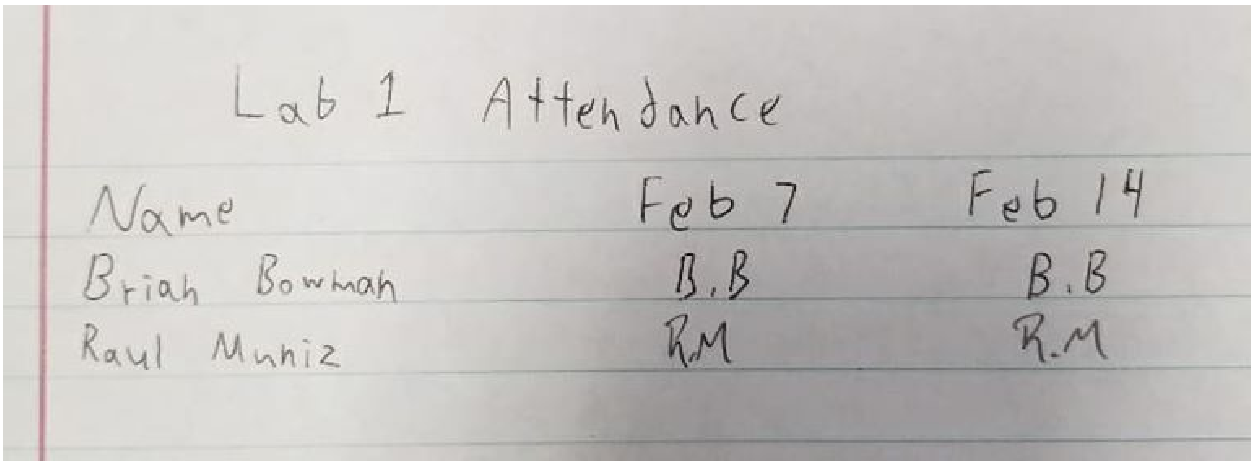
Lab1
Brian Bowman
Raul Muniz



Lab 1 Attendance

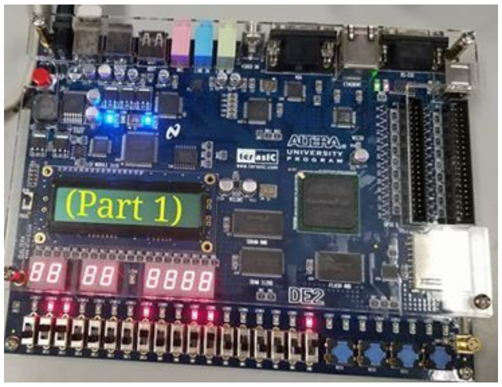
Name	Feb 7	Feb 14
Brian Bowman	B.B	B.B
Raul Muniz	R.M	R.M

Part 1

I. **Problem:** Have each switch control the red LED above it.
II. **Conceptual Design:** LEDR[0] is above SW[0], LEDR[1] is above SW[1], and so on, so all we need to do is assign SW to LEDR.
III. **Verilog Design:**

```
module part1 (input [17:0] SW, output [17:0] LEDR);
     assign LEDR = SW;
Endmodule
```

IV. **Lab Procedure:**
<u>**Picture**</u>



Troubleshooting: The only complication we had with this part was that the DE2 board would not light up. We later figured out it was due to the power outlet it was connected to wasn't functioning which after we connected to a new outlet it powered the board successfully.
Conclusion: We learned the basic connections of setting up Quartus 2 with the DE2 board.

Part 2

I. **Problem:** Implement an eight-bit wide 2-to-1 multiplexer.

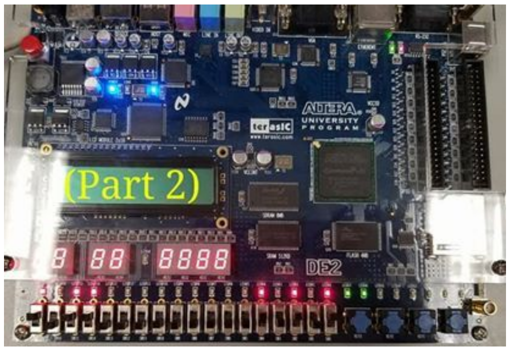
II. **Conceptual Design:** The two mux inputs are the two spans of switches SW[7:0] and SW[15:8]. The select signal is the leftmost switch, SW[17]. The output of the mux is visualized on the green LEDs. We also visualize the state of the switches on the red LEDs.

