Georgia Institute of Technology

[ECE6140] Digital System Tests

Project 2 - Fault Simulator

Lingjun Zhu

GTID # 903433936

Nov. 7, 2018

1. Data Structure

In this project, we are supposed to write a deductive fault simulator program that can determine the detected fault under a specified test vector. First, we need to implement the data structure for deductive fault lists and list event operations. Based on the program of Project 1 Logic Simulator, we have designed the data structure as shown in Figure 1.1.

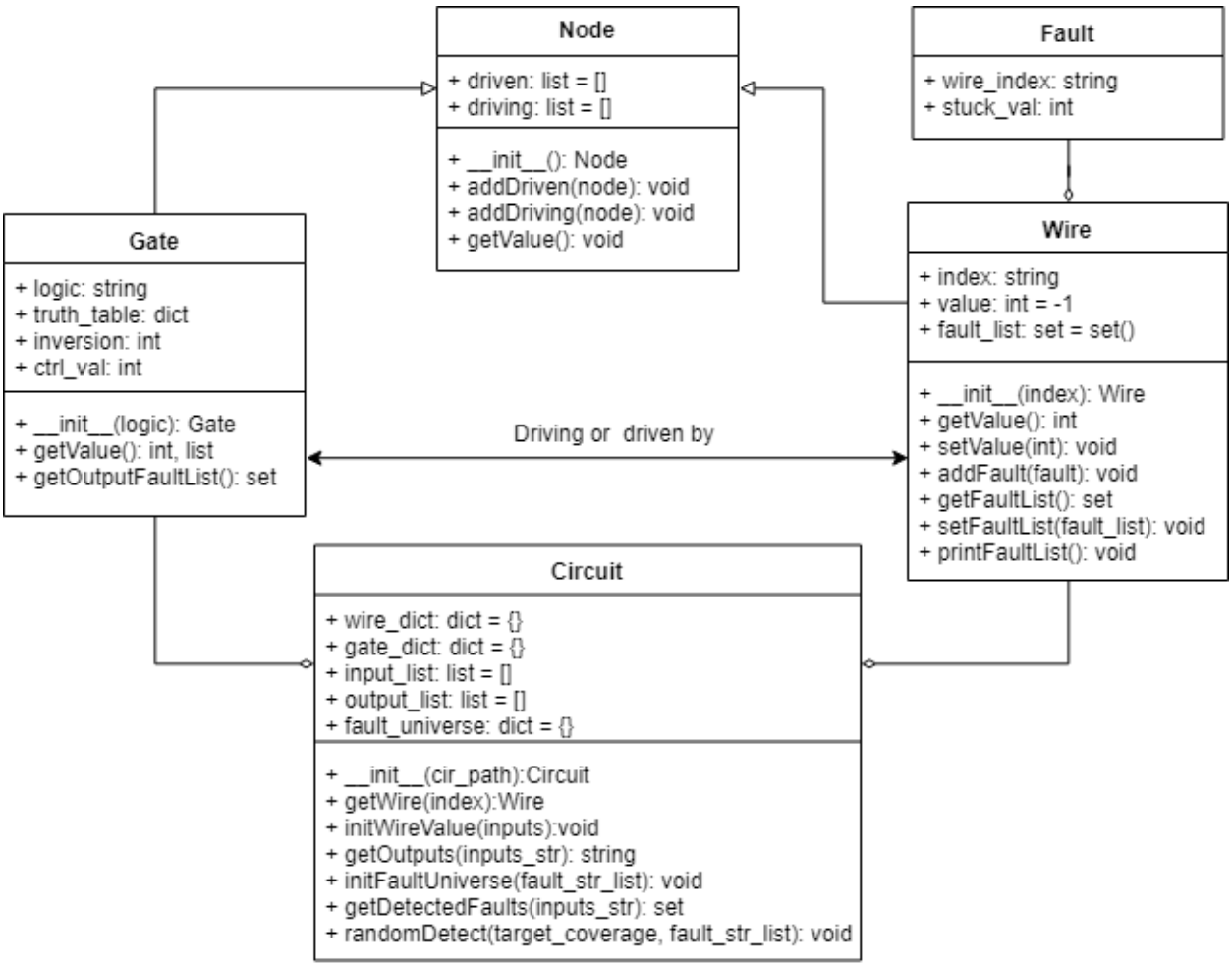

Figure 1.1 The class diagram of the fault simulator

Node remains the same as it in the logic simulator. It is the abstraction of gates and wires and used to describe the topology of the circuit. We design a new class **Fault** to describe the stuck-at faults. It has two properties: the wire index and the stuck-at value. Besides, we add two new properties to the **Gate** class: inversion parity and controlling value, which are essential for fault list calculation. And it provides a function to calculate the fault list for the output wire. In the **Wire** class, we add a set of **Fault** objects to represent the deductive fault list. The Python basic type, **set**, supports many set operation such as union and intersection and it is very useful for fault list calculation. In the **Circuit** class, we use a hash table to store all the fault in the circuit, called the fault universe. It provides a function to initialize the fault universe based on a given fault list or take all the stuck-at fault into consideration. And we implement the functions to get the detected fault based on a given test vector and find out the number of random vectors needed to achieve the target coverage.

