THE BACK ALGORITHM FOR SEQUENTIAL TEST GENERATION

Wu-Tung Cheng

AT&T Engineering Research Center
P. O. Box 900
Princeton, NJ 08540

ABSTRACT

A new test generation algorithm called the BACK algorithm is proposed for sequential test generation. The BACK algorithm is a modified D-algorithm. Instead of forward fault propagation to generate sensitized paths, the BACK algorithm creates the sensitized paths backwards by justifying the required sensitized values. Without the forward propagation process of the D-algorithm, the BACK algorithm only has backward justification process through the circuit and through time. The BACK algorithm not only is easier to implement but also requires less run time memory than the D-algorithm. A new testability measurement which guides the justification of sensitized values will be presented also. The BACK algorithm has been implemented and used very successfully for AT&T sequential circuits. Experimental results of this tool will also be given.

I. INTRODUCTION

A sequential circuit can be always modeled as composed of two parts: a combinational part and a memory part, as shown in Figure 1. A mapping of the time domain behavior into space domain is shown in Figure 2. Each copy corresponds to the behavior at one *time frame*. Such an arrangement is also called an *iterative logic array* composed of several identical copies of combinational circuits. Therefore, besides the problems involved in the test generation for combinational circuits, the test generation for sequential circuits has memory problem. The upper bound of the number of time frames needed to generate a test sequence for one fault is 9^n if the initial state is unknown, or 4^n if the initial state is known; here n is the number of state elements. Allocating enough memory for all the time frames needed during test generation process is a sever problem.

Several approaches have been used to generate tests for sequential circuits based on the iterative logic array model. The problems of those approaches will be discussed in the next section. To solve these problems, a new algorithm called BACK will be described in Section III. The BACK algorithm has been implemented and used very successfully within AT&T for sequential circuits. Experiment results of this tool will be given in Section IV. In Section V, concluding remarks will be presented. In this paper, single line stuck-at fault model is used.

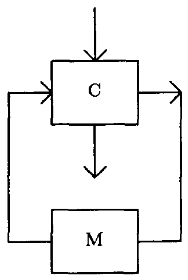

Figure 1.

II. PREVIOUS WORK

Figure 3 shows a sequential circuit using iterative logic array model. There is a single stuck-at fault which has repeated fault effect at every time frame. Each * is the fault site of a time frame. Here, the dotted line is the boundary of each time frame. The primary inputs are at the top and the primary outputs are at the bottom. Assuming this fault can be tested by a test sequence with $N+M$ time frames as shown in the figure. The curved lines are the ***sensitized paths*** which propagate the fault effects to the primary outputs. Because there are multiple fault sites which can be activated or not, the sensitized paths can have *multiple sources*. The test sequence uses N time frames for state initialization to activate the fault effects and M time frames for fault propagation to make the fault effects observable.

The D-algorithm [1] is the first complete test generation algorithm for combinational circuits. It is complete because, giving enough time, it will generate a test for any testable fault. Basically, the D-algorithm first selects all the sensitized paths needed then justifies all the values required to establish these paths. The D-algorithm was extended in [2] to handle multiple fault sites for sequential circuits. Both the sensitized path selection process and line value justification process are multiple decision processes. If any conflicts happen, the whole process has to backtrack to the previous decision to try another choice. To be able to try another choice, the circuit status has to be stored at every decision point for later recovery. Because the requirements created during the forward process have to be justified by the backward process later, in Figure 3, the circuit status must contain at least M time frames.

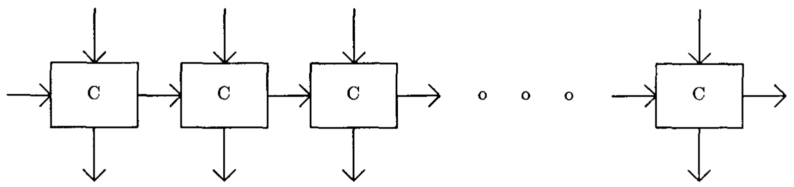

Figure 2.

