Simulation of VICTOR Algorithm for Fault-Diagnosis of Digital Circuits

Tariq Jamil and Iftaquaruddin Mohammed

***Abstract*—Of a multitude of algorithms used for fault diagnosis and testing of digital circuits, VICTOR stands out because of its multi-step approach to determine the test vectors needed for detection of a particular fault. In this paper, we have presented the procedure used by VICTOR to determine test patterns for a particular fault, first by hand-calculations and then with the help of a simulation program developed by us for this algorithm. The results of our simulation are consistent with those obtained manually which confirms the veracity and usefulness of our developed software.**

***Index Terms*—Fault diagnosis, testing, algorithm, digital circuit, VICTOR.**

I. Introduction

Fault diagnosis and testing are important requirements for any given digital circuit to be used in engineering applications. A given circuit is said to be testable with respect to a fault if a well-specified procedure can be utilized to detect that fault [1]. Testability has two important components, namely *controllability* and *observability* [2]. How one can control a certain wire of a digital circuit to exhibit logic 0 or logic 1 on that wire provides us measure of the wire's controllability. Observability refers to the ease with which it is possible to observe logic 0 or logic 1 on that particular wire. TMEAS (Testability Measurement) and SCOAP (Sandia Controllability Observability Analysis Program) are two famous algorithms which determine the testability of a digital circuit using controllability and observability values of the various wires contained within the circuit. Details about these algorithms can be found in [2]-[4]. VICTOR (VLSI Identifier of Controllability, Testability, Observability, and Redundancy) was a FORTRAN program developed at the Electronics Research Laboratory in the University of California, Berkeley and is a based on a very unique approach to determine the test vectors for a given digital circuit [5]. This paper is organized as follows: In Section II, we present a summary of VICTOR algorithm (as applied to a test circuit) which is followed, in Section III, by the output obtained by running this algorithm on the same test circuit using our developed software. Conclusions are presented in Section IV, followed by acknowledgment and references.

Manuscript received March 16, 2014; revised May 5, 2014. This work was supported in part by Sultan Qaboos University (Oman) through internal grant: IG/ENG/ECED/10/01.

The authors are with the Department of Electrical and Computer Engineering at Sultan Qaboos University, Oman (e-mail: tjamil@squ.edu.om, iftaquar@squ.edu.om).

II. Summary of VICTOR Algorithm

VICTOR is a linear algorithm requiring four passes through the circuit-under-test: circuit leveling, controllability analysis, observability analysis, and test generation [6].

A. Circuit Leveling

VICTOR starts by marking and leveling the circuit. Each circuit node is marked with a unique name, where nodes are primary inputs, the outputs of every functional block, and all fan-out destinations. The circuit is leveled to identify the relative processing order of circuit nodes. All primary inputs and their fan-out nodes are assigned to level 0. A node other than the primary input node takes the level of the functional block that drives the node (a fan-out branch is assigned the level of its stem). The level of a functional block is defined as the level of its highest-level input node plus 1. Consider the circuit given in Fig. 1 [6]. It has five levels as follows:

Level 0: A, B, C, D, G, H;
Level 1: I, J, K, L;
Level 2: M;
Level 3: N;
Level 4: Z.

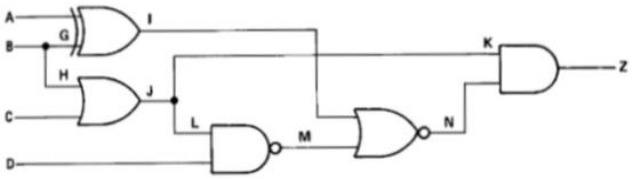

Fig. 1. Circuit-Under-Test (CUT) with levels identified.

B. Controllability Analysis

The second pass through the circuit computes the set-controllability (also called 1-controllability) and reset-controllability (also called 0-controllability) of each node. Node set and reset measures consist of triplets of three values, named *pattern*, *risk*, and *size*. For a given node V, the set and reset triplets are represented as:

$$V_{TS} = \{ V_s, \text{risk } (V_s), \text{size } (V_s)\}$$

$$V_{TR} = \{ V_r, \text{risk } (V_r), \text{size } (V_r)\}$$

V_s and V_r are the set and reset control patterns, a set of primary input values that are sufficient to force node V to a 1 and 0, respectively (or alternatively to provoke a stuck-at-0 and a stuck-at-1 fault at the node). All patterns in a circuit are of the same length, having one symbol for each primary

input. VICTOR uses four-valued logic, allowing any of the following four symbols in a pattern:

0 = logical 0 assignment

1 = logical 1 assignment

x = don't care assignment

= conflicting 0 and 1 assignment (called *clash*)

The values risk (V_s) and risk (V_r) are the set and reset risk, and heuristically measure the risk of re-convergence of fan-out nodes and, thus, the possibility of redundant faults. The values size (V_s) and size (V_r) are estimates of the number of primary input patterns that will either set or reset the node. The set and reset patterns are the essential control information generated by VICTOR. The risk and size measures in the triplets are used only to guide the selection of patterns.

