ECE 522 - VLSI circuit Testing - Fall 2019
Project Part 2 - Implement in C programming language the PODEM algorithm
Due Date: 12-06-2019

1 Objective

The objective of the second part of the project is to implement in C programming language the PODEM algorithm to generate test vectors for a given fault. Eventually you have to detect all the detectable faults and provide the corresponding test vectors for each fault. You have to implement the algorithm by using the basic functions described below and during the lab lectures. The input to the algorithm will be the circuit structure from part 1 and its fault list with size two times the number of nodes; the faults are chosen one at a time from the fault list

2 Algorithm

The basic flow of PODEM is illustrated in the following flowchart. Although it is still a deterministic search algorithm, the decisions are limited to the primary inputs. All internal signals obtain their logic values by means of logic simulation (part one of project) from the decision points. As a result, no conflict will ever occur at the internal signals of the circuit. The only possible conflicts in PODEM are either (1) the target fault is not excited or (2) the D-frontier becomes empty. In either of these cases, the search must backtrack. The algorithm examines all the possible input values till a test is found for a given fault. Initially all the primary inputs are unassigned, these inputs are assigned values one by one depending on the fault location and fault type (sa0 or sa1), the path chosen for fault propagation also determine the input assignment. The implications of primary inputs are evaluated every time a new value is assigned to an input. These implications are used to determine testability of the faults. If any input does not contribute in fault detection then it is dropped from the input list. So the algorithm basically assigns primary inputs with different values sequentially till a test vector for a given fault is found.