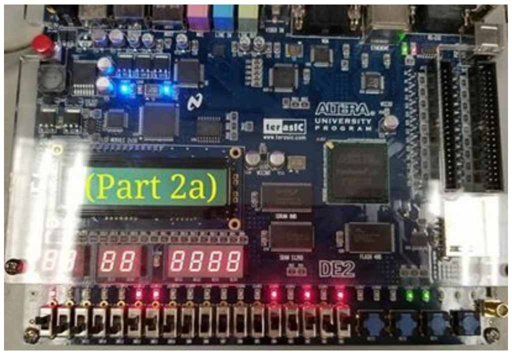
III. **Verilog Design:**

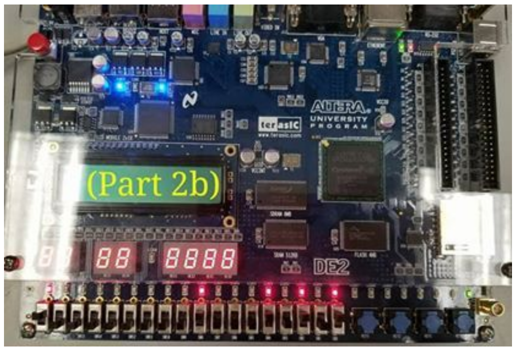
```
module part2 (SW, LEDR, LEDG);
     input [17:0] SW;
     output [17:0] LEDR;
     output reg [7:0] LEDG;

     assign LEDR = SW;
     always @ (*) begin
          if (SW[17])
               LEDG = SW[15:8];
          else
               LEDG = SW[7:0];
     end
Endmodule
```

IV. **Lab Procedure:**

Pictures







Troubleshooting: We had to restart the circuit on the computer once the code seemed correct and the procedure was executed flawlessly.

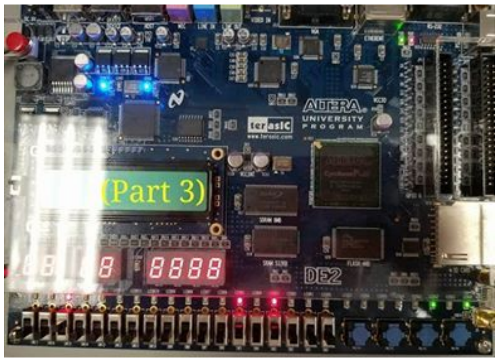
Conclusion: We've learned that each time we run a different type of instructions/code on the machine, we have to restart almost everything as if it were the first circuit we constructed for the day.

