International Journal of Technology v(i) pp-pp (2024)
Received Month Year / Revised Month Year / Accepted Month Year

International Journal of Technology

http://ijtech.eng.ui.ac.id

Optimal Test Pattern Analysis and Fault Detection using Correlation Table for the 74283 Fast Adder Circuit

Geraldio Ramadhan Safitri[1*], Reza Darmakusuma[1]

[1]*School of Electrical Engineering and Informatics, Institut Teknologi Bandung, Bandung, 40132, Indonesia*

Abstract. The complexity of fast adder circuitry makes it prone to defects and faults. Effective testing is necessary to ensure the circuit functions correclty. One of the traditional yet effective method is generating optimal test patterns. In this paper, we aimed to understand the optimal test patterns for 74283 fast adder circuit by reverse engineering the circuit, analyzing the predefined optimal test pattens, and introducing various fault models to the circuit. We also proposed a correlation table to assist in the fault location detection process. We found out that combination of optimal test patterns and the proposed correlation table could precisely locate the faults in the GP module for singe stuck-at fault and detect possible locations for OR-bridging and AND-bridging faults, but still struggles with faults that occuring in the CLA module.

Keywords: Carry-Lookahead Adder; Digital System Testing; Fast Adder; Optimal Test Pattern

1. Introduction

Binary addition process is one of the most fundamental and important parts of almost any digital system, both in general-purpose and specific application processing. Addition circuitry usually exist in the Arithmetic Logic Unit (ALU) of a system (Han and Carlson, 1987). There are various types of adders, including Ripple Carry Adder (RCA), Carry Select Adder (CSL), Carry Skip Adder (CSK), and Carry Look-Ahead Adder (CLA) (Nagendra et al., 1996). The most basic form of adder in digital system is RCA, which has relatively simpler circuit and straightforward operation (Wanhammar, 1999). The downside of this adder circuitry is that this adder suffers from propagation delay in carry calculation. This is because this adder generates each sum sequentially. Therefore, the CLA type of adder is developed to reduce the propagation delay of the carry addition process. The CLA adder, also usually called a Fast adder, is widely used in digital systems due to its superior performance over the traditional adders like RCA. One of the commonly used CLA Integrated Circuit (IC) is the 74283 IC.

74283 is a multi-bit fast adder capable of performing 4-bit binary addition operations with a fast carry circuit. It performs the addition of two 4-bit binary numbers along with a carry input (C0). The output of the circuit includes the sum (S) provided for each bit and an additional bit (C4) which serves as the carry output of the addition result. Fast adder 74283 consists of three modules, including the Generate-Propagate (GP) module, Carry Look-Ahead module, and Sun module. The Carry Look-Ahead module in this adder enables the circuit to generate four-bit outputs and a carry bit typically within ten nanoseconds. That could be achieved because the circuit performs addition for both sum outputs and carry in

*Corresponding author's email: 23223042@mahasiswa.itb.ac.id
doi: 10.14716/ijtech.v0i0.0000

parallel, rather than sequentially as in the traditional RCA. Although this adder is faster, it has relatively more complex circuitry which could lead to larger number of defects and various faults. Additionally, the complex circuitry also makes the fault detection harder. Therefore, proper testing is needed to ensure the circuit used in the system is free from defects and faults.

Test pattern is a fundamental component of the testing process in digital systems. This term refers to specific combination of inputs applied to the circuit to verify the circuit functionality. A proper test pattern can also detect defects and faults in the circuit. Usually, the process of creating test pattern starts with creating the fault models, and then the test pattern is generated and all the possible combinations is applied to the circuit. However, in large-scale circuit testing processes, applying all possible combinations is inefficient and time-consuming (Hirose et al., 1988). Thus, to improve the efficiency, the optimal test pattern needs to be determined. An optimal test pattern is a set of test patterns consisting of a smaller number of combinations that cover the same amount of test coverage as the total test patterns.

In this paper we attempt to reverse engineer the process by modelling the fault models to understand the predefined optimal test pattern generated for 74283 as shown in Table 1 and find the reasons behind the combinations selected as the optimal pattern. We will start by explaining kinds of faults that the current predefined test pattern most likely covers based on our knowledge. Then, we introduce the several type of fault models to the circuit to verify test coverage of the pattern. We will analyze the results to determine if the pattern can clearly detect the faults to a specific degree or just generally locate the fault.

Table 1 Predefined 74283 IC optimal test pattern (Hansen et al., 1999)

No	Test Pattern		
	C	A $_{[3:0]}$	B $_{[3:0]}$
1	0	1111	0000
2	1	0000	1111
3	0	1111	0001
4	0	1110	0010
5	0	1100	0100
6	0	1000	1000
7	1	1110	0000
8	1	1101	0000
9	1	1011	0001
10	0	0111	0011

This paper is organized as follows. In section 2, we explain the process used to derive the circuitry of the adder, and finding the possible fault locations that can be detected using the predefined test pattern. Furthermore, in this section we also present the types of faults modelled in the paper verify the test pattern. In section 3, we present and discuss the results of the test pattern verification using the previously created fault model. Finally, we conclude the paper in section 4.

2. Methods

This paper used three main methods to verify and understand the behavior of the predefined test patter. The proposed method consists of reverse engineering of the 74283 circuitry, fault modeling and introduction to the circuit, and the development of a VHSIC Hardware Description Language (VHDL) program. Additionally, the developed code is simulated to verify the modeled fault logic and then implemented on a Field Programmable Gate Array (FPGA) for a more comprehensive testing and verification.

