Cpr E 281 LAB4
ELECTRICAL AND COMPUTER ENGINEERING
IOWA STATE UNIVERSITY

Introduction to Circuit Design using Truth Tables in Verilog

PRELAB!

Read the entire lab, and **complete** the prelab questions (Q1-Q3) on the answer sheet **before** coming to the laboratory.

1.0 Objectives

In the last lab we learned that Verilog is a fast and easy way to create digital circuits in Altera Quartus Prime. We will tackle a different way of doing our design implementation. We will use truth tables.

In this lab we will create truth tables using Verilog code.

- In the first part of the lab we will be learning a truth table method, while revisiting the good old farmer's problem.
- The second part is more exciting, as we will integrate schematic capture, Verilog, and truth tables to make a mixed design-entry.

2.0 Setup

Begin by creating a new folder **U:\Documents\CPRE281\Lab04**, you will be saving your work for this lab in this folder.

3.0 Truth Tables using Verilog (Lab4step1)

One way to define a truth table in Verilog is to use the **case** statement. There are other methods of defining a truth table but we will use the **case** statement in this lab. We will give an example on how to enter the following truth table:

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

```
module truth_table_verilog_example (F, A, B, C);
        input A, B, C;
        output F;
        reg F;

        always @(A or B or C)
        begin   //begin the procedural statements
                case ({A,B,C})  //variables that affect the procedure

                                        //the 3'b simply means that we're
                        3'b000: F= 'b0; //considering three bit binary variable
                        3'b001: F= 'b0;
                        3'b010: F= 'b1;
                        3'b011: F= 'b0;
                        3'b100: F= 'b1;
                        3'b101: F= 'b0;
                        3'b110: F= 'b1;
                        3'b111: F= 'b0;
                endcase
        end     //end the begin statement

endmodule
```

Now, create a new project and write a code for the farmer's problem, (including the farmer as in Lab03 Section 4.0), using the truth tables method in Verilog. Refer to Lab03 to review the farmer's problem's statement.

You should get your code to run on ModelSim. Make sure to take screenshots and include them in your lab report.

4.0 Mixed Design Entry

Now that we know how to design using truth tables, we will try to integrate all the design entry methods we have learned so far (schematic capture, Verilog, and truth tables) into one single problem. The problem is described below:

Description:

A house has an air-conditioner (A/C) that can work in two modes, Normal, or Powersaving. Through mode controllers (Normal Mode and Powersaving Mode blocks in the figure below), the A/C is connected to various sensors in the house. The sensors will be reading the temperature, the humidity and whether there is somebody in the house or not. Thus the input set is: $inputs = \{Temperature, Humidity, People\}$, or just T, H and P for short. Each of these sensors has a set value and when the sensor's reading drops below that set value, it will output a logical 0. If the sensor's reading stays above the value it will output a logical 1. For example, if the temperature is above 80 F the output is a logical 1, otherwise It is a logical 0.

Each controller receives the sensor values and produces its output that is connected to a 2-to-1 multiplexer. The multiplexer can choose between the two different modes its output operates the A/C. More information on the controllers is given in the text below the diagram. The basic diagram of the sensors and controllers is as follows:

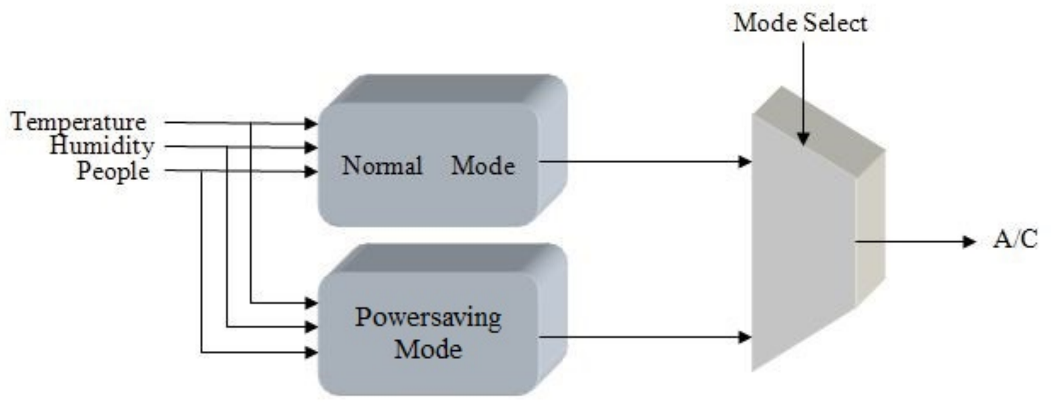

Figure 1: Diagram showing the high level view of the A/C system

<u>Normal Mode:</u>
$inputs = \{Temperature, Humidity, People\}, \ output = \{E\}$
$E = P(T+H) + \bar{P}(T \cdot H)$, simply means that the A/C should be ON (in other words, has the value of a logical 1) when somebody is present and there is either high temperature or high humidity <u>or</u> when no one is in the house and both the temperature and humidity are high. You have to enter this block in Verilog code.

