A NEW SINGLE STUCK FAULT DETECTION ALGORITHM FOR DIGITAL CIRCUITS

Ahmed K. Jameil

Diyala University, Collage of Engineering,
Department of Computer & Software Engineering,
Diyala, Iraq.
ECeng89@gmail.com

Abstract— Single stuck line is a deficiency model utilized as a part of computerized circuits. It is used for post assembling testing. The model expects one line or hub in the computerized circuit is stuck at logical high or logical low. When a line is stuck it is called a fault. Fault detection in digital circuits is very important to get accurate results. When redundancy is introduced into digital circuits to achieve fault tolerance, the necessary overhead must be weighed against the improvement in reliability. In order to achieve a new design to simulate fault for a digital combinational circuit that is made up of ten gates and three inputs with one output. Results for single stuck at all faults are presented with details and the code written by c language.

Keywords—Combinational Circuit; Stuck-at Faults; Serial Fault Simulation; Deductive Fault Simulation; and Test Pattern Generation.

INTRODUCTION

The fault testing and diagnosis of digital circuits become an important and indispensable part of the manufacturing process at the point when advanced circuits are utilized as a part of security discriminating applications, their flaw conduct and its results must be assessed. A run of the mill inquiry is: will this issue lead to an unsafe circumstance? The idea of "hazardous" makes an application-specific gathering of inadequacies. The issue of test example era is more noteworthy. Since the sweep based configuration for testability systems are progressively utilized as a part of VLSI circuits, test example pattern for combinational circuits is getting significantly more essential. The test example pattern issue can be seen as a limited space look issue of discovering suitable logical assignments to the essential inputs such that the given flaw is recognized [1].

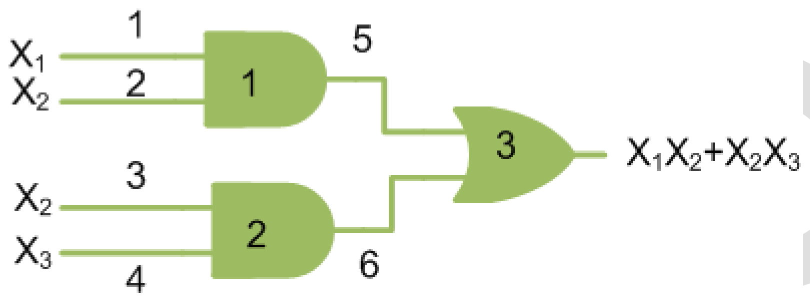
There are problems in the ASCs are the races. A race can exist in a circuit if two or more feedback signals are changing simultaneously. The race is essential for the order of changes can affect the final circuit state. Therefore another problem at the ASCs design is to avoid critical races [2]. A defect in an automated system is the unintended difference between the implemented hardware and its intended design. A representation of a weakness at the abstracted function level is called a fault. The term fault refers to electrical, Boolean, or functional malfunctions [3].

The faults of ASCs can be process faults, transient faults, or delay faults. Process faults originate from fabrication. Transient faults are the ones that might occur at some time, but are not stable in the sense they might not occur at other times. If a fault causes a circuit to exceed its timing specifications, but does not affect its logical function, it is deemed to be a delay fault [4]. The most known and used process fault model is the stuck-at fault model. The stuck-at fault is modeled by assigning a fixed logic zero or one to a signal line in the circuit [5]. In this paper, only stuck-at faults are considered.

Finally, in this paper the serial fault simulation method is utilized. The simulation is done by Computer Programming “C Language”. It is a powerful language which is being used for wide varieties of applications and Embedded Systems.

TYPES OF CIRCUITS

Types of digital circuits under study are simple two stage combinational circuits. The practical digital circuits, which are composed of AND, OR, NOT, NAND and NOR gates are alone chosen for measuring the performance of the methods in survey. Furthermore, methods derived for obtaining tests for this class of circuits are general enough to be applied to circuits consisting of gates other than these five types, such as the XOR gate, with minor modifications. It is assumed that the response delays of all the gate elements are the same. Figure (1) shows an example of a simple two stage combinational circuits. X1, X2 and X3 are the circuit inputs and X1X2+X2X3 is the fault free response of the circuit. Interconnections between the gates are numbered.