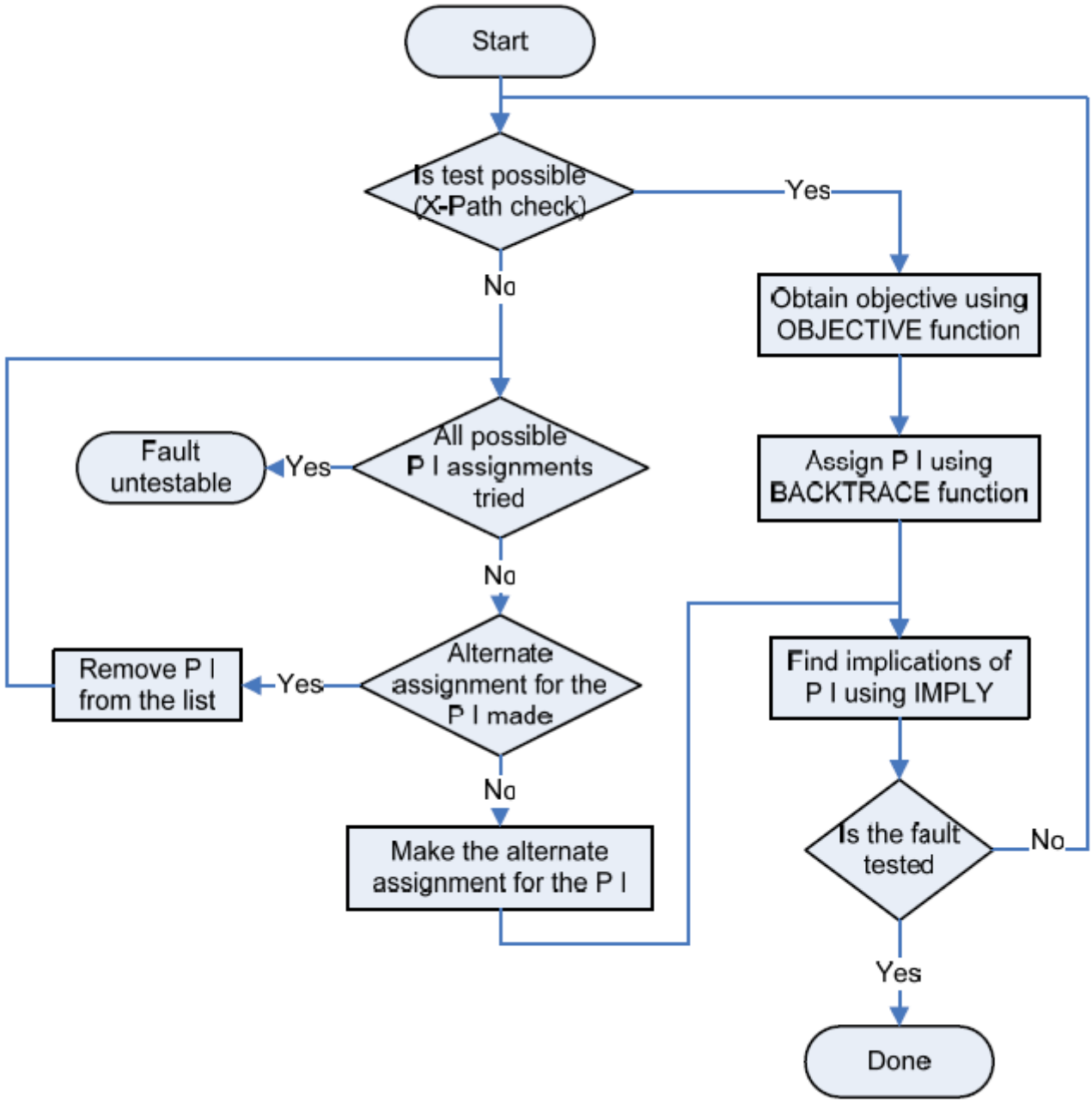

Figure 1. Flow.

3 Overview

The first ob-jective of the algorithm is to sensitize the fault. After the fault is sensitized the objectives are changed in order to propagate the fault to a Primary Output (PO).

Function OBJEC-TIVE is used to determine objectives for the program. Depending on the current objective, a function called BACKTRACE is used to determine the value of one of the PIs. For every PI assigned, logic simulation is performed to check for two conditions: desensitiza-tion of the fault and disappearance of fault propagation path (also known as X-PATH CHECK). If any one of the two conditions is violated, the program backtracks and changes the value assigned to the most recent PI. This process of assigning values to PIs is repeated till PIs form a test vector or no more combinations of PIs are possible. The latter case implies that the test is untestable

4 Major Functions

The above algorithm contains four major functions that you have to implement: Backtrace, Forward Implication, X-Path Check and Objective.

1. **Backtrace:**
   This function will take as input the target gate G and value V and backtrace until it reaches a Primary Input(PI). It will check if the target gate is PI, if yes it will return the value, if not it will decide which input has to be set in order to get the target output value V. When input has selected the coresponding gate is set as the new target gate G and the new value V is set based on the type of the gate. You repeat these steps until you reach a PI.
2. **Forward Implication:**
   This function is the function you implemented in the first part of the project. It is a logic simulations where the values of all the nodes/gates in the circuit are evaluated. These results are then used to determine the testability of the fault. This function also checks for sensitization of the fault.
3. **X-Path check:**
   This function checks for the availability of path for fault propagation. It essentially checks for existence of D-frontiers. This is done by checking for D/D on one of the in-puts of the gate on the fault propagation path and non-controlling/unknown values on the remaining inputs.
4. **Objective:**
   This function will take as input the fault location F which is the gate where the fault occurs and the fault value V. The output will be the next objective which is the next gate and value that has to be set. The first objective will be to sensitize the fault. This will be the first condition, when this is true then you have to select a path to propagate the fault to a Primary Output. The objective will be the unassigned input gate on the propagation path

5 Exercises

1. Main assignment is to implement the podem algorithm
2. Run the algorithm for every stack at fault in the circuit
3. You have to implement at least the four functions mentioned above plus any other function.

6 Submission

Submission is online through d2l. Printout submission is not needed. The submission contains below mentioned files.

Note: Codes and results should be verified by TA before submission. Unverified submission will not be considered and results in zero points. Deadline is the week before finals week, there is not extension and the TA will track your progress every week

1. All the program code files.
2. Generated vector file for each combinational circuit (**one file with all input vectors for each fault**). The output file contains the gate and fault value with corresponding input vector that detects the fault