Project 2

Deductive Fault Simulator

Georgia Institute of Technology

ECE 6140

Martin, Michael

902912673

November 9, 2017

**Data Structures**

The simulation defines three classes, Gate, Net, and Fault. The Gate class has three data members, a String to store the gate function (e.g. “INV”, “BUF”, etc.), an integer array to store the input nets, and an integer to store the output net. The Net class has two data members, an integer to store the value on the net and a List of Fault instances. The Fault class has two data members, an integer to store the net and an integer to store the stuck-at-value. Instances of these classes are stored in three arrays and two linked lists. The “nets” array stores an instance of the Net class at each index corresponding to the net number. The “inputs” array stores the index to the “nets” array of each net that is an input to the circuit. The “outputs” array stores the index to the “nets” array of each net that is an output to the circuit. The “gates” linked list stores instances of the Gate class corresponding to each gate in the circuit. Finally, the “faults” linked list stores instances of the Fault class corresponding to each potential fault in the circuit.

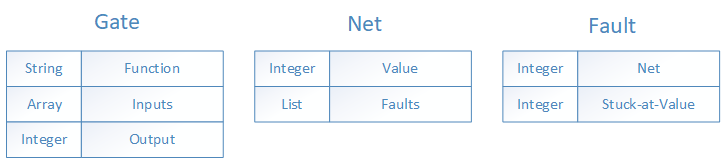
**Simulation Flow**

The simulation has two main loops. The first loop parses the input file line by line, and the second loop evaluates the logic gates in the circuit. In each iteration of the first loop, a new instance of the Gate class is created and added to the end of the “gates” linked list. Next, the “nets” array is created and each net is initialized with a value of -1 to indicate that the value is unknown. Then each input net is set to its value in the input vector. In each iteration of the second loop, the next gate in the list of gates is fetched. If all of the inputs to the gate are ready (i.e. none of the nets have a value of -1), then the gate function is evaluated and the output is stored at its index in the “nets” array. The list of faults that propagate to the output net are also computed. The gate is then removed from the list of gates. The second loop will terminate after all gates have been evaluated and the list of gates is empty.