Computing these controllability measures begins with the primary input nodes (at level 0) and flows forward by level until the primary outputs are reached. A primary input set (reset) pattern is a 1 (0) for the input itself and an *x* (don't care) for all other inputs.

The re-convergence risk value for a primary input with no fan-out is set at 1. However, where fan-out exists (as with input B in the Fig. 1), the risk value is the product of the number of fan-out nodes and the number of levels to the farthest primary output. Thus the set and reset risks of node B are both 2 × 4 = 8.

TABLE I: CUT SET TRIPLETS

Node	ABCD	Risk	Size
A	1*xxx*	1	1
B	*x*1*xx*	8	1
C	*xx*1*x*	1	1
D	*xxx*1	1	1
G	*x*1*xx*	8	1
H	*x*1*xx*	8	1
I	10*xx*	9	1
J	*xx*1*x*	7	2
K	*xx*1*x*	7	2
L	*xx*1*x*	7	2
M	*xxx*0	1	2
N	0011	17	9
Z	0011	24	18

TABLE II: CUT RESET TRIPLETS

Node	ABCD	Risk	Size
A	0*xxx*	1	1
B	*x*0*xx*	8	1
C	*xx*0*x*	1	1
D	*xxx*0	1	1
G	*x*0*xx*	8	1
H	*x*0*xx*	8	1
I	00*xx*	9	1
J	*x*00*x*	15	1
K	*x*00*x*	15	1
L	*x*00*x*	15	1
M	*xx*11	8	2
N	*xxx*0	1	3
Z	*xxx*0	1	4

Finally, the set and reset sizes for primary inputs are both 1 because an input node can be set or reset in only one way. Controllability triplets for the primary inputs and all other nodes of the example circuit have been derived and are listed in Tables I and II.

The controllability values for fan-out branches take the values for their driving node. Hence the set and reset triplets for the fan-out nodes G and H in the sample circuit are the same as those for node B.

In the propagation of controllability triplets through the circuit, logical elements are handled in level order. Triplets on the outputs of blocks are computed using one of two operations called pattern selection and pattern intersection.

Pattern selection defines an output pattern as the lowest-risk lowest-level input pattern. The risk of the output pattern is the same as the risk of the chosen input pattern. The size of the output pattern is the sum of the sizes of all input patterns. Pattern intersection defines an output pattern as the symbol by symbol intersection of the input patterns. The intersection operation is governed by Table III.

TABLE III: RULES GOVERNING INTERSECTION OPERATION

Intersection	**0**	**1**	***x***	**#**
0	0	#	0	#
1	#	1	1	#
x	0	1	*x*	#
#	#	#	#	#

The risk of an intersection output pattern is the sum of risks of the input patterns, and the size is the product of the input sizes. The choice of whether to do selection or intersection is governed by the function of the element. Crudely, one can think of selection as an OR operation and intersection as an AND operation.

In the circuit of Fig. 1, the order of the logical elements is XOR, OR, NAND, NOR, and AND. The first step is to select a set and reset pattern for XOR (node I). Node I can be set to 1 if either A is set and G is reset or vice versa. These two choices are both intersection operations, thus

$$I_s = \text{intersection } (A_s, G_r) \text{ or intersection } (A_r, G_s)$$

Similarly there are two choices for resetting *I*:

$$I_r = \text{intersection } (A_s, G_s) \text{ or intersection } (A_r, G_r)$$

When faced with a choice (either during pattern intersection or pattern selection), VICTOR chooses that pattern with the lowest risk. In the case of a tie, the lowest-level pattern is chosen. If the tie still exists, the choice is arbitrary. As stated earlier, the risk of a pattern resulting from pattern intersection is the sum of input risks. Let's consider the choices for setting node I. The resulting patterns and risks are:

$$I_s = \text{intersection } (A_s, G_r) = 10xx$$

$$\text{risk } (I_s) = \text{risk } (A_s) + \text{risk } (G_r) = 1 + 8 = 9$$

$$I_s = \text{intersection } (A_r, G_s) = 01xx$$

$$\text{risk } (I_s) = \text{risk } (A_r) + \text{risk } (G_s) = 1 + 8 = 9$$

Since both patterns have the same risk and since there is no difference in levels, the set pattern 10*xx* is arbitrarily chosen. Similarly, the reset pattern choices have the same risk and levels and 00*xx* is arbitrarily chosen. The size (for both set and reset) is the product of the input sizes or size (*I*) = size (*A*)

x size (G) = 1. Thus, the {pattern, risk, size} triplet for node I is: I_{TS} = {10xx, 9, 1} and I_{TR} = {00xx, 9, 1}.