2. Algorithm

First, we implement the function to calculate the fault list from the input wires to the output wires and get the detected fault, based on a First-In-Last-Out stack. The flow is shown as Figure 2.1.

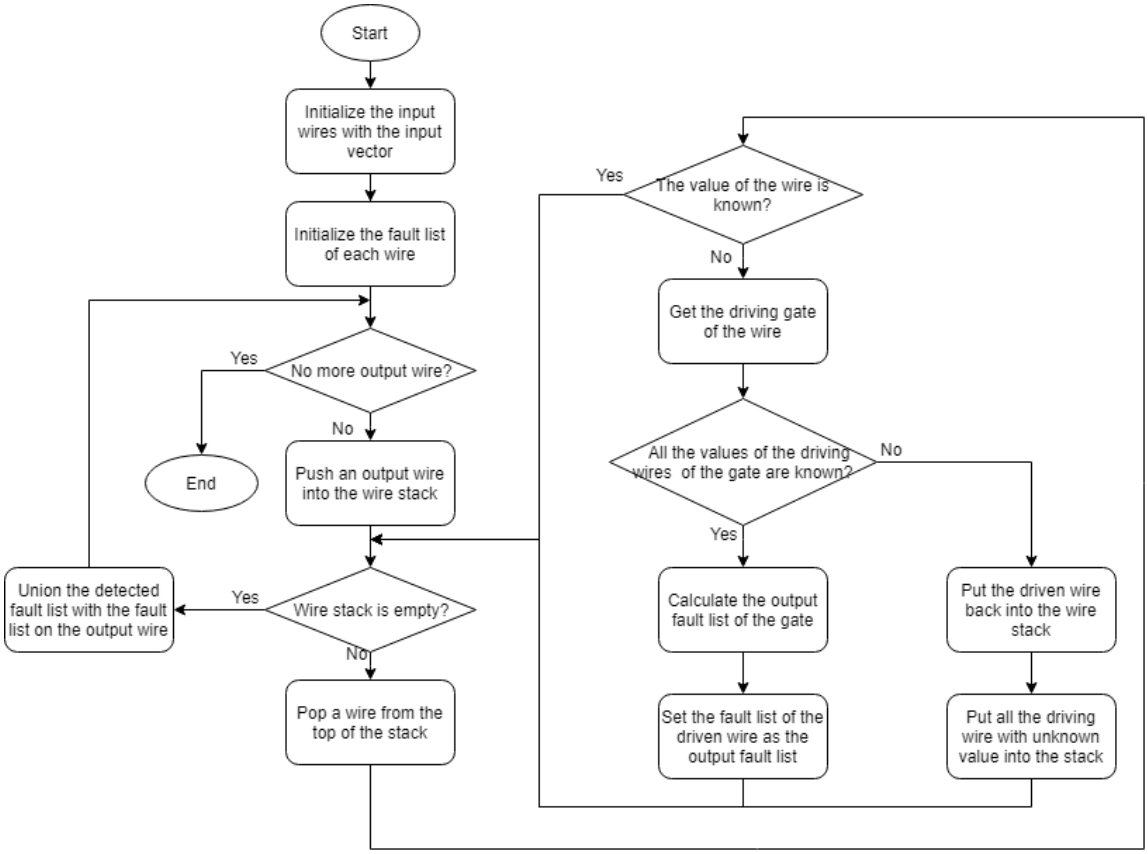

Figure 2.1 The flowchart of the subroutine to get the detected faults

This subroutine is very similar to the output logic value calculation flow in the logic simulator. We just replace the logic value calculation with fault list calculation. Based on this subroutine, we implement another function to find out the number of random test vectors needed to achieve the target coverage.

