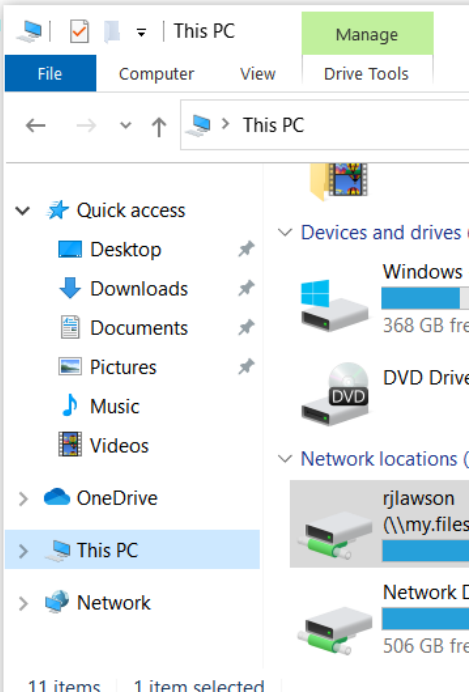
Cpr E 281 LAB4
ELECTRICAL AND COMPUTER ENGINEERING
IOWA STATE UNIVERSITY

Lab 4 Answer Sheet

```
// Generated by Quartus Prime Version 20.1 (Build Bu
// Created on Tue Sep 22 13:21:47 2020

//  Module Declaration
module Multiplexer
)(
// {{ALTERA_ARGS_BEGIN}} DO NOT REMOVE THIS LINE!
E, F, M, AC
// {{ALTERA_ARGS_END}} DO NOT REMOVE THIS LINE!
);
// Port Declaration

// {{ALTERA_IO_BEGIN}} DO NOT REMOVE THIS LINE!
input E;
input F;
input M;
output AC;
// {{ALTERA_IO_END}} DO NOT REMOVE THIS LINE!

   reg AC;

   always @(E or M or F)

   if(M == 0)
      AC = E;

   else
      AC = F;

endmodule
```



Name and Student ID: Riley Lawson rjlawson@iastate.edu Lab Section: 6

Date: 9/17/2020

Submission Instructions:

Prelab:

1. Complete the prelab (questions Q1 and Q2)
2. Submit this report with the prelab completed to Canvas **before** your lab starts

Lab:

1. Complete the lab according to the instructions
2. Take screenshots of your ModelSim waveform (note: to receive points your NetID has to be present in the screenshot) and insert them into this document.

Complete this report and reupload it to Canvas.

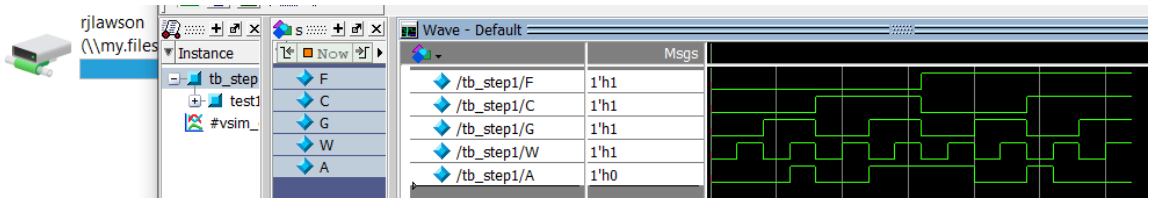
PRELAB:

Q1. Consider the Verilog code in section 3.0. Briefly explain how the **always @** structure works.

Always: blocks are used to describe events that should happen certain condition.

Q2. Write the Verilog code for ***lab4step1***. Use the example code given in Section 3.0 and make the necessary changes.

```
module lab4step1(A, F, C, G, W);
   input F, C, G, W;
   output A;
   reg A;

   always @(C or G or W or F)
   begin
      case({C, G, W, F})

         4'b0000: A = 'b0;
         4'b0001: A = 'b0;
         4'b0010: A = 'b0;
         4'b0011: A = 'b1;
         4'b0100: A = 'b0;
         4'b0101: A = 'b0;
         4'b0110: A = 'b1;
         4'b0111: A = 'b1;
         4'b1000: A = 'b1;
         4'b1001: A = 'b1;
         4'b1010: A = 'b0;
         4'b1011: A = 'b0;
         4'b1100: A = 'b1;
         4'b1101: A = 'b0;
         4'b1110: A = 'b0;
         4'b1111: A = 'b0;
      endcase
   end

endmodule
```



Lab 4 Answer Sheet

Q3. Read Section 4.0 and fill in the Truth Table for ***lab4step2***.

