SPLIT CIRCUIT MODEL FOR TEST GENERATION

Wu-Tung Cheng

AT&T Engineering Research Center
Princeton, NJ 08540
Tel. (609) 639-2422

ABSTRACT

Over the years, the D-algorithm has been successfully used to generate tests for sequential circuits and combinational circuits. There are 5-valued and 9-valued circuit models used for the D-algorithm. The disadvantage of a model with lower value count is its inability to assign a more precise value for a test generation requirement without some undue assumptions or decisions which may cause backtracks or even may find no test for testable faults. However, the availability of more values increases the search space and makes enumeration more complicated. In this paper, we will present a new circuit model SPLIT which is a modified 9-valued circuit model. SPLIT has the precision of the 9-valued model and the simplicity of the 5-valued model, such that the D-algorithm will have better performance using the SPLIT model than using the 5-valued model or the 9-valued model.

I. INTRODUCTION

Over the years, the D-algorithm has been successfully used to generate tests for single stuck-at faults in combinational circuits [1-3] and in sequential circuits [4-9]. In the process of test generation, the D-algorithm uses either the 5-valued model or the 9-valued model, as shown in Table I, to represent the circuit status. In Table I, i/j means the good machine (fault-free circuit) has value i and the bad machine (circuit with assumed faults) has value j. For sequential circuits, a single fault behaves like multiple faults for its repeated fault effects through time. In [6], Muth showed that the D-algorithm using the 5-valued model [4,5] cannot take into account these repeated effects of a single fault, such that it may not find any test, although one exists. To overcome this problem, the 9-valued model has to be used.

For combinational circuits, the advantage of the 9-valued model is its ability to assign a more precise value for a test generation requirement, thereby avoiding some undue assumptions or decisions which may cause backtracks. However, the availability of more values increases the search space and makes enumeration more complicated. In this paper, we will discuss in detail the pros and cons of the 5-valued model and the 9-valued model. To keep the pros and get rid of the cons of both models, a new circuit model SPLIT which is a modified 9-valued circuit model will be presented. SPLIT has the precision of the 9-valued model and the simplicity of the 5-valued model, such that the D-algorithm will have better performance using the SPLIT model than using the 5-valued model or the 9-valued model.

We also developed a new test generation algorithm called the BACK algorithm for sequential circuits [10]. The BACK algorithm is a modified D-algorithm to combine the advantages of the D-algorithm [4-6] and the Extended Backtrace method [7-9] for sequential circuits. In this paper, we will focus on combinational test generation only, therefore, the well-known D-algorithm instead of the BACK algorithm is used to show the advantages of the SPLIT model. However, we will present the results of the SPLIT model with the D-algorithm and with the BACK algorithm. The detailed description of the BACK algorithm is presented in [10].

This paper is organized as follows: In Section II, we present the notation and definitions used in this paper. In Section III, we present the detailed comparison between the 5-valued model and the 9-valued model. In Section IV, the SPLIT circuit model will be presented. In Section V, experimental results will be presented. Concluding remarks are presented in Section VI.

II. Notation and Definitions

Here, we use single stuck-at fault model and gate-level circuit description. We describe a circuit as a set of *nodes* and *lines*. Nodes represent gate nodes, primary input nodes and primary output nodes. Lines are the edges that connect nodes together. In all the figures of this paper, we label nodes without labeling lines. Therefore, if we say one node is faulty, we mean the output of this node is faulty. If a node has more than one fanout branch, and the fault is at a fanout branch, then we say the fault is at a line between two nodes.

The D-algorithm has two basic operations: *C− operation* and *D− operation*. When a test is searched for a target fault, D-operation is invoked to select a single or multiple paths such that the sensitized value (0/1 or 1/0) at the fault site can be driven forward till it reaches any primary output to be observed. This operation is also called the *forward fault sensitization*. A *sensitized path* is a path which propagates sensitized value from the fault-site to a primary output. C-operation is employed to find a primary input pattern (test pattern) that will realize all the necessary gate input values on sensitized paths. C-operation is done in a backward mechanism. This operation is also called the *backward line justification*. D-operation is processed in several steps started from the fault-site. Each step is named a *D− drive*. Each D-drive drives the sensitized value closer to primary outputs. All the input nodes with values required in a D-drive are called *C− nodes*. To justify the required value of a C-node is called a C-drive. Using C-drives iteratively, all the C-nodes will be justified backward till the primary inputs are reached.

Table I.