2.1. Reverse Engineering Process

In reverse engineering process, we start by recreating the gate-level logic diagram of the 74283 circuit. Figure 1 shows both the pin configuration and logic symbol module representation of the 74283 circuit, which will be used as a reference. The logic diagram is divided into the three corresponding modules based on Figure 1b. The purpose of this process is to make the analysis, modification, and fault modelling more flexible and easier to perform. We utilized a digital logic simulator for recreating the circuit, simulating the predefined test pattern, and analyzing the digital logic circuits. The Gate-Level diagram of the 74283 resulting from this process is depicted in Figure 2.

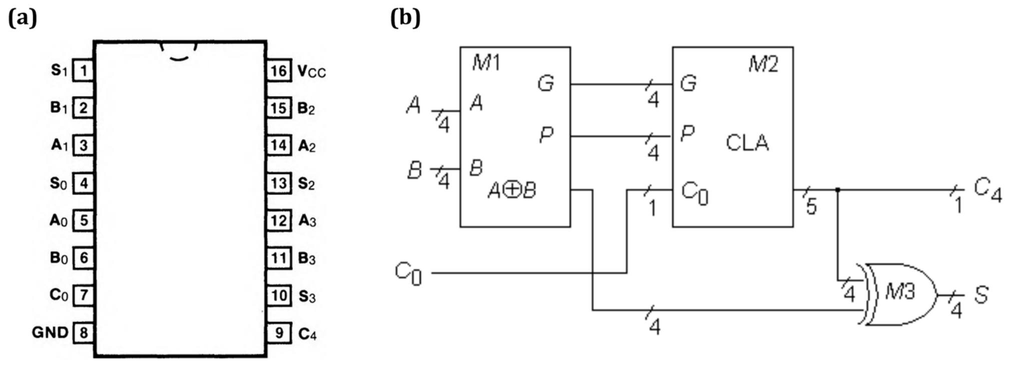

Figure 1 IC 74283 Fast Adder (a) pin configuration, (b) logic symbol module representation

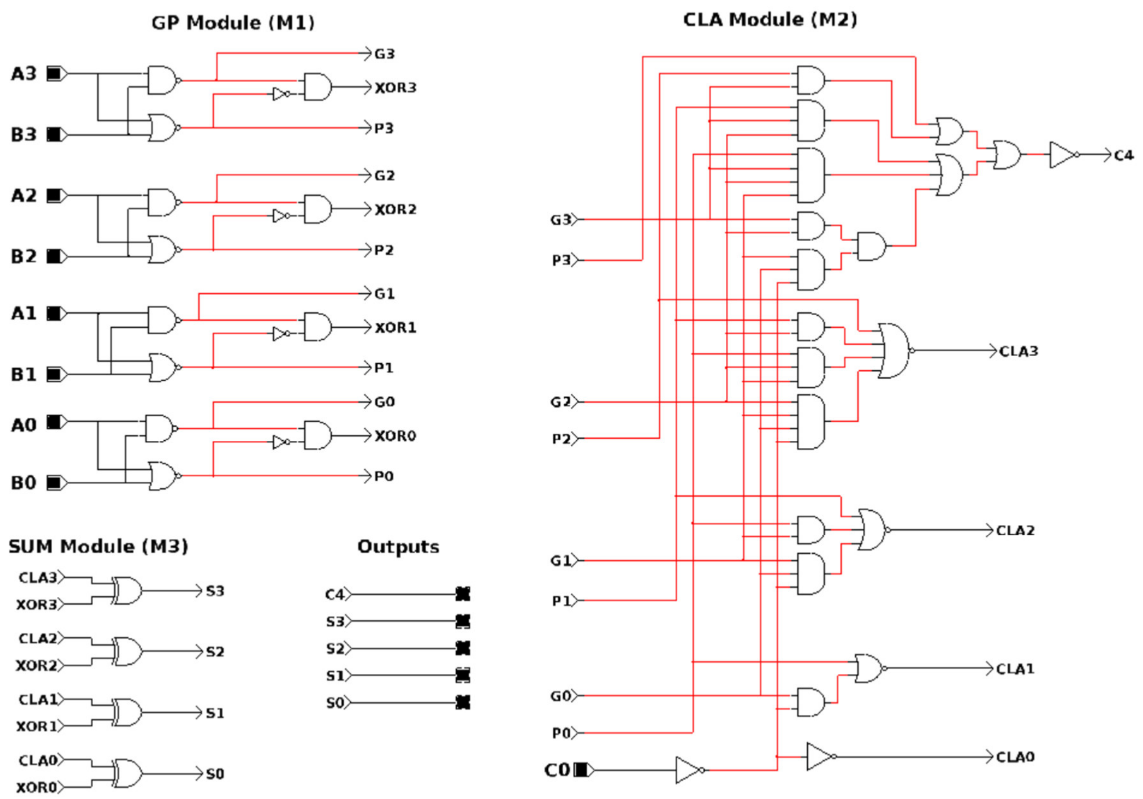

Figure 2 Gate-Level diagram derived from each module in IC 74283 Fast Adder

In the next step, we applied all the predefined test patterns to the previously recreated circuitry. This process is performed to analyze the faults that can potentially be detected using the test patterns. To identify the fault, we introduced various faults across different wires in the circuit. We mainly employed the stuck-at-type faults because they are easy to introduce to the circuit and can represent abnormalities similar to the other types of faults. We then determined the locations where the faults could be detected by the patterns. In this paper, we consider the GP module as the most critical module in 74283 circuit. As shown in Figure 2, any output wire from this module can be sensitized as a path that could directly affect the digital logic of the outputs. Additionally, the first NOT gate (nC_0) of the CLA module also considered a crucial part of the circuit since it is the main part that determine the addition result. Table 2 presents our insights into the faults that can be detected by the predefined test patterns.