Figure(1)sample circuit

PROBLEM STATEMENT

The issue of deterministically creating a test example for a given stuck at deficiency (s_a_{0, 1 }) is to discover a mix of assignments of logical values (0 or 1) to the essential inputs which recreate the given shortcoming (line support) and screen the given shortcoming no less than one of the essential yields (deficiency proliferation) [6].

The undertaking of a deterministic test example generator is to create a test design or recognize the shortcoming as untestable (repetitive) for each deficiency in the defect rundown and structure a test set for testable flaws. Since the pursuit of discovering a test example devours inadmissible measures of calculation time, a prematurely ending measure is utilized as a part of all test generators. On the off chance that the quest for a given issue couldn't be completed, the defect is recognized as prematurely ended [7].

SERIAL SIMULATION

This is the simplest algorithm for simulating faults. The circuit is first simulated in the true-value mode for all vectors and primary output values are stored in a file. Next, faulty circuits are simulated one by one. This is achieved by modifying the circuit description for a target fault and then using the true-value simulator. As the simulation proceeds, the output values of the faulty circuit are dynamically opposed to the saved true responses. The simulation of a faulty circuit is switched off as soon as the comparison indicates detection of the target fault. All faults are simulated serially in this way.

ALGORITHM of SERIAL SIMULATION

Logical faults affect the state of logic signals. Normally, the state may be modeled as {0, 1, X (unknown), Z (high impedance) }, and a fault can transform the correct value to any other value. The serial fault simulation is a result of stuck _at_ 1 and stuck_ at _0 which is briefly outlined in the following and illustrated by the flowchart of serial fault simulation. Total number of faults is generated is:

Total number of faults(x) = #PI + #gates + # (fanout branches) (1)

Where

PI is a primary inputs of the circuit.

#Gates is the number of gates that it is a digital circuit used, and

fanout branch that are necessary components for a circuit.

In flowchart below x equal 25 that it is absolute fault sites. The count is to determine fault detected, in initialized set count to zero, also flag, when flag is one in each state (stuck_at_0 and stuck_at_1) which it is mean detected fault then account it. As showed in Figure (2).