5-V	9-V
0	0/0
1	1/1
D	1/0
$\bar{D}$	0/1
X	X/X
	$0/X$
	$X/0$
	$1/X$
	$X/1$

Basically, every D-drive and C-drive involves a decision making process. If a conflict situation happens, it is necessary to return the whole process to the previous decision point and try another choice. This is called a *backtrack*. To reduce the number of backtracks, we need good testability measurement to guide the decision making in D-operation and C-operation [11-13]. However, no practical testability measurement can completely avoid making bad choice, therefore, *implication* is very important. When node values are assigned by C-drives or D-drives, some other node values can also be determined by forward implication and backward implication. With more node values decided, conflicts are detected earlier and the number of choices for C-drives and D-drives becomes smaller, such that the number of backtracks is reduced.

III. The 5-Valued Model and The 9-Valued Model

Before Roth presented the D-algorithm, there was a test generation method called *single path sensitization* [14]. Intuitively, to detect a fault we need a path to propagate the sensitized value at the fault site to any primary output. The idea behind single path sensitization is that only a single sensitized path is selected explicitly to propagate the sensitized value until a successful one is found or all paths are exhausted. Unfortunately, single path sensitization is incomplete in the 5-valued circuit model. For example, in Figure 1 [15], there is a stuck-at-0 fault at node f. If single path sensitization is used, the sensitized path will be either the path through nodes i and l or the path through nodes j and l. If the path through nodes i and l is selected, then nodes a, h, j and k (C-nodes) must have value 0. With implication, input nodes a, b and c must have value 0, as shown in Figure 1. With further implication, whatever value is assigned to node d, there is a conflict either at node j or node k. If we try another path through nodes j and l, it will fail by symmetry. However, this fault is detectable if multiple sensitized paths are selected as shown in Figure 2. With careful observation, the reason why single path sensitization is incomplete is the inadequacy of the 5-valued model. In Figure 1, in order to propagate the sensitized value 1/0 to node l, actually nodes h, j and k only need to have value 0 at the good machine and don't care value at the bad machine. However, with only 5 values available, this situation cannot be expressed and a bad decision was made to assign 0/0 at nodes j which later blocked the needed sensitized path through nodes j in Figure 2. Therefore, to be complete in the 5-valued model, D-operation in the D-algorithm has to explicitly find all the sensitized paths needed otherwise they may be blocked by C-nodes. This is called *multiple path sensitization*. There is no such problem using single path sensitization in the 9-valued model. In the previous example, the 9-valued model will assign value $0/X$ at nodes h, j and k to avoid

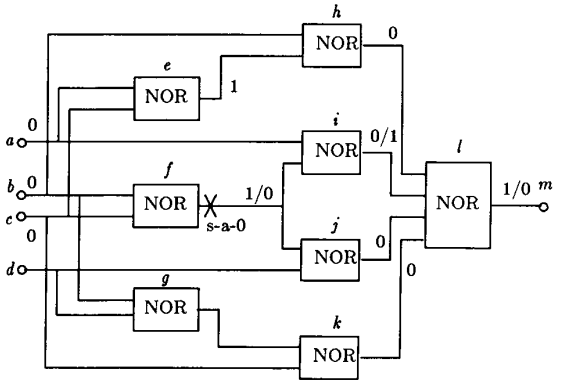

Figure 1.

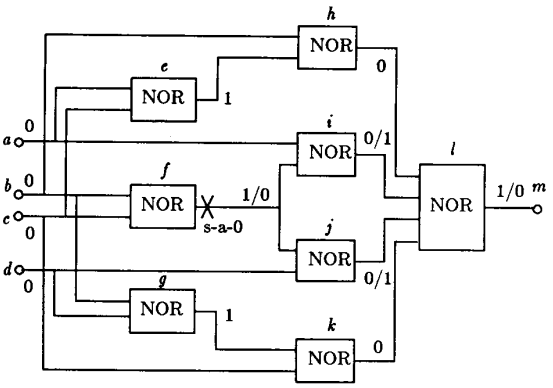

Figure 2.

any undue decision. With this assignment, interested readers may find the needed sensitized path through nodes j can be generated implicitly without explicitly selecting it in D-operation.

Obviously, the linear search space for single path sensitization is smaller than the exponential search space for multiple path sensitization. Therefore, in D-operation, the 9-valued model is much faster than the 5-valued model.

The another advantage of using the 9-valued model is in implication. In Figure 3, assume that the sensitized value 1/0 reaches node a. In the 9-valued model, node d has value 0/0 already; while in the 5-valued model, node d still has value x.

