Automation Conf.*, pp. 335-339, June 1978.

[4] R. Marlett, "An effective test generation system for sequential circuits," *23rd Design Automation Conference*, pp. 250-256, June 1986.

[5] H.K. Tony Ma , S. Devadas , A.R. Newton , and A. Sangiovanni-Vincentelli , "Test generation for sequential circuits ," *IEEE Trans. Computer-Aided Design* , pp. 1081-1093 , Oct. 1988.

[6] Hi-Keung Tony Ma, Srinivas Devadas, A. Richard Newton, and Alberto Sangiovanni-Vincentelli, "Test generation for sequential finite state machines," *IEEE Int. Conf. on Computer-Aided Design 1987*, pp. 288-291, Nov. 1987.

[7] A. Ghosh, S. Devadas, and A.R. Newton, "Test generation for highly sequential circuits," *IEEE Int. Conf. on Computer-Aided Design 1989*, pp. 362-365, Nov. 1989.

[8] P. Goel, "An implicit enumeration algorithm to generate tests for combinational circuits," *IEEE Trans. Computer*, vol. C-30, pp. 215-222, March 1981.

[9] H. Fujiwara and T. Shimono, "On the acceleration of test generation algorithms," *IEEE Trans. on Computers*, pp. 1137-1144, Dec. 1983.

[10] T. Kirkland and M.R. Mercer, "A topological search algorithm for ATPG," *24th Design Automation Conference*, pp. 502-508, June 1987.

[11] M.H. Schulz, E. Trischler, and T.M. Sarfert, "SOCRATES: a highly efficient automatic test pattern generation system," *IEEE Trans. on CAD*, pp. 126-137, Jan. 1988.

[12] R.E. Tarjan, "Finding dominators in directed graphs," *SIAM Journal of Computing*, pp. 62-89, 1974.

[13] T. Gruning, U. Mahlstedt, W. Daehn, and C. Ozcan, "Accelerated test pattern generation by cone-oriented circuit partitioning," *1990 European Design Automation Conference*, pp. 418-421, Feb. 1990.

[14] M. H. Schulz and E. Auth, "Advanced automatic test pattern generation and redundancy identification techniques," *Proc. 18th Int. Symposium on Fault-Tolerant Computing*, pp. 30-35, June 1988.

[15] W.T. Cheng, "Split circuit model for test generation," *25th Design Automation Conf.*, pp. 96-101, June 1988.

[16] P. Muth, "A nine-valued circuit model for test generation," *IEEE Trans. Computer*, pp. 630-636, June 1976.

[17] L.H. Goldstein and E.L. Thigpen, "SCOAP: sandia controllability/observability analysis program," *Dig. 17th Design Automation Conf.*, pp. 190-196, June 1980.

[18] S.J. Chandra and J. Patel, "Accurate logic simulation in the presence of unknowns," *IEEE Int. Conf. on Computer-Aided Design (ICCAD-89)*, pp. 34-37, November 1989.

[19] T. Niermann, W.T. Cheng, and J.H. Patel, "PROOFS: a fast memory efficient fault simulator for sequential circuits," *27th Design Automation Conf.*, June 1990.

[20] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," *Proc. 1989 Int'l. Symp. Circuits and Systems*, pp. 1929-1934, 1989.