As another example, let's take node J, the output of the OR gate, is an internal fan-out stem. Whenever a cell output is a fan-out stem, its risk is increased by adding to the computed risk the product of the fan-out count and the number of levels to the furthest primary output. An OR is set if any input is set, so that the set equation for J_s is select (H_s, C_s). Selection chooses the lowest risk input pattern, which in this case is C_s so that J_s becomes $xx1x$. The risk for C_s is 1, to which is added the product of the fan-out 2 and the distance 3. The size is the sum of the sizes of H_s and C_s. Thus J_{TS} = {$xx1x$, 7, 2}. Resetting node J requires both inputs at 0 or J_r = intersect(H_r, C_r). The intersection J_r = $x00x$, risk = 9+6=15, and size =1. The controllability triplets for the remaining logic gates are computed similarly.

C. Observability Analysis

Once the controllability triplets are completed for all nodes, VICTOR begins the third pass through the circuit by computing the observability of the nodes. The term used in VICTOR for observability is "monitor" and these measures are computed by starting at the primary outputs and working backward through the levelized cell-list. Monitor measures are again a triplet of pattern, risk, and size of each node. In the example circuit of Fig. 1, monitor analysis starts with output Z of the AND gate. An output is observable without any primary input contraints, so the appropriate pattern for Z is $xxxx$. An output has no risk and its size is 1. Hence the monitor triplet for Z is Z_{TM} = {$xxxx$, 0, 1}. The rules to derive the input monitor triplets from output triplets are called the monitor equations, and use the select and intersect concepts described earlier. In the sample circuit, the AND gate has monitor equations:

$$K_m = \text{intersect } (N_s, Z_m)$$

$$N_m = \text{intersect } (K_s, Z_m)$$

which result in

$$K_m = \text{intersect } (0011, xxxx) = 0011$$

$$N_m = \text{intersect } (xx1x, xxxx) = xx1x$$

Monitor risk and size computation generally follows the rules described earlier. Using these rules, the monitor triplets for K and N would be K_{TM} = {0011, 17, 9] and N_{TM} = {$xx1x$, 7, 2}. However, when a fan-out branch enters a cell, VICTOR computes a monitor risk adder that is the product of the fan-out and the number of levels from the fan-out stem to the cell output. In the example, the AND gate has an input K that is a fan-out point from stem node J. Since stem J is at level 1 and the AND gate output is at level 4, the risk for input K is 2x (4-1) = 6. This risk is added to the computed risk for the other cell input, node N. Thus, the monitor triplets for nodes K and N are:

$$K_{TM} = \{0011, 17, 9\}$$

$$N_{TM} = \{xx1x, 13, 2\}$$

The next cell is the NOR with monitor equations:

$$I_m = \text{intersect } (M_r, N_m)$$

$$M_m = \text{intersect } (I_r, N_m)$$

resulting in the triplets

$$I_{TM} = \{\text{xx11}, 21, 4\}$$

$$M_{TM} = \{001x, 22, 2\}$$

The NAND gate, processed next, also has a fan-out node as one input. The NAND monitor equations are:

$$D_m = \text{intersect } (L_s, M_m) = \text{intersect } (xx1x, 001x) = 001x$$

$$L_m = \text{intersect } (D_s, M_m) = \text{intersect } (xxx1, 001x) = 0011$$

Because L is a fan-out node, the risk adder is the product of the fan-out and the level difference between the fan-out node J and the cell output M, or 6 + 2 × (2-1) = 8, and is applied to the other cell input D. Without the fan-out, the risk at D is risk (L_s) +risk (M_m) = 29. With the adder, risk (D_m) becomes 37. The risk at L is 23. The size of D_m is 4 and of L_m is 2. Hence D_{TM}= {001x, 37, 4} and L_{TM} = {0011, 23, 2}.