However, comparing to the 5-valued model, the 9-valued model has problems in C-operation. In the 5-valued model, only nonsensitized values are processed in C-operation. In the 9-valued model, to have the capability to implicitly derive sensitized paths needed, sensitized values need to be processed in C-operation. Therefore, in C-operation, all 9 values have to be used in the 9-valued model, while only 3 values are used in the 5-valued model. Thus, the 9-valued model has the following problems:

1. In a C-drive, to justify an N-input gate's output value, there are N^2 choices in the 9-valued model instead of N choices in the 5-valued model. For example, to justify a 2-input AND gate to have output value 0/0, 4 choices (0/0,X/X) (X/X,0/0) (0/X,X/0) (X/0,0/X) are available for its inputs. However, in the 5-value model, there are only 2 choices (0,X) and (X,0). Therefore, the search space in the 9-valued model is bigger.
2. No simple testability measurement is able to rank these N^2 choices. Thus, it is difficult to make a right choice.
3. Because more values are involved, the tables used in the 9-value model are much bigger than the tables used in the 5-valued model.

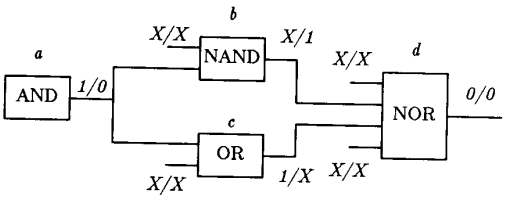

Figure 3. Forward implication.

4. It is possible that a node which is not reachable from the fault site can be assigned with a sensitized value which can never be justified. However, this unjustifiable situation cannot be detected until after several C-drives when we find some primary inputs have sensitized values. This problem was first pointed out in [3].

To overcome these problems, a new circuit model called SPLIT was created. The SPLIT circuit model will be presented in the next section.

IV. The SPLIT Circuit Model

We use circuit structure to further explain the problems of the 9-valued model. Basically, the 9-valued model views the good machine and the bad machine as two independent machines connected at the primary inputs only. Therefore, the whole system is viewed as a big circuit composed of two subcircuits connected at the primary inputs, as shown in Figure 4. Unfortunately, to save memory for circuit status the 9-valued model combines the values in these two subcircuits together, such that in C-operation two independent subcircuits have to be handled ***simultaneously***. The problem is it may fruitlessly remake the decision of one subcircuit to resolve the conflict of the other subcircuit, such that the number of backtracks is increased. It disobeys the general rule that ***two independent subcircuits should be treated separately in C-operation***. To overcome the problem, we split the values in the 9-valued model into two sets of values, one for the good machine and the other for the bad machine such that we can treat them separately in C-operation. With the values of the two machines split, Problem 1, 2 and 3 in the previous section is solved for the following reasons.

1. To justify an N-input gate's output value in one machine, there are at most N choices as in the 5-valued model. Because we justify each machine separately, we can avoid remaking the decision of one machine to resolve the conflict of the other machine.
2. Since each C-drive is for one machine only, the choice selection can be based on the testability of each machine as in the 5-valued model.
3. Since each C-drive involves 3 values only, the tables used will have the same size as the tables used for the 5-valued model.

Furthermore, the idea of ***default value*** mentioned in [2] is used in the SPLIT model and produces much better result. For example, there are two choices (0,X) (X,0) to justify a value 0 on the output of a 2-input AND gate in one machine. If the first choice (0,X) failed, the second choice would be (X,0). However, (0,0) was included in (0,X) choice already, thus, the second choice is (1,0) by default. According to the experimental results in [16], by using default value, the number of backtracks is reduced significantly.

The circuit structure in Fig. 4 can explain Problem 4 of the 9-valued model also. Actually, the good machine and the bad machine are not completely independent. They are different only at the area affected by the fault-site. This affected area is dynamically reduced during test generation process. Identifying this area precisely improves implication process to detect conflicts earlier for C-operation which solves Problem 4, and to reduce the search space for D-operation. In the SPLIT model, ***relation information*** is included in the circuit status to dynamically identify this area. We define nodes which have different values at the two machine as having ***difference relation*** while nodes which have same values at the two machines as having ***equivalence relation***. Thus, only nodes with difference relation and unknown relation are within this area. At first, the fault-site is marked as having difference relation and all the nodes which are not reachable from the fault site are marked as having equivalence relation even their values are unknown. The implication of the

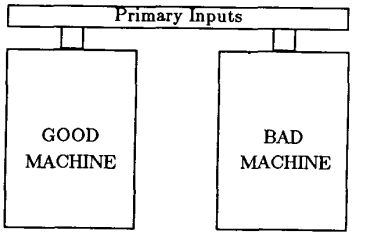

Figure 4. The 9-valued circuit model.

relation information processes as follows.