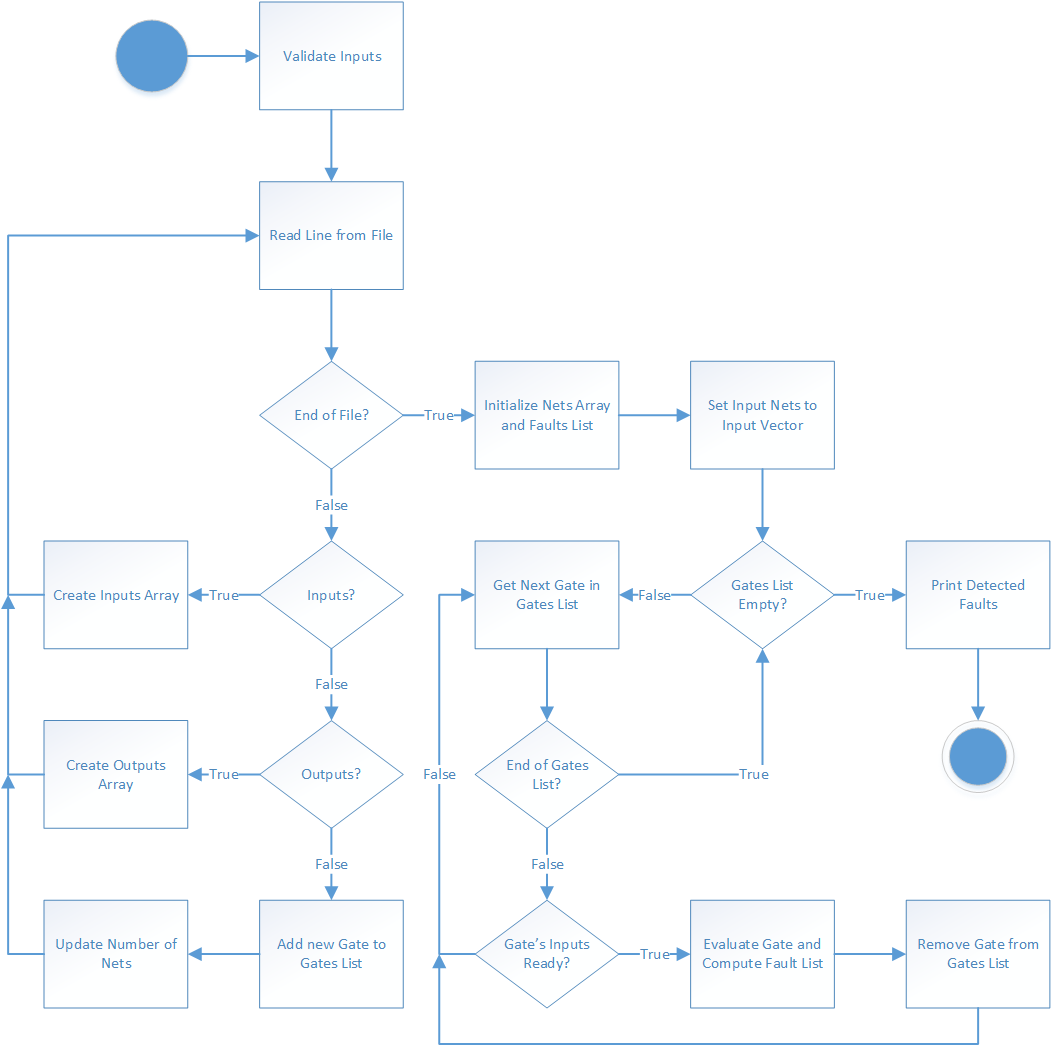
**Figures and Results**

**Table 1.** Number of Faults Detected in Four Test Circuits

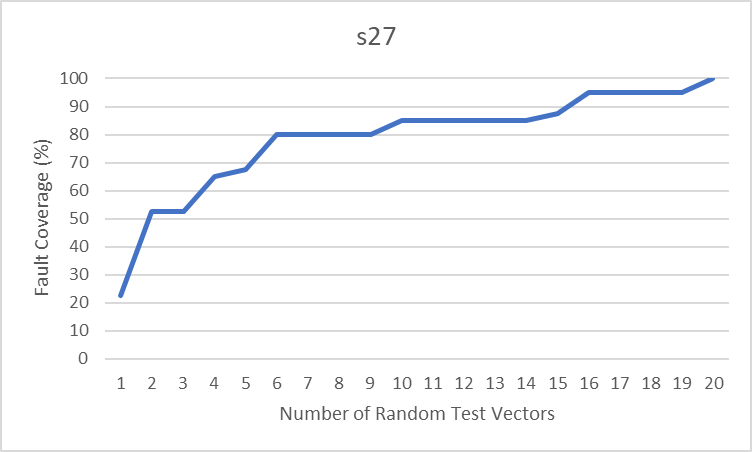
|  |  |  |
| --- | --- | --- |
| Circuit | Test Vector | Number of Faults Detected |
| s27 | 1110101 | 8 |
| 1010001 | 13 |
| s298f\_2 | 10101010101010101 | 82 |
| 11101110101110111 | 53 |
| s344f\_2 | 101010101010101011111111 | 101 |
| 111010111010101010001100 | 132 |
| s349f\_2 | 101010101010101011111111 | 101 |
| 111111101010101010001111 | 137 |