Now, to calculate observability of a fan-out stem, VICTOR first computes the set-monitor and the reset-monitor intersections for each of its branches and then assigns to the stem the triplet of that branch whose set-monitor and reset-monitor patterns are both clash-free. In the example, nodes K and L are the branches from fan-out stem J. The set-monitor and reset-monitor intersections for K are:

$$\text{Intersect } (K_s, K_m) = \text{intersect } (xx1x, 0011) = 0011$$

and

$$\text{Intersect } (K_r, K_m) - \text{intersect } (x00x, 0011) = 00\#1$$

And for L are:

$$\text{Intersect } (L_s, L_m) = \text{intersect } (xx1x, 0011) = 0011$$

$$\text{Intersect } (L_r, L_m) = \text{intersect } (x00x, 0011) = 00\#1$$

Note that the reset-monitor intersection for both branches produces a clash in the third symbol. When all patterns have clashes, VICTOR sets the fan-out stem monitor to {##...#, 99999, 0}. Hence J_{TM} = {####, 99999, 0}. In a general network, however, there may be several clash-free patterns, in which case VICTOR takes the pattern that has the most don't cares (x). Observability of the remaining cells is computed in a similar fashion and monitor triplets for CUT are shown in Table IV.

TABLE IV: CUT MONITOR TRIPLETS

Node	ABCD	Risk	Size
A	x111	31	4
B	1x11	22	4
C	####	99999	0
D	001x	37	4
G	1x11	22	4
H	####	99999	0
I	xx11	21	4
J	####	99999	0
K	0011	17	9
L	0011	23	2
M	001x	22	2
N	$xx1x$	13	2
Z	$xxxx$	0	1

D. Test Generation

The set and reset patterns are primary input values that are sufficient to provoke a stuck-at-0 and a stuck-at-1 fault at the node. Similarly, the monitor patterns are sufficient input values to propagate a fault from a node to an observable output. The intersection of the nodal control and monitor patterns, then, can be a test for the node. Two tests are required for each node *V*:

V/0 test: intersect (V_s, V_m)

V/1 test: intersect (V_r, V_m)

The set, reset, and monitor triplets from Tables I, II, and III are intersected and, after intersection, produce *V*/0 and *V*/1 test patterns as shown in Table V.

TABLE V: CUT V/0 AND V/1 TEST PATTERNS

Node	V/0 Test	V/1 Test
	ABCD	ABCD
A	1111	0111
B	1111	1011
C	####	####
D	0011	0010
G	1111	1011
H	####	####
I	1011	0011
J	####	####
K	0011	00#1
L	0011	00#1
M	0010	0011
N	0011	*xx*10
Z	0011	*xxx*0

Note that eight of the node fault tests contain clashes (#) and are potential redundancies. Clashes occur in the monitor patterns for nodes *C*, *H*, and *J* and carry over to the tests. Nodes K and L are clash-free in the control and monitor patterns, but clashes appear in the *V*/1 test.

V/0 and V/1 patterns in which no clashes occur are valid test. From Table V, after test compaction, we find that there are five test patterns which will detect 18 of the possible 26 possible nodal faults: 0010, 0011, 0111, 1011, 1111.

III. SIMULATION RESULTS

The sample combinational circuit of Fig. 1 was re-labeled as shown in Fig. 2 so that it could be described in our software as: 1, 5, XOR, 7, 6, 3, OR, 8, 10, 4, NAND, 11, 7, 11, NOR, 12, 9, 12, AND, 13.

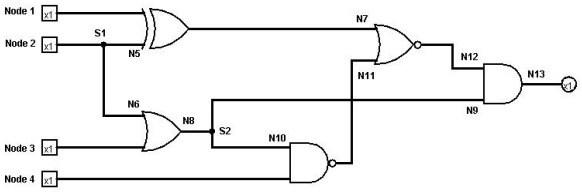

Fig. 2. Circuit-Under-Test (CUT) re-labeled for simulation.

The results of simulation obtained from our software are shown in Fig. 3 and Fig. 4 which are consistent with the manual calculations done for CUT.

VICTOR RESULTS									
	SET			RESET			MONITOR		
Node	ABCD	Risk	Size	ABCD	Risk	Size	ABCD	Risk	Size
1	1xxx	1	1	0xxx	1	1	x111	31	4
2	x1xx	8	1	x0xx	8	1	1x11	22	4
3	xx1x	1	1	xx0x	1	1	####	99999	0
4	xxx1	1	1	xxx0	1	1	001x	37	4
5	x1xx	8	1	x0xx	8	1	1x11	22	4
6	x1xx	8	1	x0xx	8	1	####	99999	0
7	10xx	9	1	00xx	9	1	xx11	21	4
8	xx1x	7	2	x00x	15	1	####	99999	0
9	xx1x	7	2	x00x	15	1	0011	17	9
10	xx1x	7	2	x00x	15	1	0011	23	2
11	xxx0	1	2	xx11	8	2	001x	22	2
12	0011	17	9	xxx0	1	3	xx1x	13	2
13	0011	24	18	xxx0	1	4	xxxx	0	1

Fig. 3. VICTOR simulation results output by simulation software.