1. If all the inputs of one node have equivalence relation, then this node must have equivalence relation unless it is the fault site.
2. If one node has difference relation, then at least one of its inputs must have difference relation, unless this node is the fault site.

In addition to enhancing the effectiveness of implication, the relation information helps our D-operation also.

1. For D-drives, we propagate difference relation, not the sensitized value, through the single sensitized path selected. Therefore, if one primary output has difference relation already, even its value is still unknown, D-operation is not required any more. One example will be given later.
2. If the choice of a node to have difference relation causes conflicts, then this node must have equivalence relation while trying other choices. This kind of ***default relation*** has the same effect as the default value mentioned before.
3. In the process of test generation, if all the primary outputs have equivalence relation, a fruitless situation has been encountered. This is similar to the ***X-path check*** in PODEM [17].

Furthermore, the relation information covers the extra value added in the 10-valued model [18] to improve the test generation performance for circuits with XOR gates. For XOR gates, more implication is done as follows.

1. If two inputs of an XOR gate have difference relation and equivalence relation respectively, then it must have difference relation; while if both of its two inputs have difference relation or equivalence relation, then it must have equivalence relation.
2. If an XOR has difference relation and one of its inputs has difference (equivalence) relation, then the other input must have equivalence (difference) relation.
3. If an XOR has equivalence relation and one of its inputs has difference (equivalence) relation, then the other input must have difference (equivalence) relation also.

The redundant circuit used by Goel in [17] to show the ineffectiveness of the D-algorithm happens to be a good example to show the effectiveness of the relation information for XOR gates. In Figure 5 and 6, 'DI' stands for difference relation and 'EQ' stands for equivalence relation. In Figure 5, assuming the fault is node g stuck at 0, then node g has difference relation and all the other nodes except nodes n and o have equivalence relation. With implication, nodes n and o will have difference relation and node g has value 1 at the good machine and nodes e and f have value 1 at both machines. Thus, test vector 11$XXXX$ is found right away, even the value at node o is still unknown. In Figure 6, assuming the fault is node a stuck at 0, then node a has difference relation

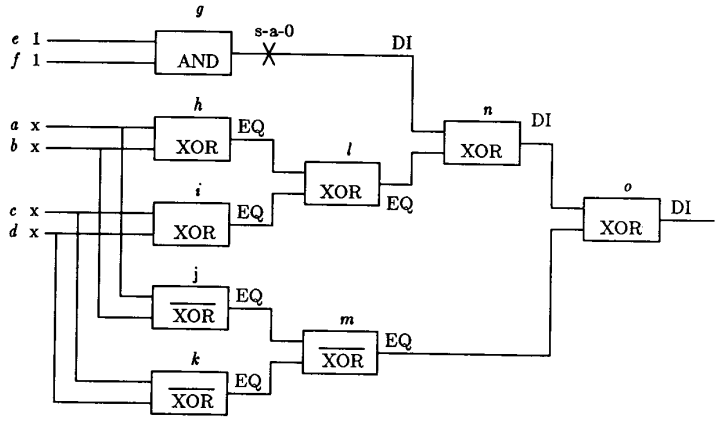

Figure 5.

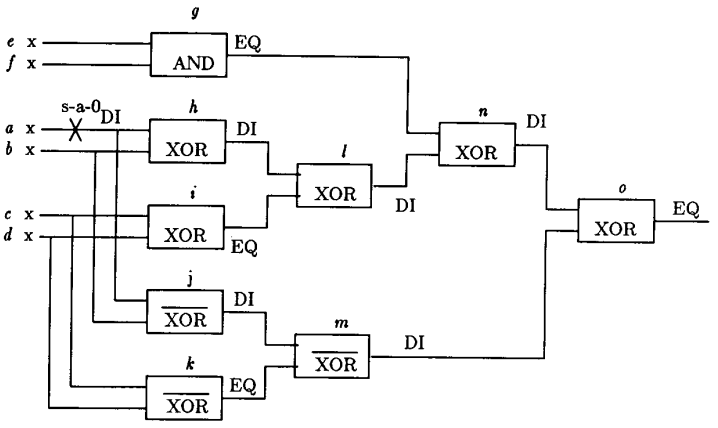

Figure 6.

and nodes b, c, d, e, f, g, i, and k have equivalence relation. With implication, nodes h, j, m, l and n have difference relation and node o has equivalence relation which means this fault is untestable. The test vectors generated for this circuit will be more compact with the SPLIT model than those with other models.