Table 2 Fault detected by the predefined test pattern

No	Test Pattern			Fault Detected																			
	C_0	$A_{[3:0]}$	$B_{[3:0]}$	All A	All B	$P_0/0$	$P_0/1$	$G_0/0$	$G_0/1$	$P_1/0$	$P_1/1$	$G_1/0$	$G_1/1$	$P_2/0$	$P_2/1$	$G_2/0$	$G_2/1$	$P_3/0$	$P_3/1$	$G_3/0$	$G_3/1$	$nC_0/0$	$nC_0/1$
1	0	1111	0000	✓	✓			✓			✓	✓			✓	✓			✓	✓		✓	
2	1	0000	1111	✓	✓		✓				✓	✓			✓	✓			✓	✓			✓
3	0	1111	0001						✓														
4	0	1110	0010										✓										
5	0	1100	0100														✓						
6	0	1000	1000																		✓		
7	1	1110	0000			✓																	
8	1	1101	0000							✓													
9	1	1011	0001											✓									
10	0	0111	0011															✓					

Based on the presented table, we can infer that each predefined test pattern is chosen as optimal test pattern because it covers all the potential faults in the circuit especially in the GP module. During this process, we discovered that one test pattern could cover several fault detections. However, in Table 2, we only highlight the faults that are optimally detected by each pattern, which marked with a red checkmark symbol. The black checkmarks indicate that those types of faults can also be detected using other test patterns. Additionally, different outputs on the circuits indicate different potential locations where the fault occurs. In Table 3, we present several correlations between the output combinations and the possible detected faults.

Table 3 Correlation table between output combinations and potential fault locations

No	Test Pattern			Results $C_0A_{[3:0]}B_{[3:0]}$ (Potential Fault Detected)
	C_0	$A_{[3:0]}$	$B_{[3:0]}$	
1	0	1111	0000	10000 ($nC_0/0$ | $G_0/0$); 10001 ($G_1/0$); 10011 ($G_2/0$); 10111 ($G_3/0$); 01101 ($P_1/1$); 01011 ($P_2/1$); 00111 ($P_3/1$)
2	1	0000	1111	01111 ($nC_0/1$ | $P_0/1$); 10010 ($G_1/0$); 10100 ($G_2/0$); 11000 ($G_3/0$); 01110 ($P_1/1$); 01100 ($P_2/1$); 01000 ($P_3/1$)
3	0	1111	0001	01111 ($G_0/1$); 10010 ($G_1/0$); 10100 ($G_2/0$); 11000 ($G_3/0$); 01110 ($P_0/1$ | $P_1/1$); 01100 ($P_2/1$); 01000 ($P_3/1$)
4	0	1110	0010	10001 ($P_0/0$); 01110 ($G_1/1$)
5	0	1100	0100	10001 ($P_0/0$); 10010 ($P_1/0$); 01100 ($G_2/1$); 01000 ($P_3/1$)
6	0	1000	1000	10001 ($P_0/0$); 10010 ($P_1/0$); 00100 ($P_2/0$); 01000 ($G_3/1$)
7	1	1110	0000	01110 ($nC_0/1$); 10000 ($P_0/0$); 01101 ($P_1/1$); 01011 ($P_2/1$); 00111 ($P_3/1$)
8	1	1101	0000	10000 ($P_1/0$)
9	1	1011	0001	10000 ($P_2/0$)
10	0	0111	0011	10010 ($P_3/0$)

2.2. Fault Modelling

To further verify the test patterns, we introduced several different types of faults into the circuit. In this paper, we generally chose two types of faults consisting of stuck-at-type and bridging-type. We use two variants of each fault on the circuit, which resulting in total of four fault model including single stuck-at fault, multiple stuck-at fault, OR-bridging fault, and AND-bridging fault. A stuck-at fault occurs when a single or multiple lines have a fixed logical value (stuck-at) either at logic 0 or 1. Bridging faults occur when two different line short, resulting in the lines being connected together, behaving either as an OR-bridge (1-dominant) or an AND-bridge (0-dominant). In bridging faults, the bridge-gate produces a single output that branches into two new lines acting as new paths (Bushnell and Agrawal, 2002).

In this paper, we introduced the four faults separately across four different 74283 circuits. The implementation of the stuck-at fault variations is shown in Figure 3. We applied a single stuck-at fault on the GP module, specifically at G_1 line. As for the multiple stuck-at fault, we implemented it on the CLA module of another circuit, specifically at the output of three AND-gates in the CLA_3 path.