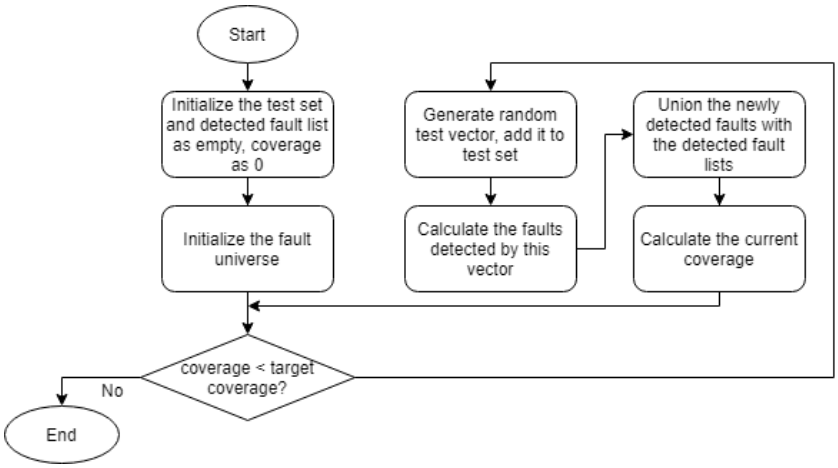

Figure 2.2 The flowchart of the circuit output calculation subroutine

3. Results

The detected faults for each circuit are shown as Table 3.1. For more details, please see the detected_faults.xlsx or the output files in the uploaded package.

Table 3.1 The inputs and outputs of the simulated circuits

Circuit	Input Vector	Detected Faults
s27	1110101	1 stuck at 0
		3 stuck at 0
		5 stuck at 0
		7 stuck at 0
		9 stuck at 1
		11 stuck at 1
		12 stuck at 0
		13 stuck at 0
	0101001	1 stuck at 1
		3 stuck at 1
		5 stuck at 0
		7 stuck at 1
		8 stuck at 1
		9 stuck at 1
		11 stuck at 0
		12 stuck at 1
		14 stuck at 0
		15 stuck at 1
		16 stuck at 1
		19 stuck at 1
		20 stuck at 1
s298f_2	10101010101010101	(82 faults detected in total)
		(see outputs/s298f_2_output_1.txt for details)
	11101110101110111	(53 faults detected in total)
		(see outputs/s298f_2_output_2.txt for details)
s344f_2	10101010101010101011111111	(101 faults detected in total)
		(see outputs/s344f_2_output_1.txt for details)
	11101011101010101010001100	(132 faults detected in total)
		(see outputs/s344f_2_output_2.txt for details)
s349f_2	10101010101010101011111111	(101 faults detected in total)
		(see outputs/s349f_2_output_1.txt for details)
	11111110101010101010001111	(137 faults detected in total)
		(see outputs/s349f_2_output_2.txt for details)

To achieve 75% and 90% coverage, the number of random test vectors needed for each circuit is shown as Table 3.2.

Table 3.2 The number of random test vectors needed to achieve the target coverage for each circuit

Circuit	Target Coverage (%)	Number of Random Test Vector
s27	75%	7
	90%	11
s298f_2	75%	8
	90%	21
s344f_2	75%	6
	90%	14
s349f_2	75%	8
	90%	12

Some of the random test vector number vs. coverage curves are shown as Figure 3.1.

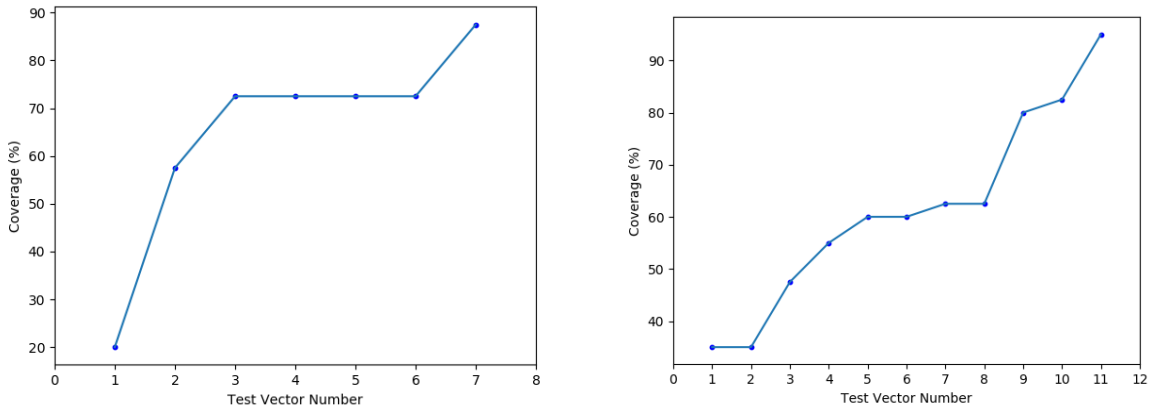

Figure 3.1 Random test vector number vs. coverage curve for circuit s27 (target coverage: left 75%, right 90%)

For more curves, please see figures/ directory in the uploaded package.

4. Conclusion

In this project, we have implemented a deductive fault simulator using Python. We have designed a data structure to store the deductive fault list and facilitate the list event operation. After that, we have implemented the functions to get the detected fault for a specified circuit with a given test vector, and calculate the number of random vectors needed to achieve the target coverages. Based on the functions, we have obtained the fault simulation results for the four given circuits, and draw the number of random vector vs. coverage curves.