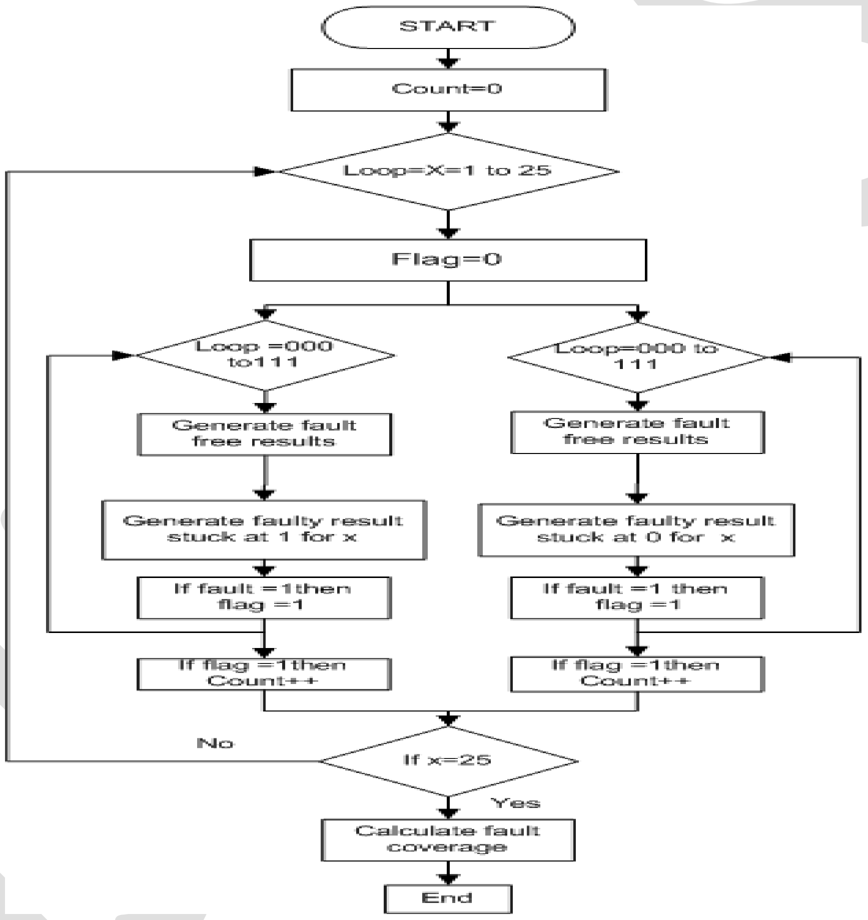


Figure (2): Algorithm flowchart for analysis of the serial fault simulation method.

SIMULATION RESULTS

Fault simulation comprises of simulating a circuit's behavior in the presence of faults. Comparing the faulty response of the circuit to that of the fault-free response using the same test set T, can determine the faults detected by T. Fault simulation has many applications, such as test set evaluation, fault-oriented test generation, fault dictionaries construction, and examination of circuit operation in the vicinity of deficiencies. There are various calculations for deficiency recreation: serial, parallel, deductive, simultaneous, parallel-example single-flaw engendering and basic way is following. The greater part of these calculations focused on the single stuck at deficiency model for physical flaws.

Figure (3) shows the Logic circuit under test, the combinational circuit which is composed of at ten gates and three inputs and only one output to be used to demonstrate fault simulator.

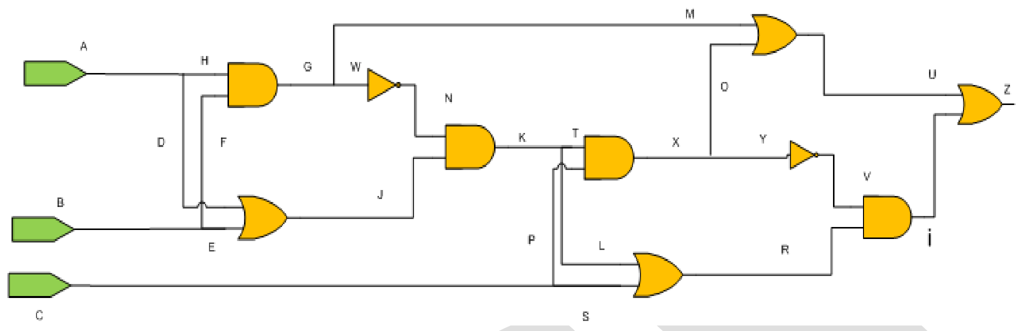
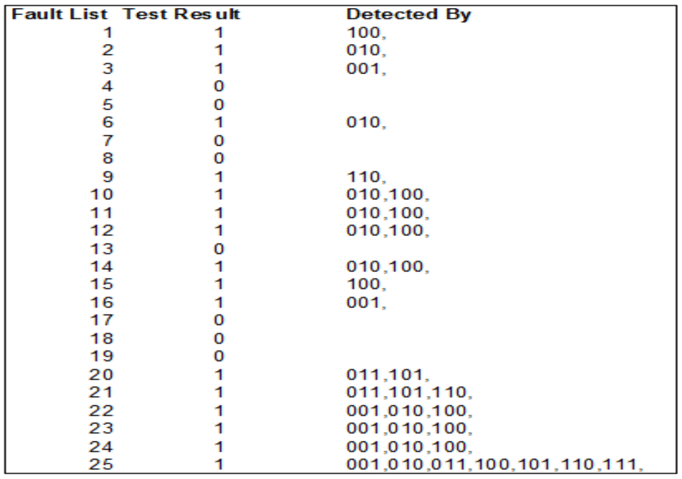


Figure (3): Logic circuit under test

For Figure (2) shows that the circuit through three inputs lines a total eight inputs was given from 000 to 111. Output obtained by the simulation program for both cases of stuck _at_0 and stuck _at_ 1 are given in Table (1) and Table (2).