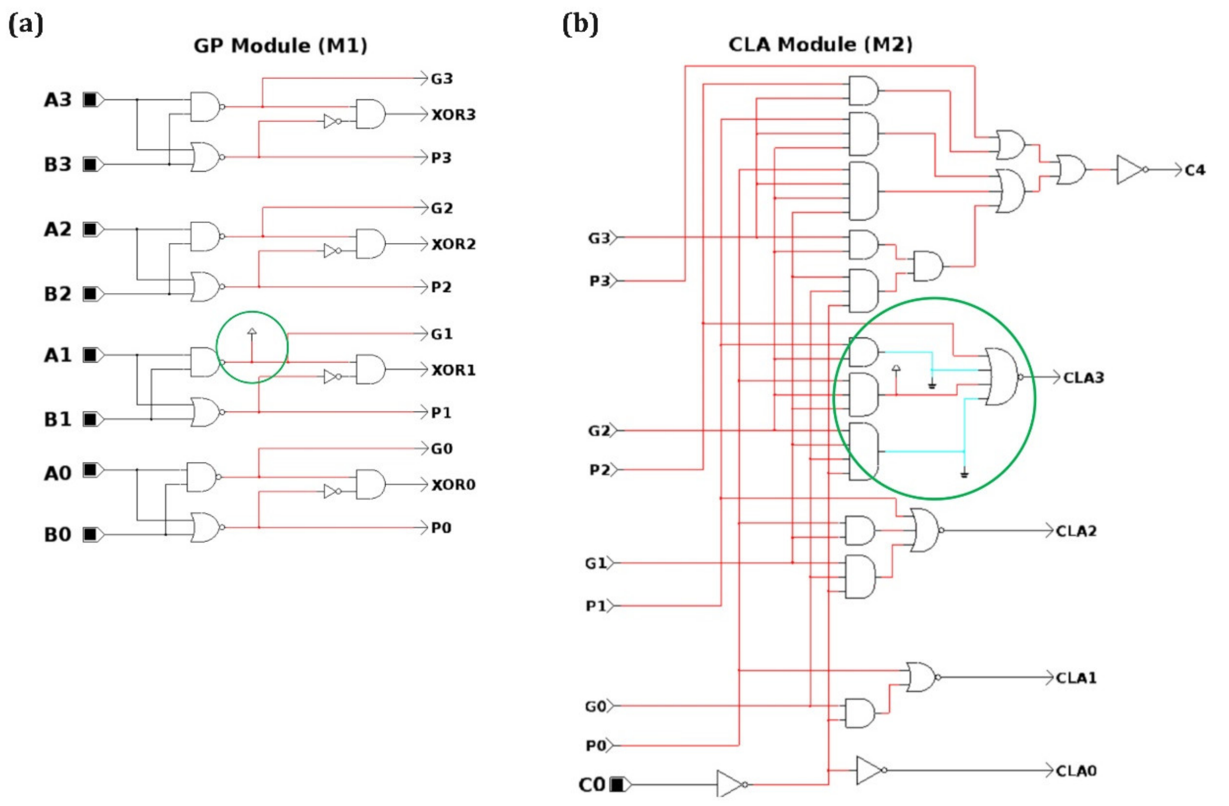

Figure 3 Stuck-at fault model variants modelling (a) single stuck-at fault on GP Module, (b) multiple stuck-at fault on CLA Module

For the bridging fault variations, we implemented the fault in the GP module and the SUM module of different circuits. Figure 4 shows the locations of the faults in the circuit. In the GP module, we implemented an OR-bridge (1-dominant) at the inputs of the AND-gate of XOR_2. In the SUM module, we introduced an AND-bridge (0-dominant) at the inputs of XOR gate that functions as the addition path for the S_2.

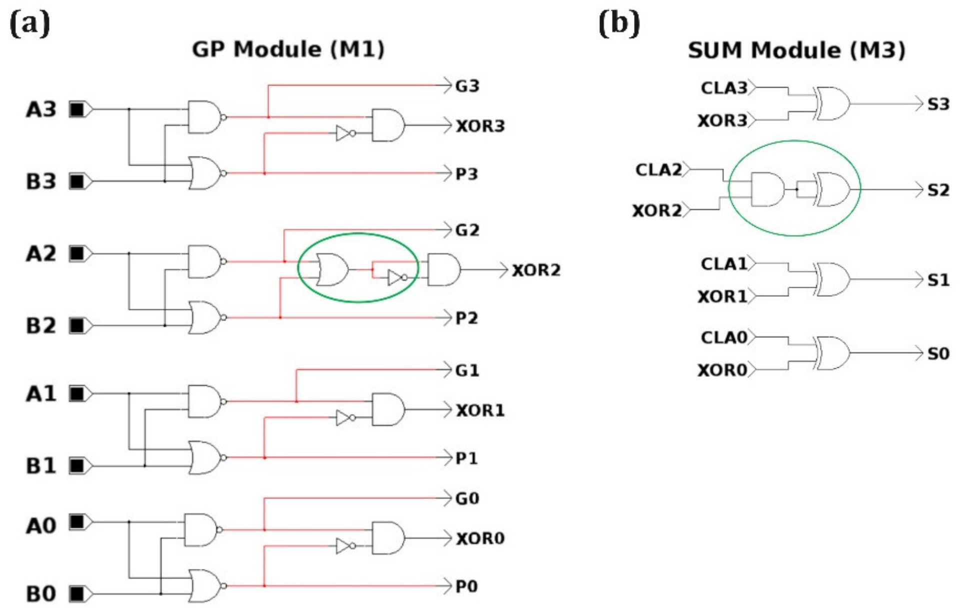

Figure 4 Bridging faults model (a) OR bridging at GP Module, (b) AND bridging at SUM Module

2.3. Simulation and Implementation

Based on the fault models implemented in previous subsection, we developed VHDL code for the corresponding adder with the introduced faults. The developed code is then simulated using Modelsim software to verify the code logical algorithm. To automate the testing process, we created testbenches for all the developed codes that automatically applied all the predefined test patterns during the simulations. Figures 5 through 8 show snapshots of the waveforms generated by the testing process through simulation. The verified fault model code was then implemented into an FPGA. This approach enabled us to conduct more comprehensive testing using real-world validation via the device. In this paper, we used Terasic DE-10 Standard Development Kit as the hardware system to simulate the adder logic. The implementation results and analysis will be discussed in Section 3.

