VLSI Testing (CMPE 418/646, ENEE 646)
University of Maryland Baltimore County
Fall 2024

Homework 3: EQUIVALENT FAULT COLLAPSING
Due: Oct. 15, 2024

Goal: Write a program in the programming language of your choice to read a combinational Verilog netlist file and perform equivalent fault collapsing for the circuit. The format for the input Verilog file, output file, and expected deliverables are given below.

Deliverables

1. A .zip file containing your source code, and the output files for all 4 benchmark circuits you were given, A short Readme describing how to run your code.

Input File Format (Verilog Netlist)

1. The file begins with the “module” keyword followed by the module name and list of primary inputs and primary outputs within parenthesis and separated using ','.
   Example: module sample (N1,N2,N4);

2. The primary inputs of a circuit are identified using the keyword “input” followed by list of inputs separated using ','.
   Examle: input N1,N2;

3. The primary outputs are designated with keyword “output” and the list of outputs follows.
   Example: output N4;

4. The wires in the circuit are identified using the “wire” keyword.
   Example: wire N5;

5. The architecture is then described using the gate types mentioned below.

Gate Types

Two-input, one-output gates or One-input, one-output gates as follows:

- AND2X1
- OR2X1
- NAND2X1
- NOR2X1
- XOR2X1
- INVX1
- BUFX1

In addition to the above gates, we have some modules called *fanout$i modules*, where $*i* is an integer number. The definition of *fanout$i* module can be found in the library file (GSCLib). In practice *fanout$i* defines a fanout with one node as fanout stem and *$i* nodes as fanout branches.

6. A Gate definition includes a number of items separated by one or more spaces.

- 2-Input, 1-Output Gate:

Gate_Type Gate_name (.Y(Output_of_Gate),.A(Input_1),.B(Input_2));
Example: AND2X1 AND_1 (.Y(N5),.A(N1),.B(N2));

- 1-Input, 1-Output Gate:

Gate_Type Gate_name (.Y(Output_of_Gate),.A(Input_1));
Example: INVX1 INV_1 (.Y(N4),.A(N5));

7. All lines end with semicolon.

8. Comments are to be ignored when the file is being processed
   //single line comment
   /* Multi line
      Comment*/

9. The definition of the module ends with the keyword “endmodule”.

Sample input file: mycircuit.v

```
module sample (N1,N2,N4);

input N1,N2;
output N4;
wire N5;

//Gates in the module
AND2X1 AND_1 (.Y(N5),.A(N1),.B(N2));
INVX1 INV_1 (.Y(N4),.A(N5));
endmodule
```

Expected Deliverables

1. You are to write a program to find equivalent faults in the four given benchmark circuits. This program should have two output files. The first file *($name_BF.txt*) includes the list of all faults before applying fault collapsing scheme and the second file *($name_AF.txt*) includes the list of faults left after applying equivalence collapsing along with their equivalent collapsed faults.

In addition, your program must report:

- The total number of faults before applying your equivalence fault collapsing method (goes to *$name_BF.txt*)
- Number of faults after applying your equivalence fault collapsing method (goes to *$name_AF.txt*)
- Collapse ratio (goes to *$name_AF.txt*)
- *$name_AF*.txt also includes the list of equivalent faults (See the Output File Format)

The output file formats are as follows:

Format of Output Files

The output files for mycircuit.v are as below. In the output files the items listed in each line are separated by 1 or more spaces. Note that in Mycircuit_AF.txt, the list of equivalent faults are separated by one comma and then one or more spaces.

mycircuit_BF.txt

```
sa1     N1
sa1     N2
sa1     N4
sa1     N5
sa0     N1
sa0     N2
sa0     N4
sa0     N5

Total Faults_BF = 8
```

mycircuit_AF.txt

```
sa1     N1
sa1     N2
sa1     N4
sa0     N4

Total Faults_AF = 4
Collapse_ratio = 0.5

Equivalent Classes:
sa0 N1, sa0 N2, sa0 N5, sa1 N4
sa0 N4, sa1 N5
sa1 N1
sa1 N2
```

Note that the items written in red appear in the output files as they are.