VICTOR RESULTS					
	SET	RESET	MONITOR	V/0 Test	V/1 Test
Node	ABCD	ABCD	ABCD	ABCD	ABCD
1	1xxx	0xxx	x111	1111	0111
2	x1xx	x0xx	1x11	1111	1011
3	xx1x	xx0x	####	####	####
4	xxx1	xxx0	001x	0011	0010
5	x1xx	x0xx	1x11	1111	1011
6	x1xx	x0xx	####	####	####
7	10xx	00xx	xx11	1011	0011
8	xx1x	x00x	####	####	####
9	xx1x	x00x	0011	0011	00#1
10	xx1x	x00x	0011	0011	00#1
11	xxx0	xx11	001x	0010	0011
12	0011	xxx0	xx1x	0011	xx10
13	0011	xxx0	xxxx	0011	xxx0

Fig. 4. VICTOR test generation results output by simulation software.

IV. CONCLUSIONS

In this paper we have presented calculated results and simulation results obtained by running VICTOR algorithm on a circuit-under-test using a simulation software developed at SQU (Oman). The results are consistent and in conformity with the calculated values.

ACKNOWLEDGMENT

The support for the work presented in this paper was provided by Sultan Qaboos University (Oman) through internal grant: IG/ENG/ECED/10/01.

REFERENCES

[1] Chapter 3 testability measures. (2002). [Online]. Available: http://larc.ee.nthu.edu.tw/~cww/n/625/6250/03.pdf

[2] T. Jamil and I. Mohammed, "Transcription of algorithms used for fault-diagnosis of digital systems into computer programs," in *Proc. IEEE Southeastcon*, 2013, pp. 1-6.

[3] J. Grason, "TMEAS—a testability measurement program," in *Proc. IEEE/ACM Design Automation Conf. (DAC)*, 1979, pp. 156–161.

[4] L. H. Goldstein, "Controllability/observability analysis for digital circuits," *IEEE Trans. Circuits and Systems*, vol. 26, no. 9, pp. 685–693, Sept. 1979.

[5] I. M. Ratiu, A. Sangiovanni-Vincentelli, and D. O. Pederson, "VICTOR: a fast VLSI testability analysis program," in *Proc. Int. Test Conf. (ITC)*, Philadelphia, PA, Nov. 1982, pp. 397–401.
[6] P. H. Bardell, W. H. McAnney, and J. Savir, *Built-in Test for VLSI – Pseudorandom Techniques*, John Wiley and Sons, New York, 1987.

Tariq Jamil has been a faculty member in the Department of Electrical and Computer Engineering at Sultan Qaboos University (SQU, Oman) since 2000. Before joining SQU, he had been a lecturer at the University of New South Wales, Sydney (Australia) and the University of Tasmania, Launceston (Australia). Dr. Jamil holds a B.Sc. (Honors) degree in electrical engineering from the NWFP University of Engineering and Technology (Pakistan) and M.S. and Ph.D. degrees in computer engineering from the Florida Institute of Technology (USA).

He has authored three books, holds an Australian Innovation Patent on Complex Binary Associative Dataflow Processor, and has written about fifty research papers in refereed international conferences and journals. He has been a recipient of several research grants from the Australian Research Council and SQU. His research interests are in computer arithmetic, computer architecture, parallel processing, digital systems, and cryptography.

On account of his outstanding academic achievements and for contributions to activities related to the computing discipline, Dr. Jamil was awarded the IEEE Computer Society (USA)/Upsilon Pi Epsilon Honor Society Award for Academic Excellence (1996). He has served as a distinguished speaker in the IEEE Computer Society (USA) Distinguished Visitors Program (DVP) and his biography has been published in such renowned directories as Marquis's Who's Who in the World (USA), Who's Who in Science and Engineering (USA), and Dictionary of International Biography (UK). He is a senior member of IEEE (USA), member of the IET (UK), a Chartered Engineer (UK), and a registered Professional Engineer (Pakistan).

Iftaquaruddin Mohammed is a software engineer with a B.Sc. degree in electronics and M.Sc degree in computer science from SMU, India. He also holds the postgraduate diploma in computer science from Victoria University (Australia) and is a technical staff member in the Department of Electrical and Computer Engineering at Sultan Qaboos University (Oman). He has been involved in conducting digital logic design laboratory for undergraduate students for over five years.