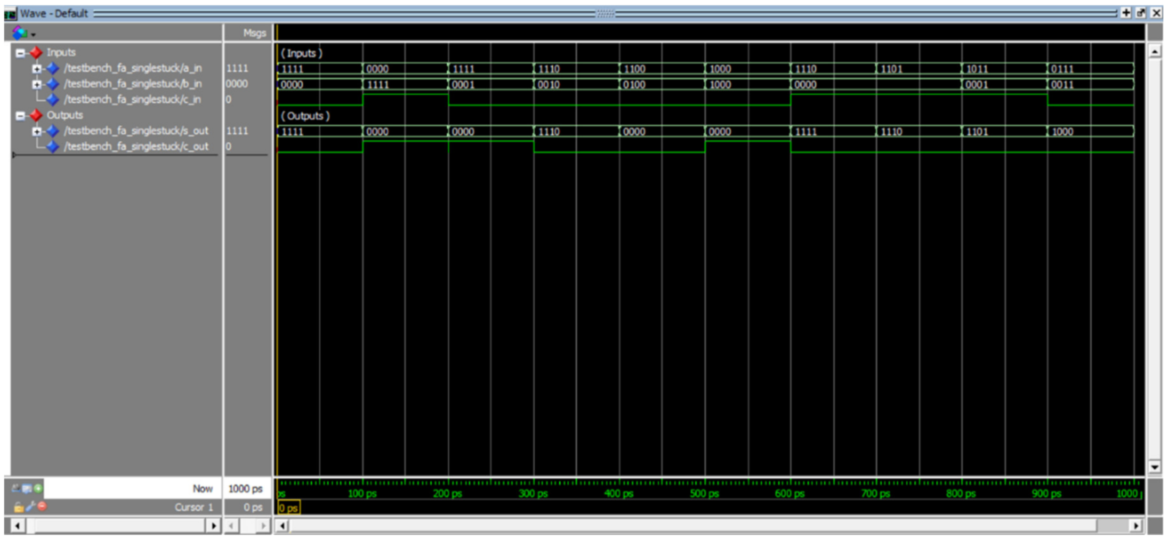
Wave - Default
Msgs
Inputs
/testbench_fa_singlestuck/a_in 1111
/testbench_fa_singlestuck/b_in 0000
/testbench_fa_singlestuck/c_in 0
Outputs
/testbench_fa_singlestuck/s_out 1111
/testbench_fa_singlestuck/c_out 0
(Inputs)
1111 0000 1111 1110 1100 1000 1110 1101 1011 0111
0000 1111 0001 0010 0100 1000 0000 0001 0011
(Outputs)
1111 0000 0000 1110 0000 0000 1111 1110 1101 1000
Now 1000 ps
Cursor 1 0 ps
100 ps 200 ps 300 ps 400 ps 500 ps 600 ps 700 ps 800 ps 900 ps

Figure 5 Simulation results of single stuck-at fault model in 74283 circuit

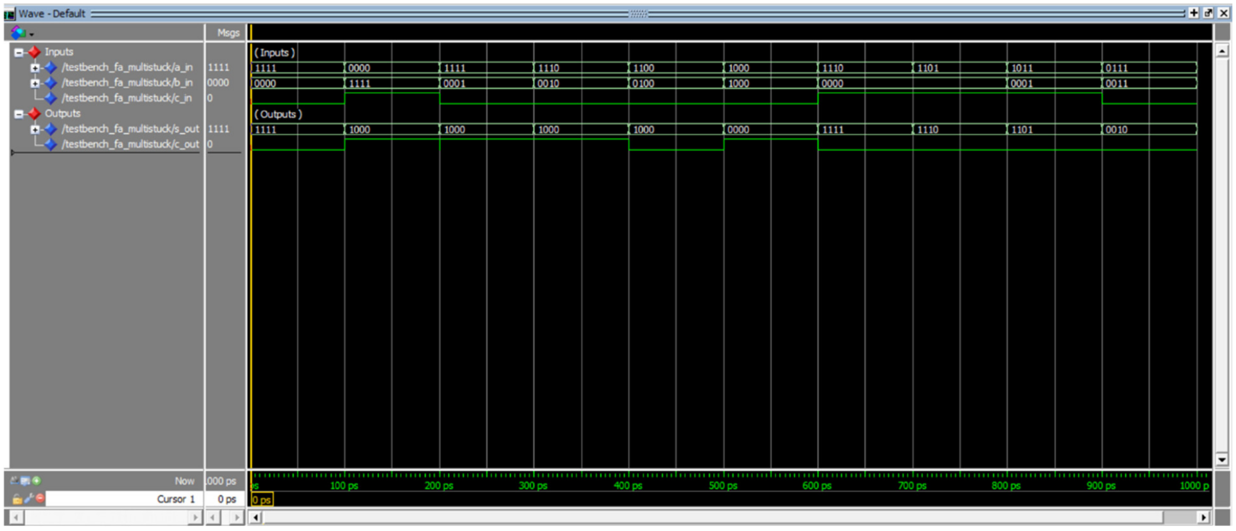
Wave - Default
Msgs
Inputs
/testbench_fa_multistuck/a_in 1111
/testbench_fa_multistuck/b_in 0000
/testbench_fa_multistuck/c_in 0
Outputs
/testbench_fa_multistuck/s_out 1111
/testbench_fa_multistuck/c_out 0
1111 0000 1111 1110 1100 1000 1110 1101 1011 0111
0000 1111 0001 0010 0100 1000 0000 0001 0011
1111 1000 1000 1000 1000 0000 1111 1110 1101 0010
Now 1000 ps
Cursor 1 0 ps
100 ps 200 ps 300 ps 400 ps 500 ps 600 ps 700 ps 800 ps 900 ps 1000 ps