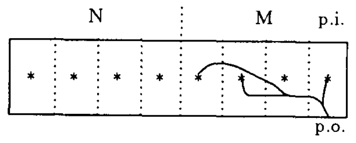

Figure 3.

If M is big, the total memory requirement will be very large, especially for large circuits which usually have lots of decision points during test generation. Since M is not predictable and its upper bound is very large, the memory management is very complicated. There is another problem pointed out by Muth in [3]. If the forward process does not selects all the sensitized paths needed as shown in Figure 4, *then the test sequence cannot* be found and this conflict situation cannot be detected until in justification process which is very late and will take a lot of time to recover. Thus, the 9-valued D-algorithm was proposed to allow sensitized paths to be created during justification process. Hence it only has to select a sensitized path which is from a fault site to a primary output as shown in Figure 4. During justifying this path the other sensitized paths needed will be created to get the test sequence in Figure 3. However, the 9-valued D-algorithm did not solve the memory problem.

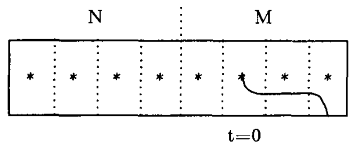

Figure 4.

The EBT (Extended Backtrace) method [4] which is a modified 9-valued D-algorithm can solve the memory problem. This method gives up the forward process by allowing no choice on sensitized path selection. That means it only tries to find a test by justifying the one and only one preselected sensitized path. As shown in Figure 5, with the preselected sensitized path, this method starts from the sensitized primary output backward through the circuit and through time to justify this path. In this manner, all the requirements of one time frame can be handled simultaneously. Thus, the circuit status does not have to include any time frame after the requirements of this time frame are all satisfied. Furthermore, for every sequential or combinational element, its output value requirement can be justified by its input value of current time frame and/or its state value of previous time frame. Therefore, independent of the number of sequential elements, at any time during test generation process of the EBT method the circuit status only has to contain two time frames: current time frame and previous time frame. The disadvantage of the EBT method is that the preselected sensitized path is not always correct. Since the number of possible paths is very large, to try path by path is impractical. Therefore, the EBT method is not a complete algorithm. In [5], in order to try more paths, a diagram composed of several possible paths is preconstructed. If one path failed, several paths can be eliminated from the diagram by their topological relation to speed up the whole process. However, the maintenance of this diagram is costly in time and memory and it is impractical to include all the possible paths in the diagram.

Another popular and complete test generation algorithm for combinational circuits is called PODEM [6]. Basically, PODEM only assigns values at the primary inputs. After values are assigned, forward line value implication is carried out to create sensitized paths. With this strategy, PODEM establishes sensitized paths without backward line value justification. There are several

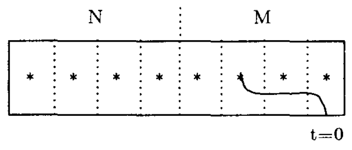

Figure 5.

heuristics proposed in [7-8] to guide the value assignments at the primary inputs such that sensitized paths can be created efficiently. To apply PODEM for sequential circuits, the process would look like Figure 6. PODEM starts from the leftmost time frame forward through the circuit and through time, such that no backward line value justification is needed. However, there are two problems that make this approach impractical. First, there is no guidance to decide how many time frames (the value of N) needed for state initialization. Second, to guide the assignment at the primary inputs for fault propagation purpose, the circuit status needs to contain $N+M$ time frames which has worse memory problem than the D-algorithm has. In [9], PODEM was used in different manner. Two separate approaches were used for the M time frames and the N time frames. First, PODEM is used to establish sensitized paths in the M time frames. Then, the state initialization of the N time frames is done through a preconstructed state diagram. It has the same memory problem as the D-algorithm has. Furthermore, the memory requirement of the preconstructed state diagram makes the memory problem even worse.