Inputs				Outputs		
M	**T**	**H**	**P**	**E**	**F**	**AC**
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	0	1	1	1	0	1
0	1	0	0	0	0	0
0	1	0	1	1	0	1
0	1	1	0	1	0	1
0	1	1	1	1	1	1
1	0	0	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	0	0	0
1	0	1	1	1	0	0
1	1	0	0	0	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	1	1	1

Lab 4 Answer Sheet

LAB:

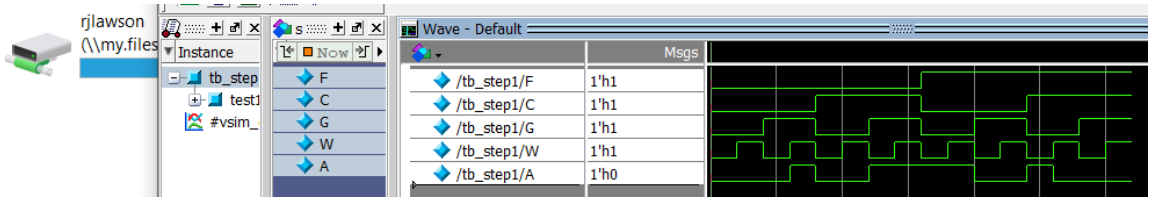
3.0 Use the hardware results to fill in the truth table for ***lab4step1.***

Farmer	Cabbage	Goat	Wolf	Alarm
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Screenshots:

<<<Copy and paste your Verilog code here>>>

```
module lab4step1(A, F, C, G, W);
    input F, C, G, W;
    output A;
    reg A;

    always @(C or G or W or F)
    begin
        case({C, G, W, F})

            4'b0000: A = 'b0;
            4'b0001: A = 'b0;
            4'b0010: A = 'b0;
            4'b0011: A = 'b1;
            4'b0100: A = 'b0;
            4'b0101: A = 'b0;
            4'b0110: A = 'b1;
            4'b0111: A = 'b1;
            4'b1000: A = 'b1;
            4'b1001: A = 'b1;
            4'b1010: A = 'b0;
            4'b1011: A = 'b0;
            4'b1100: A = 'b1;
            4'b1101: A = 'b0;
            4'b1110: A = 'b0;
            4'b1111: A = 'b0;
        endcase
    end

endmodule
```