Figure 6 Simulation results of multiple stuck-at fault model

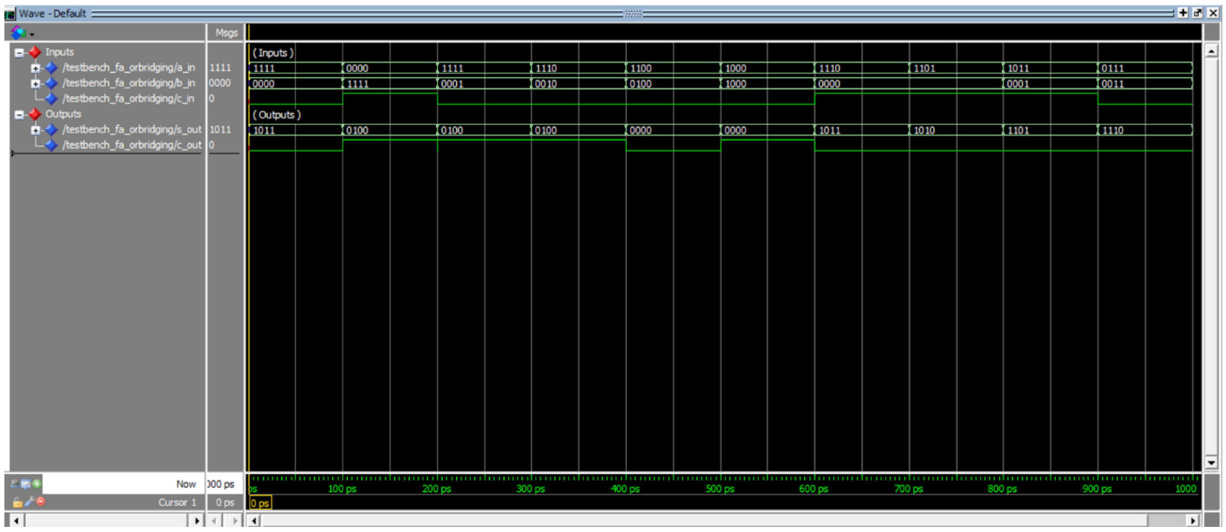
Wave - Default
Msgs
Inputs
/testbench_fa_orbridging/a_in 1111
/testbench_fa_orbridging/b_in 0000
/testbench_fa_orbridging/c_in 0
Outputs
/testbench_fa_orbridging/s_out 1011
/testbench_fa_orbridging/c_out 0
1111 0000 1111 1110 1100 1000 1110 1101 1011 0111
0000 1111 0001 0010 0100 1000 0000 0001 0011
1011 0100 0100 0100 0000 0000 1011 1010 1101 1110
Now 1000 ps
Cursor 1 0 ps
100 ps 200 ps 300 ps 400 ps 500 ps 600 ps 700 ps 800 ps 900 ps 1000