Besides improving the performance of test generation, the SPLIT model is easy to expand. For example, if circuits contains bus nodes, high impedance value has to be used to avoid generating test vectors which cause bus conflicts. To include high impedance value in the circuit status, the 5-valued model becomes a 13-valued model [19]; while the 9-valued model becomes a 16-valued model [9]. The dramatical increase of signal values in the circuit status makes not only the test generation operation more complicated but also the search space much bigger. In the SPLIT model, to include high impedance value in the circuit status, only one extra value is added in the circuit status for the good machine and for the bad machine. With this arrangement, the overhead of adding extra values in the SPLIT model is always minimum.

Finally, we like to point out the memory requirement of the SPLIT model is actually smaller than other models. The most significant memory required during test generation is the circuit status stored at every decision point for later recovery. Although the circuit status of the SPLIT model is 3 times larger than those of other models, the number of decision points of the SPLIT model is much smaller than those of other models. The number of decision points is reduced by using more information in the circuit status to do more complete implication. The reduction is bigger for bigger circuits. For sequential circuits, the reduction is even bigger. In our experiment, around 1000 gate combinational circuits, the reduction achieves 3 times already.

IV. EXPERIMENTAL RESULTS

The D-algorithm with the SPLIT model was implemented in C language. The SCOAP testability measurement [11] was used. The Brglez-Fujiwara circuits [20] are used here for benchmark. The results on a CONVEX system C-1 which is a 4 MIPS machine are shown in Table II. To unbiasedly compare test generation performance, no fault simulator was used and every non-equivalent stuck-at fault was tried. The time limit is set to 2 seconds. The total time is the accumulation of the user time and the system time spent for all the faults. Efficiency is calculated as the percentage of the sum of detected and untestable faults divided by the total faults.

Table II.
The SPLIT model and the D-algorithm

Circuit	Faults	Limit	Time(s)	Detected	Untestable	Dropped	Efficiency
c432	524	2.00	38.68	516	1	7	98.66
c499	758	2.00	53.49	750	8	0	100.00
c880	942	2.00	29.51	942	0	0	100.00
c1355	1574	2.00	461.43	1438	8	128	91.87
c1908	1879	2.00	376.96	1790	7	82	95.64
c2670	2747	2.00	330.32	2618	86	43	98.43
c3540	3428	2.00	722.64	3229	137	62	98.19
c5315	5350	2.00	749.38	5291	59	0	100.00
c6288	7744	2.00	4559.84	7479	34	231	97.02
c7552	7536	2.00	2782.84	7229	76	245	96.75

Table III.

	Efficiency			Time/fault (ms)		
Circuit	D-ALG SPLIT	PODEM 2 5-V	PODEM 10 5-V	D-ALG SPLIT	PODEM 2 5-V	PODEM 10 5-V
c432	98.66	98.61	98.84	73.82	137.72	209.05
c499	100.00	91.58	91.78	70.57	321.68	949.99
c880	100.00	100.00	100.00	31.33	98.43	79.56
c1355	91.87	97.71	98.01	293.16	346.43	443.44
c1908	95.64	93.11	93.13	200.62	371.62	872.29
c2670	98.43	91.85	93.78	120.25	273.08	545.21
c3540	98.19	83.35	89.70	210.87	729.19	1616.27
c5315	100.00	98.91	99.15	140.07	271.63	290.82
c6288	97.02	88.45	91.82	588.82	999.51	1781.33
c7552	96.75	96.98	97.48	369.27	505.48	698.54
Avg.	97.66	94.06	95.37	209.88	405.48	748.65

Instead of implementing the D-algorithm with other circuit models to do the comparison, we compare our results with the results reported in [21]. In [21], the PODEM algorithm was used with the 5-valued model. Those results were run on SUN 3/50 which is about 2.5 MIPS, as shown in Table III. PODEM 2 and PODEM 10 are the same program except they use different time limits. PODEM 2 uses 2 seconds time limit and PODEM 10 uses 10 seconds time limit. No fault simulator was used there also. Unfortunately, they executed their program for every stuck-at faults including equivalent faults. To be fair, we use efficiency and time per fault for comparison. The results shows the D-algorithm with the SPLIT model is a little bit better than the PODEM algorithm with the 5-valued model. Since it is a general belief that the PODEM algorithm with the 5-valued model has much better performance than the D-algorithm with the 5-valued model or the 9-valued model does, we can claim that the D-algorithm with the SPLIT model has better performance than the D-algorithm with the 5-valued model or the 9-valued model does.