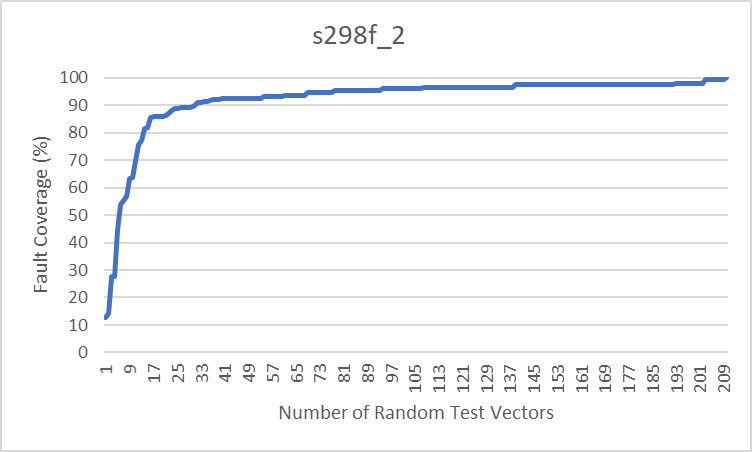
**Figure 1.** Class definitions used by the simulator.



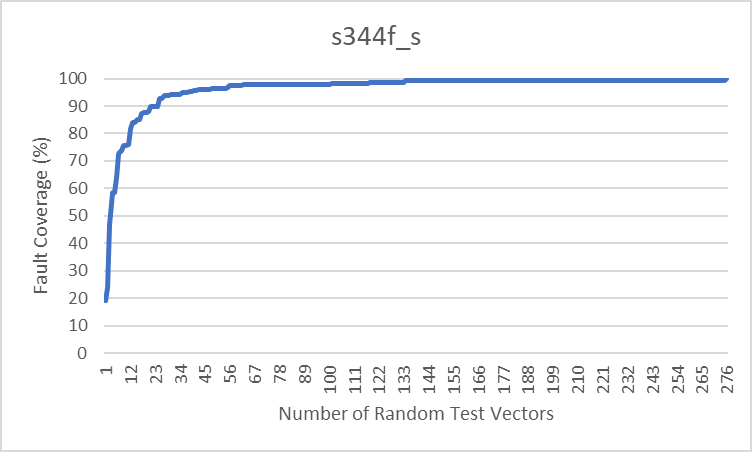
**Figure 2.** Algorithmic State Machine that describes the flow of the simulator.



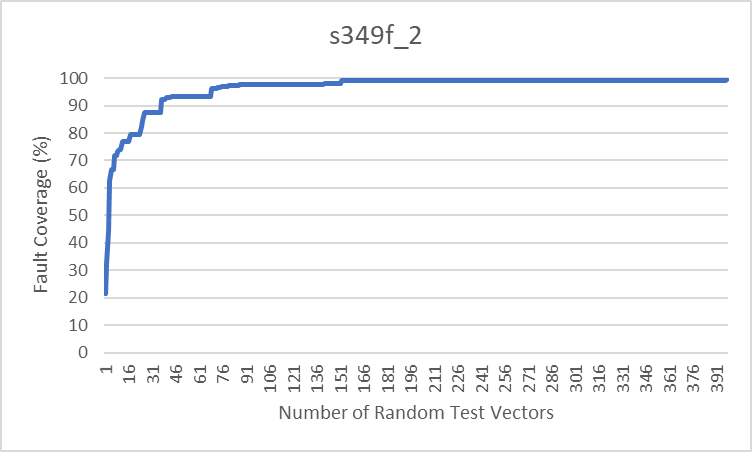
**Figure 3.** Fault coverage versus number of random test vectors applied to the s27 circuit. 6 test vectors are needed for at least 75% coverage, and 16 test vectors are needed for at least 90% coverage.



**Figure 4.** Fault coverage versus number of random test vectors applied to the s298f\_2 circuit. 12 test vectors are needed for at least 75% coverage, and 31 test vectors are needed for at least 90% coverage.



**Figure 5.** Fault coverage versus number of random test vectors applied to the s344f\_2 circuit. 9 test vectors are needed for at least 75% coverage, and 25 test vectors are needed for at least 90% coverage.



**Figure 6.** Fault coverage versus number of random test vectors applied to the s349f\_2 circuit. 12 test vectors are needed for at least 75% coverage, and 37 test vectors are needed for at least 90% coverage.