Figure 7 Simulation results of OR bridging fault model

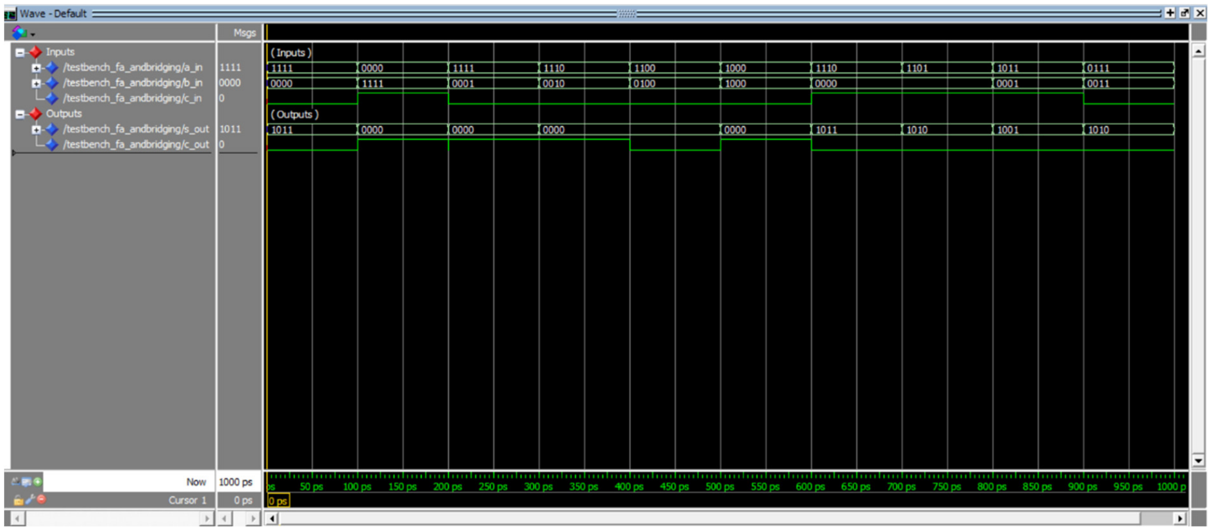
Wave - Default
Msgs
Inputs
/testbench_fa_andbridging/a_in 1111
/testbench_fa_andbridging/b_in 0000
/testbench_fa_andbridging/c_in 0
Outputs
/testbench_fa_andbridging/s_out 1011
/testbench_fa_andbridging/c_out 0
1111 0000 1111 1110 1100 1000 1110 1101 1011 0111
0000 1111 0001 0010 0100 1000 0000 0001 0011
1011 0000 0000 0000 0000 1011 1010 1001 1010
Now 1000 ps
Cursor 1 0 ps
50 ps 100 ps 150 ps 200 ps 250 ps 300 ps 350 ps 400 ps 450 ps 500 ps 550 ps 600 ps 650 ps 700 ps 750 ps 800 ps 850 ps 900 ps 950 ps 1000 ps

Figure 8 Simulation results of AND bridging fault model

3. Results and Discussion

3.1. *FPGA Implementations*

In this paper, an FPGA is used for the real-word validation process. Figure 9 presents the implementation of all four fault model codes developed in the previous section into the FPGA. We applied all the predefined optimal test patterns to these FPGAs, then observed and analyzed the output based on each fault scenario. As shown in Figure 9, with the same input of "011100010", each FPGA produces a different output due to the variations of faults introduced to the fast adder circuit.

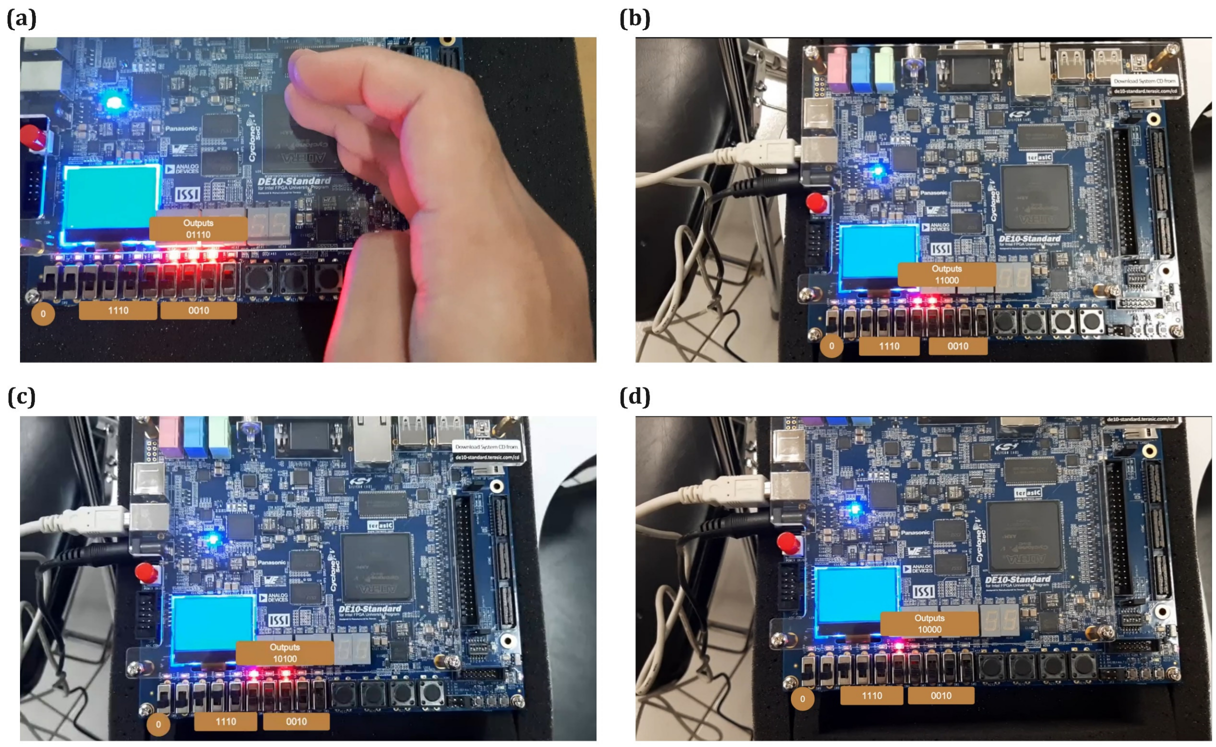

Figure 9 Implementation of fault models in FPGA at input 011100010 for (a) single stuck-at, (b) multiple stuck-at, (c) OR-bridging, and (d) AND-bridging fault.