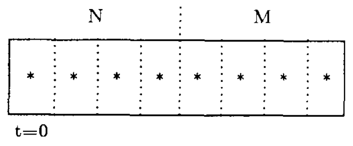

Figure 6.

Finally, it may be concluded that the forward process should be avoided in test generation for sequential circuits to solve memory problem. The BACK algorithm which will be presented in the next section only uses the backward process as the EBT method but covers all the possible sensitized paths automatically.

III. THE BACK ALGORITHM

In this section, the notation i/j means "good machine (circuit without fault) has value i and bad machine (circuit with the assumed fault) has value j".

When test sequence for one target fault are applied to the circuit, the good machine and the bad machine should have sensitized values (1/0 or 0/1) on at least one of its primary outputs. With this basic requirement, the BACK algorithm first predicts one primary output to have the sensitized value needed and then justifies this value backward to generate the test sequence. In other words, the BACK algorithm preselects a sensitized primary output to work with as shown in Figure 7; while the EBT method preselects a sensitized path to work with as shown in Figure 5. The whole process of the BACK algorithm is only a backward justification process as the EBT method. Thus, there is no memory problem in

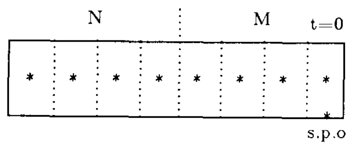

Figure 7.

the BACK algorithm. Moreover, because the number of primary outputs is small, the BACK algorithm can try all the possible primary outputs. Thus, given enough time, all the testable faults of any initializable sequential circuits [10] can be detected by the BACK algorithm. It should be noted that the sensitized paths needed in Figure 3 will be created automatically in the BACK algorithm when the justification of the sensitized primary output is completed.

In order to accurately predict which primary output must have sensitized value, a new testability measurement called ***drivability*** is proposed for the BACK algorithm. The drivability measurement in the BACK algorithm is corresponding to the observability measurement [11] in other algorithms. The observability of one line is measured by the difficulty of observing the fault effect at this line from the primary outputs. The drivability of one line is measured by the difficulty of driving the fault effect from the fault site to this line. Thus, the observability measurement is calculated from the primary outputs backward to the primary inputs; while the drivability measurement is calculated from the fault site forward to the primary outputs. Since the drivability measurement depends on the fault site, it has to be calculated for every target fault. Also, since the sensitized value at the preselected primary output can take value 1/0 or 0/1, we need two different drivability values for 1/0 and 0/1 respectively. To accurately calculate the drivability values, both the controllability measurements [11] of the good machine and of the bad machine are used. The bad machine controllability has to be calculated for every fault also. So far, we have found that the overhead to calculate the bad machine controllability and drivability measurement is insignificant compared to test generation time.

The procedure of the BACK algorithm can be summarized as follows.

1. Calculate the good machine controllability measurement.
2. Pick a target fault and calculate the bad machine controllability measurement and the drivability measurement.
3. Based on the drivability measurement, select a primary output to have a sensitized value (1/0 or 0/1). If no primary output is available, this fault is not testable and go to Step 2. Set the current time frame as $t=0$.
4. Justify all the values required at the current time frame. The detailed justification process will be explained in the following paragraphs. It should be noted that the previous state requirements will become values required in the previous time frame which will be justified later. If no conflict happens, go to Step 5. Otherwise, change the latest choice and set the current time frame to be the time when the choice was made and go to Step 4. If no choice left, go to Step 3 and try another sensitized primary output.
5. If the state requirement matches the initial state, stop. If there is no previous state requirements, stop also. Otherwise, set the previous time frame as the current time frame and go to Step 4.