Table 1: Stuck at fault 0 (stuck_at_0)



Fault List	Test Result	Detected By
1	1	100,
2	1	010,
3	1	001,
4	0	
5	0	
6	1	010,
7	0	
8	0	
9	1	110,
10	1	010,100,
11	1	010,100,
12	1	010,100,
13	0	
14	1	010,100,
15	1	100,
16	1	001,
17	0	
18	0	
19	0	
20	1	011,101,
21	1	011,101,110,
22	1	001,010,100,
23	1	001,010,100,
24	1	001,010,100,
25	1	001,010,011,100,101,110,111,

Figure (4) shows that the Relationship between fault and its occurrence for stuck _at_0.

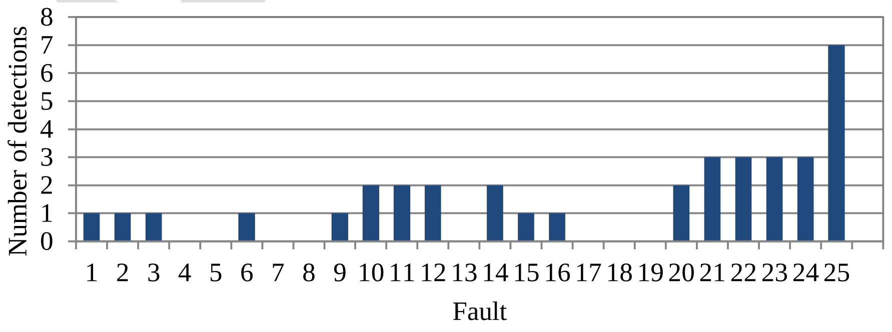
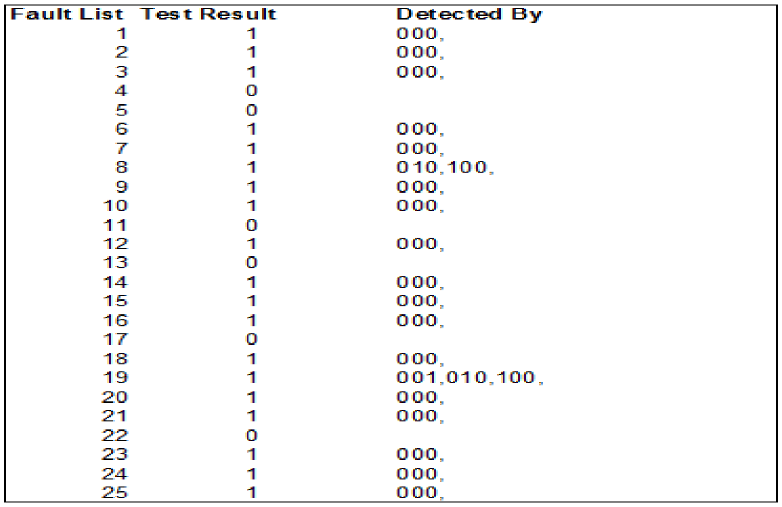


Figure (4): Relationship between fault and its occurrence

Table2: Stuck at fault 1(stuck_at_1)



Fault List	Test Result	Detected By
1	1	000,
2	1	000,
3	1	000,
4	0	
5	0	
6	1	000,
7	1	000,
8	1	010,100,
9	1	000,
10	1	000,
11	0	
12	1	000,
13	0	
14	1	000,
15	1	000,
16	1	000,
17	0	
18	1	000,
19	1	001,010,100,
20	1	000,
21	1	000,
22	0	
23	1	000,
24	1	000,
25	1	000,

Figure (5) shows that the Relationship between fault and its occurrence for stuck _at_1.