3.2. *Predefined Test Pattern Coverage Testing*

In this subsection, we perform coverage testing of the predefined optimal test patterns. The objective of this testing is to verify that the test patterns are capable of detecting all the faults occurring in the fast adder simulated in the FPGA. We conducted this test by applying all the test patterns to the four FPGA implementations. The results of each test were recorder and analyzed. We determined the capability of the test patterns by comparing the expected addition process results with the actual result obtained from the circuit with faulty, as indicated by the LEDs on the FPGA. A good test pattern set should have at least one test pattern that detects the fault in the circuit by producing a different result than the expected result. If none of the ten test patterns can detect the fault introduced to the circuit, the set is considered ineffective, and should be changed to more effective test pattern. The results of the coverage testing are presented in Table 3.

Table 3 Coverage of the predefined test patterns on the four proposed fault models

No	Test Pattern			Results (C_4 $S_{[3:0]}$)								
	C_0	$A_{[3:0]}$	$B_{[3:0]}$	Expected	Singe Stuck-at	?	Multiple Stuck-at	?	OR-Bridging	?	AND-Bridging	?
1	0	1111	0000	01111	01111		01111		01011	✓	01011	✓
2	1	0000	1111	10000	10000		11000	✓	10100	✓	10000	
3	0	1111	0001	10000	10000		11000	✓	10100	✓	10000	
4	0	1110	0010	10000	01110	✓	11000	✓	10100	✓	10000	
5	0	1100	0100	10000	00000	✓	01000	✓	00000	✓	00000	✓
6	0	1000	1000	10000	10000		10000		10000		10000	
7	1	1110	0000	01111	01111		01111		01011	✓	01011	✓
8	1	1101	0000	01110	01110		01110		01010	✓	01010	✓
9	1	1011	0001	01101	01101		01101		01101		01001	✓
10	0	0111	0011	01010	01000	✓	00010	✓	01110	✓	01010	

In Table 3, faults detected or covered by the test patterns are marked with blue checkmarks. As shown in the table, the predefined test patterns successfully detect all the faults introduced to the fast adder circuit. The most effective test pattern for detecting the introduced fault models is number 5, as it can detect multiple type of faults, followed by test patterns number 4 and 10. We also discovered that the predefined test patterns could locate the fault, especially for the single stuck-at fault. From Table 3, we can see that the test pattern number 4 produces an output of "01110". Referring back to the Table 2 in Section 2, we can determine that the single stuck-at fault in this circuit is most likely located at the GP module, specifically at G1 with the line stuck at logic 1. This is correct if we refer to Figure 3a, where we modeled the single stuck-at fault to be at G_1. Similarly, the predefined test patterns can detect the OR-bridging fault location. While it cannot directly indicate the type of fault, it can show the locations where the fault is most likely occurring. Referring to Table 2, test patterns number 1, 2, 3, and 7 of Table 3 suggest a possible fault at the GP module on the G_2 and P_2 lines, which is correct if we refer bac to the OR-bridging fault model in Figure 4a. For the AND-bridging fault, although more challenging and not as direct, the outputs produced by test patterns number 1 and 7 indicate an issue in line P_2. This is somewhat accurate since this line shorted with another path. However, for multiple stuck-at fault, our currently proposed relation table cannot show the location or possible path since the fault occur in the CLA module, which is not deeply explored in this paper.

4. Conclusions

In this paper, we aim to evaluate the optimal test pattern proposed by Hansen et al. (1999). We conducted the study by reverse engineering the 74283 fast adder circuit, dividing it into modules, and analyzing the predefined test patterns accordingly. We also proposed a correlation table to determine the fault location in the GP module of the fast adder based on its output. We introduced four kinds of faults to circuit, including single stuck-at, multiple stuck-at, OR-bridging, and AND-bridging faults. Coverage tests were performed to verify the fault detection coverage of the optimal test pattern. We found out that the optimal test pattern is indeed effective, as it detected all the faults introduced to the circuit. Furthermore, we attempted to locate the faults in circuit by combining test pattern results with our proposed correlation table. The proposed relations table precisely identified the location of the single stuck-at fault in the GP module. It could also find the possible fault locations for OR-bridging and AND-bridging faults, which is useful for the initial circuit inspection. However, our table struggled to locate the fault location for multiple stuck-at faults. Therefore, out future work will focus on including more complex combinations and considering both the CLA and SUM modules for a better range of detection.

References

Bushnell, M.L., Agrawal, V.D., 2002. Essentials of Electronic Testing for Digital, Memory & Mixed-Signal VLSI Circuits, Frontiers in Electronic Testing.

Han, T., Carlson, D.A., 1987. FAST AREA-EFFICIENT VLSI ADDERS., in: Proceedings - Symposium on Computer Arithmetic. https://doi.org/10.1109/arith.1987.6158699

Hansen, M.C., Yalcin, H., Hayes, J.P., 1999. Unveiling the ISCAS-85 benchmarks: a case study in reverse engineering. IEEE Des. Test Comput. 16. https://doi.org/10.1109/54.785838

Hirose, F., Takayama, K., Kawato, N., 1988. Method to generate tests for combinational logic circuits using an ultrahigh-speed logic simulator, in: Digest of Papers - International Test Conference. https://doi.org/10.1109/test.1988.207786

Nagendra, C., Irwin, M.J., Owens, R.M., 1996. Area-time-power tradeoffs in parallel adders. IEEE Trans. Circuits Syst. II Analog Digit. Signal Process. 43. https://doi.org/10.1109/82.539001

Wanhammar, L., 1999. Processing Elements, in: DSP Integrated Circuits. https://doi.org/10.1016/b978-012734530-7/50011-8