To speed up line value justification process, the SPLIT circuit model [12] is used. The SPLIT model is an enhanced 9-valued model [3]. In the SPLIT model, each line has three values. They are ***good value***: the value at the good machine, ***bad value***: the value at the bad machine, and ***relation***: the relation between the good machine and the bad machine. The good value and the bad value can be 0, 1, X (don't care), and Z (high impedance). High impedance value is used to avoid generating vectors that cause bus conflicts and is only used around tri-state gates and bus nodes. The relation can be UNKNOWN, DIFFERENCE (good value and bad value must be different), and EQUIVALENCE (good value and bad value must be the same). With the values of the good machine and the bad machine separately recorded, the justification is done for these two machines separately. This is simpler with smaller search space than justifying these two machines simultaneously as in the 9-valued model [3-5]. Furthermore, with the relation recorded, the choices of two machines will be more restricted which means the search space will be further reduced.

In [3-5], only the controllability measurement is used as the choice guidance during justification process. However, using this guidance to justify sensitized values will guide toward the primary inputs not toward the fault site. To solve this problem, the drivability measurement should be used instead. In the BACK algorithm, two testability measurements are used for justification guidance: ***If the relation is DIFFERENCE, use the drivability measurement; otherwise, use the controllability measurement.***

IV. EXPERIMENTAL RESULTS

In AT&T, a sequential test generator STG was implemented using the EBT method in 1985 [5]. Recently, STG was modified to use the BACK algorithm and the SPLIT model and is called NSTG. In [12], I had another implementation which uses the D-algorithm and the SPLIT model to show that it has better performance than the PODEM algorithm with the 5-valued model for combinational circuits. However, that implementation cannot be used for sequential circuits. All three implementations used SCOAP testability measurement. The characteristics of experimental circuits are shown in Table I. Here, io pins are either input pins or output pins but not both at one time. To avoid conflicts at io pins and bus nodes, high impedance value should be used. The first five circuits are Brglez-Fujiwara combinational benchmark circuits [13]. The other circuits are AT&T sequential circuits. The experiment was done on a CONVEX C-1 computer which is a 4 MIPS machine.

Table II shows the detailed results of NSTG. In order to fairly compare test generation performance, no fault simulator was used and every non-equivalent stuck-at fault was tried. The initial state for every fault of sequential circuits is assumed to be unknown. The CPU time limit was set to 2 seconds per fault. Efficiency is calculated as the percentage of the sum of detected and untestable faults divided by the total faults to show the efficiency of test generators. Since no fault simulator was used here, the real fault coverage is not known.

Table III shows the comparison of efficiency and time. It is clear that NSTG which uses the BACK algorithm with the SPLIT model has the best performance. STG which uses the EBT method has the worst performance. For circuit c499 and c880, the BACK algorithm spent more time than the D-algorithm did. It shows that the overhaed to calculate drivability and bad machine controllability for every fault could be high for easily testable circuits. However, for other circuits, the overall effect of this overhead increases efficiency and reduces total time.

Table I.
Characteristics of Circuits

Circuit	gates	pi	po	io	bus	ff
c432	160	36	7	0	0	0
c499	202	41	32	0	0	0
c880	383	60	26	0	0	0
c1355	546	41	32	0	0	0
c1908	880	33	25	0	0	0
s103	69	8	4	0	0	4
s157	105	10	4	4	0	4
s181	133	12	4	6	0	4
s455	395	21	15	0	0	33
s1197	780	13	13	0	12	39

Table II.
NSTG Results

Circuit	Faults	Limit	Time	Detected	Untestable	Dropped	Efficiency
c432	524	2.00	41.76	520	1	3	99.43
c499	758	2.00	90.96	750	8	0	100.00
c880	942	2.00	56.52	942	0	0	100.00
c1355	1574	2.00	377.76	1566	8	0	100.00
c1908	1879	2.00	356.63	1870	7	2	99.89
s103	127	2.00	12.36	127	0	0	100.00
s157	181	2.00	16.55	169	11	1	99.45
s181	193	2.00	13.36	155	38	0	100.00
s455	408	2.00	138.04	314	68	26	93.63
s1197	1063	2.00	570.84	822	129	112	89.46