We also implemented the SPLIT model with the new BACK algorithm for sequential circuits. Table IV shows the characteristics of the experimental sequential circuits. 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 has to be used. Table V shows the results for 5 Brglez-Fujiwara combinational circuits and 5 sequential circuits. For sequential circuits, state initialization is done for every fault. Table VI compares the results of three implementations run on CONVEX C-1. The implementation of the 9-valued model and the Extended Backtrace method was presented in [9]. Our implementation of the SPLIT model and the D-algorithm cannot run for sequential

Table IV.
Characteristics of experimental sequential circuits

Circuit	gates	pi	po	io	bus	ff
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

circuits. The results show that the SPLIT model with the BACK algorithm has the best performance. The detailed of the BACK algorithm is presented in [10].

VI. CONCLUSIONS

In this paper, a new circuit model SPLIT was presented. Basically, we divided circuit status into three sets for the good machine, the bad machine and their relation. With these three sets of information, the SPLIT model achieves more precision and more complete implication than the 9-valued model and keep the same simplicity as the 5-valued model. Although each circuit status needs more memory, the total memory required is smaller. The benchmark results showed that the D-algorithm has better performance with the SPLIT model than with the 5-valued model or the 9-valued model. In this paper, we only discussed the test generation problems for combinational circuits. The test generation problems for sequential circuits are presented in [10].

Table V.
The SPLIT model and the BACK algorithm

Circuit	Faults	Limit	Time(s)	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

Table VI.

	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

ACKNOWLEDGMENT

The author would like to thank Scott Davidson for his support and encouragement to write this paper and to express his deep gratitude to Tapan J. Chakraborty and Shiangling Wu for coding portion of experimental programs. The help of Susheel J. Chandra and Janak H. Patel for using their results is also appreciated.

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] C. W. Cha, W. E. Donath and F. Ozguner, "9-V algorithm for test pattern generation of combinational digital circuits," *IEEE Trans. Comput.*, Vol. C-27, pp. 193-200, March 1978.

[3] M. Murakami, N. Shiraki and K. Hirakawa, "Logic verification and test generation for LSI circuits," *International Test Conference*, pp. 467-472, 1980.

[4] H. Kubo, "A procedure for generating test sequences to detect sequential circuit failures," *NEC Research & Development*, No. 12, pp. 69-78, Oct. 1968.

[5] 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.

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

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

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

[9] R. Marlett, "An efficient test generation system for sequential circuits," *23rd Design Automation Conference*, Las Vegas, NV, June 1986, pp. 250-256.

[10] W. -T. Cheng, "The BACK algorithm for sequential test generation," submitted to *International Conference on Computer Design*, 1988.

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

[12] R. G. Bennetts, *Design of Testable Logic Circuits*. Addison-Wesley Publishing Company, 1984.

[13] F. Brglez, P. Pownall, and P. Hum, "Application of testability analysis: from ATPG to critical path tracing," *IEEE International Test Conference*, pp. 705-712, 1984.

[14] H. Y. Chang, E. Manning and G. Metze, *Fault diagnosis of digital systems*. John Wiley & Sons, 1970.

[15] P. R. Schneider, "On the necessity to examine D-chains in diagnostic test generation - an example," *IBM Journal of Research & Development*, Vol. 11, p. 114.

[16] R-S. Wei and A. Sangiovanni-Vincentelli, "New front-end and line justification algorithm for automatic test generation," *International Test Conference*, pp. 121-128, 1986.

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

[18] Y. Takamatsu and K. Kinoshita, "An efficient test generation method by 10-V algorithm," *Proc. 1985 IEEE Int. Symp. Circuits & Systems (ISCAS)*, Kyoto, Japan, pp. 679-682, 1985.

[19] N. Itazaki and K. Kinoshita, "Test pattern generation for circuits with three-state modules by improved Z-algorithm," *International Test Conference*, pp. 105-112, 1986.

[20] 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.

[21] S. J. Chandra and J. H. Patel, "Experimental evaluation of testability measures for test generation," submitted to *IEEE Trans. on CAD*.