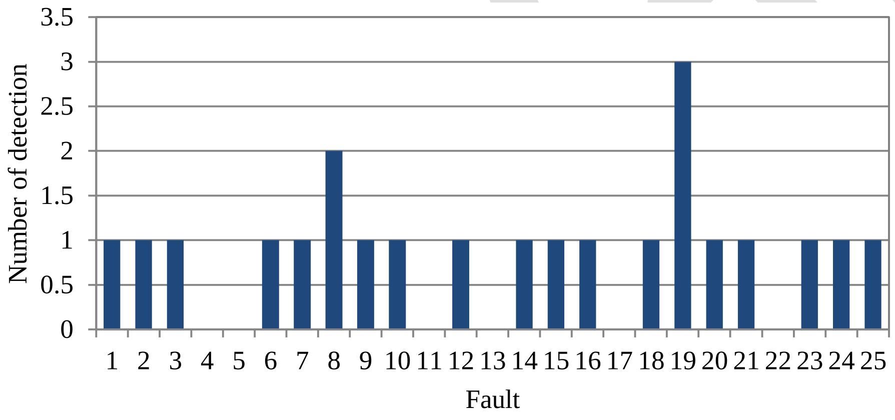


Figure (5): Relationship between fault and its occurrence

From results shows that a fault is detected during any of the give 8 inputs (000 to 111). Results for fault stuck _at _0 and stuck _at_1. Various faults were detected by various injected input values ranging from 000 to 111. The detected method is done by generating a truth table by hand calculation as well as by stimulating program. The results were compared with final output pin that is pin 25. The truth table for accurate circuit (Non-faulty circuit) is given Table (3).

Table (3) TEST PATTERNS

TEST PATTERNS	RESULT
000	0
001	1
010	1
011	1

100	**1**
101	**1**
110	**1**
111	**1**

The simulation program starts with the value 1 and finishes at 25 because the chosen digital circuit has 25 lines. So each number represents that the corresponding line is faulty. For each line, the both faulty circuits stuck at fault will compare the output value for inputs 000 to 111. It turns the flag HIGH if the output of the faulty circuit is different from Non-Faulty circuit. Hence this sequence will run upto 25 cycles. After each iterating, the number of faults is counted by using a variable "count". In the end, fault coverage is calculated by dividing number of fault with 50 (because total possible faults for a given circuit are 50; 25 faults for each stuck_at_0 and stuck_at_1). From the simulated results shown above, total number of detected fault is 48. So Fault Coverage = 48/50 * 100%= 96 %.

As a contrast between the program output and hand calculation the results as showed in Table (4).

Table (4).Truth table for test pattern and fault coverage

Number of test	Test pattern	Fault coverage(fc)
1	000	20%
2	001	14%
3	010	12%
4	011	9%
5	100	20%
6	101	7%
7	110	7%
8	111	7%
Fault coverage		96%

CONCLUSION

The deductive fault simulator was proposed for the digital circuit with ten gates and three inputs with one output, its implementation and tested. In this paper the comparison results generated by simulation program and calculation results, it is proved that this program is 100% capability for fault detection and gives accurate results. The simulator can be extended to modify for any other circuits, having different number of inputs as well as outputs.

REFERENCES:

[1] P. Goel. “An Implicit Enumeration to Generate Tests for Combinational Logic Circuits”.IEEE Transactions on Computers, Vol. 30, pp.215-222, 1981.

[2] Eichelberger, E. B.”Hazard Detection in Combinational and Sequential Switching Circuits”. IBM Journal of Research and Development, Vol. 9, 1965, No. 2, pp. 90–99.

[3] Bushnell, M. L. Agrawal, V. D"Essentials of electronic testing for digital, memory, and mixed-signal VLSI circuits". IEEE Circuits and Devices. 2001. Vol. 17 Issue: 4 Pages: 39-40.

[4] LaFrieda, Ch. Manohar, R. “Fault Detection and Isolation Techniques for Quasi Delay-Insensitive Circuits”. Proceedings of the 2004 International Conference on Dependable Systems and Networks, DSN ’04, 2004, pp. 41–50.

[5] Novak, O.Gramatov´ a, E.Ubar, R.´ et al. “Handbook of Testing Electronic Systems”. Czech Technical University Publishing House, 2005.

[6]. M. Schulz, E. Trischler, and T. Sarfert, SOCRATES. “A Highly Efficient Automatic Test Pattern Generation System”, IEEE Transactions on Computer-Aided Design, Vol. 7, pp. 1 26- 1 36, 1 988.

[7]. W. Kunz, and D. Stoffel, Reasoning in Boolean Networks.”Logic Synthesis and Verification Using Testing Techniques”, Kluwer Academic Publishers, London, 1 997