V. CONCLUSIONS

A new test generation algorithm for sequential circuits called BACK was presented. The BACK algorithm has been implemented and used very successfully for AT&T sequential circuits. Compared with the D-algorithm, the PODEM algorithm, and the EBT method, the BACK algorithm not only requires lesser run-time memory but also is more efficient. Since the performance of the BACK algorithm is heavily affected by the sensitized primary output prediction which is based on the drivability measurement, the BACK algorithm needs accurate testability measurement. The effect of other testability measurement to the BACK algorithm needs further investigation.

REFERENCES

[1] J. P. Roth, W. G. Bouricius, and P. R. Schneider, "Programmed algorithms to compute tests to detect and distinguish between failures in logic circuits," *IEEE Trans. Comput.*, Vol. C-16, pp. 567-579, Oct. 1967.

[2] G. R. Putzolu and J. P. Roth, "A heuristic algorithm for the testing of asynchronous circuits," *IEEE Trans. Comput.*, Vol. C-20, pp. 639-647, June, 1971.

[3] P. Muth, "A nine-valued circuit model for test generation," *IEEE Trans. Comput.*, Vol. C-25, pp. 630-636, June 1976.

[4] R. Marlett, "EBT, a comprehensive test generation technique for highly sequential circuits," *15th Design Automation Conference*, Las Vegas, NV, June 1978, pp. 332-339.

[5] S. Mallela and S. Wu, "A sequential circuit test generation system," *International Test Conference*, pp. 57-61, 1985.

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

[7] H. Fujiwara and T. Shimono, "On the acceleration of test generation algorithms," *IEEE Trans. Comput.*, Vol. C-32, pp. 1137-1144, Dec. 1983.

[8] M. Abramovici, J. J. Kulikowski, P. R. Menon and D. T. Miller, "Test generation in LAMP2: concepts and algorithms," *International Test Conference*, pp. 49-56, 1985.

[9] H. -K. T. Ma, S. Devadas, A. R. Newton and A. Sangiovanni-Vincentelli, "Test generation for sequential finite state machines," *International Conference on CAD*, pp. 288-291, 1987.

[10] A. Miczo, "The sequential ATPG: a theoretical limit," *Proceedings of 1983 International Test Conference*, pp. 143-147, Philadelphia, PA, October 1983.

[11] L. H. Goldstein, "Controllability/observability analysis for digital circuits," *IEEE Trans. Circuit Syst.*, Vol. CAS-26, pp. 685-693, Sept. 1979.

[12] W. -T. Cheng, "SPLIT circuit model for test generation," *Proceedings of 25th Design Automation Conference*, pp. 96-101, Anaheim, CA, June, 1988.

[13] F. Brglez and H. Fujiwara, "A neutral netlist of 10 combinational benchmark circuits and a target translator in FORTRAN," *Proc. 1985 IEEE Int. Symp. Circuits & Systems (ISCAS)*, Kyoto, Japan, pp. 663-698, 1985.

Table III.
Comparison Results

	Efficiency			Time		
Circuit	EBT 9-V	BACK SPLIT	D-ALG SPLIT	EBT 9-V	BACK SPLIT	D-ALG SPLIT
c432	82.25	99.43	98.66	262.29	41.76	42.94
c499	86.81	100.00	100.00	828.13	90.96	74.05
c880	100.00	100.00	100.00	77.97	56.52	32.18
c1355	79.99	100.00	91.87	1314.85	377.76	634.28
c1908	97.76	99.89	95.64	705.21	356.63	383.32
s103	94.49	100.00	NA	35.46	12.36	NA
s157	89.27	99.45	NA	38.04	16.55	NA
s181	81.77	100.00	NA	43.82	13.36	NA
s455	NA	93.63	NA	NA	138.04	NA
s1197	NA	89.46	NA	NA	570.84	NA