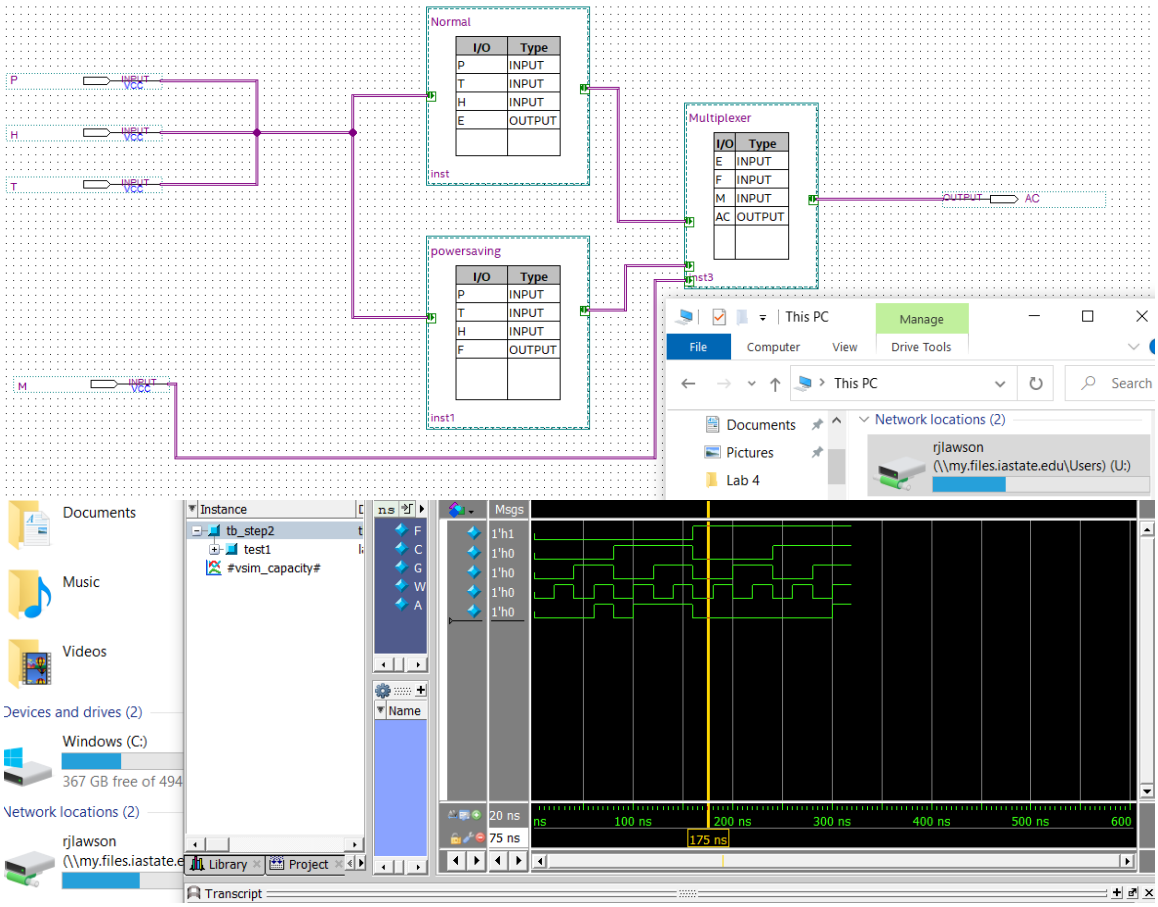


4.0 Demonstrate hardware results for correct code.

<<<Insert a screen shot of your lab4step2 BDF file>>>

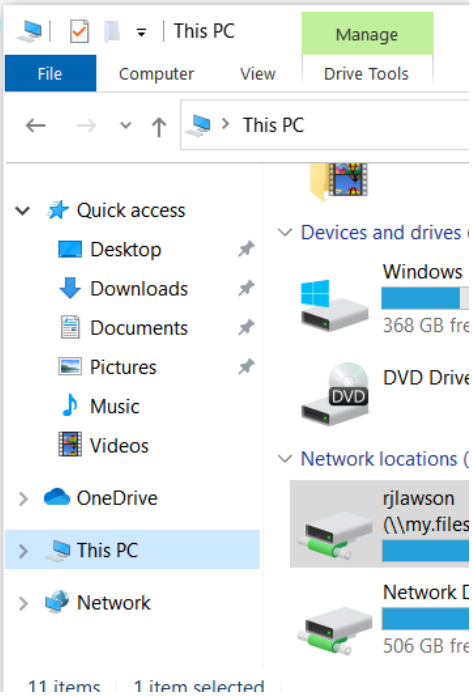
Lab 4 Answer Sheet

Design result:

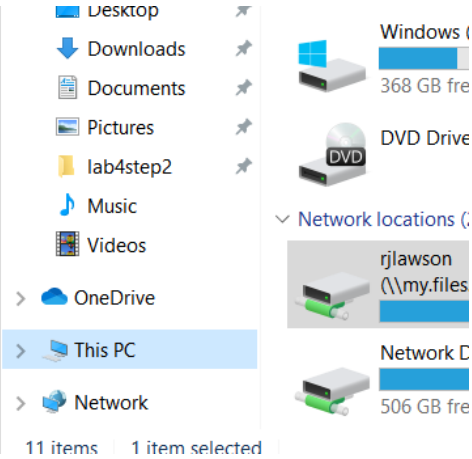


Lab 4 Answer Sheet

```
// Generated by Quartus Prime Version 20.1 (Build Bu
// Created on Tue Sep 22 13:21:47 2020

//  Module Declaration
module Multiplexer
(
// {{ALTERA_ARGS_BEGIN}} DO NOT REMOVE THIS LINE!
E, F, M, AC
// {{ALTERA_ARGS_END}} DO NOT REMOVE THIS LINE!
);
// Port Declaration

// {{ALTERA_IO_BEGIN}} DO NOT REMOVE THIS LINE!
input E;
input F;
input M;
output AC;
// {{ALTERA_IO_END}} DO NOT REMOVE THIS LINE!

   reg AC;

   always @(E or M or F)

   if(M == 0)
      AC = E;

   else
      AC = F;

endmodule
```




```
// Module Declaration
module Normal
(
// {{ALTERA_ARGS_BEGIN}} DO NOT REMOVE THIS LINE!
P, T, H, E
// {{ALTERA_ARGS_END}} DO NOT REMOVE THIS LINE!
);
// Port Declaration

// {{ALTERA_IO_BEGIN}} DO NOT REMOVE THIS LINE!
input P;
input T;
input H;
output E;

// {{ALTERA_IO_END}} DO NOT REMOVE THIS LINE!

assign E = ((P & (T | H)) | (~P & T & H));

endmodule
```



Lab 4 Answer Sheet

```
module powersaving
(
// {{ALTERA_ARGS_BEGIN}} DO NOT REMOVE THIS LINE!
P, T, H, F
// {{ALTERA_ARGS_END}} DO NOT REMOVE THIS LINE!
);
// Port Declaration

// {{ALTERA_IO_BEGIN}} DO NOT REMOVE THIS LINE!
input P;
input T;
input H;
output F;
// {{ALTERA_IO_END}} DO NOT REMOVE THIS LINE!

assign F = P & T & H;

endmodule
```