<u>Powersaving Mode:</u>
$inputs = \{Temperature, Humidity, People\}, \ output = \{F\}$
You have to enter this block in Verilog code. The logical expression is: $F = P \cdot T \cdot H$, meaning that the A/C should be ON when someone is in the house and both the

temperature and the humidity are high.

<u>Multiplexer:</u>
$inputs = \{M, E, F\},\ outputs = \{AC\}$
You have to enter this block in Verilog. When M = 0, we will choose Normal Mode (E), when M = 1, we will choose Powersaving Mode (F).

- Start a new project and name it *lab4step2*. Ensure that you start the new project in a new folder (by the same name). Open a new Block Diagram file. This is the Graphic Editor file where all the components of the A/C system will be instantiated.
- Next, create the blocks that would represent the A/C components. Click on the block tool from the upper toolbar (see Figure 2). Left click anywhere on the blank block diagram file and drag your mouse. Repeat this process 2 more times. The three blocks would represent the components of your A/C system. Right click on the 1st block and click **Properties**, rename the block *normal*, click on I/O tab and enter P,T,H as inputs and E as the output. Repeat the same step for the other blocks. Make sure that the naming convention is same as mentioned in the problem statement above.
- Enter 4 inputs pins (P, H, T, M) and one output pin (AC) in the design file.
- Click on the **Orthogonal Conduit Tool** from the upper toolbar and connect the input pins, output pin and the blocks using the Conduit. This would automatically map the output of one block onto the input of the other block, assuming the names are the same. Refer to the figure below to make sure your design is right.

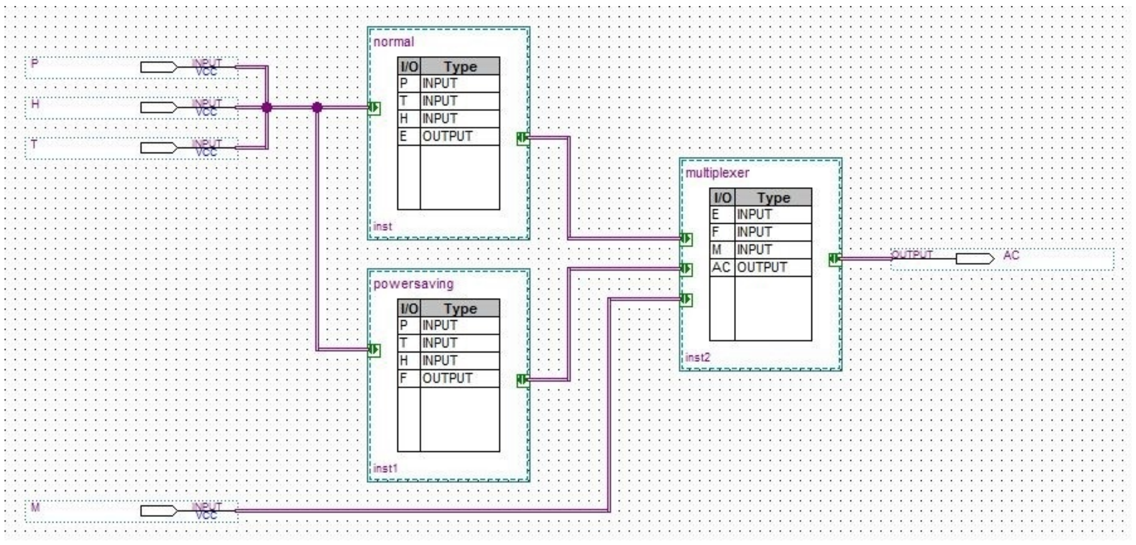

Figure 2: Final Layout of *Lab4step2*.

- Now that the design is done, write the back-end code for the components. Right click on the *normal* block and click on **Create Design File from selected Block**. Choose Verilog and click OK. This will open a Verilog file. Write the code for the block before the endmodule statement. Be aware that the inputs and outputs have already been declared. Repeat this step for the rest of the blocks.
- Compile the project and fix any errors. Use ModelSim to verify your circuit. This time make sure that in the modelsim, you **compile all** the files that are created (There is Verilog file created for each corresponding block)!

5.0 Complete

You are done with this lab. Ensure that all lab files are closed, exit Quartus Prime, log off the computer, power down the DE2-115 board, and submit your answer sheet. **Don't forget to write down your name, student ID, and your lab section number**.