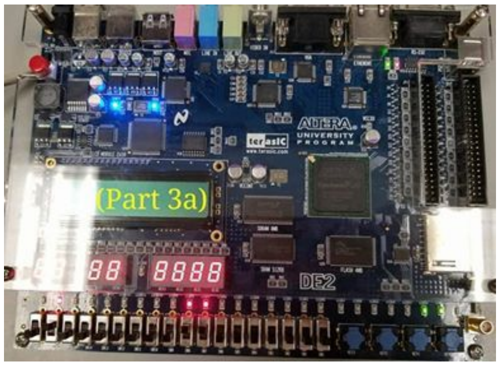
Part 3

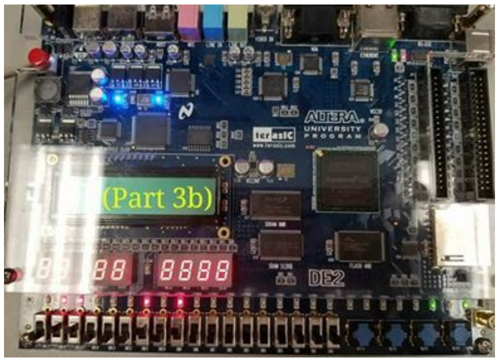
I. **Problem:** Implement a three-bit wide 5-to-1 multiplexer.
II. **Conceptual Design:** Each mux input corresponds to a span of three switches, beginning with SW[2:0] and ending with SW[14:12]. The select signal are the three leftmost switches, SW[17:15]. The output of the mux will be visualized on the green LEDs. We also visualize the state of the switch inputs on the red LEDs.
III. **Verilog Design**

```
module part3 (SW, LEDR, LEDG);
     input [17:0] SW;
     output [17:0] LEDR;
     output reg [2:0] LEDG;

     assign LEDR = SW;
     always @ (*) begin
          case (SW[17:15])
               3'b000 : LEDG = SW[2:0];
               3'b001 : LEDG = SW[5:3];
               3'b010 : LEDG = SW[8:6];
               3'b011 : LEDG = SW[11:9];
               3'b100 : LEDG = SW[14:12];
               default : LEDG = 3'b000;
          endcase
     end
Endmodule
```

IV. **Lab Procedure:**
<u>**Pictures**</u>







Part 4

I. **Problem:** Implement a 7-segment decoder that can display the letters “H”, “E”, “L”, and “O”.

II. **Conceptual Design:** The inputs to the decoder are the three rightmost switches SW[2:0]. The output of the decoder is visualized on the green LEDs. We also visualize the state of the switch inputs on the red LEDs.

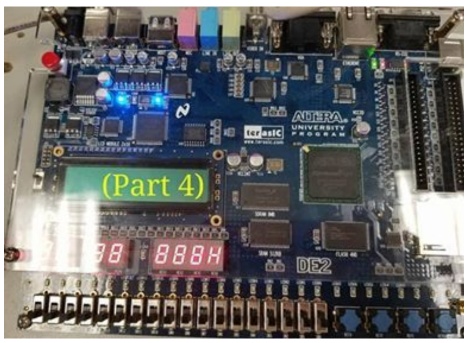
III. **Verilog Design:**

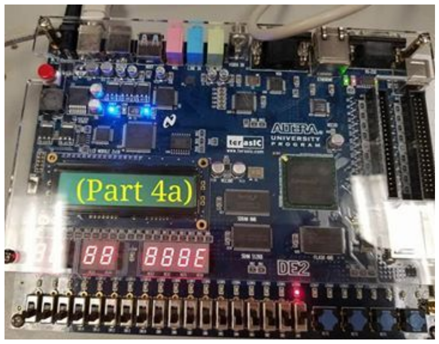
```
module part4 (SW, LEDR, HEX0);
     input [2:0] SW;
     output [17:0] LEDR;
     output reg [6:0] HEX0;

     assign LEDR = SW;
     always @ (*) begin
          case (SW[2:0])
               3'b000 : HEX0 = 7'b0001001; // H
               3'b001 : HEX0 = 7'b0000110; // E
               3'b010 : HEX0 = 7'b1000111; // L
               3'b011 : HEX0 = 7'b1000000; // O
               default : HEX0 = 7'b1111111; // blank
          endcase
```

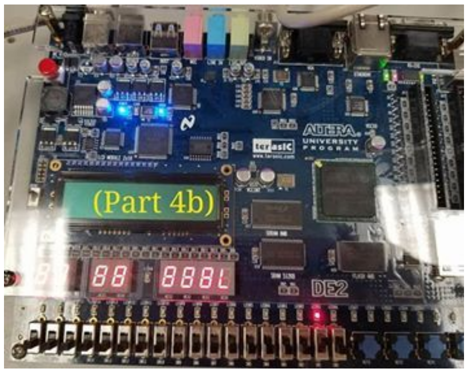
```
      end
Endmodule
```

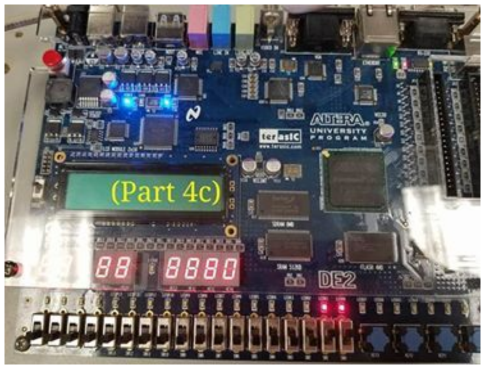
IV. Lab Procedure:

Pictures:









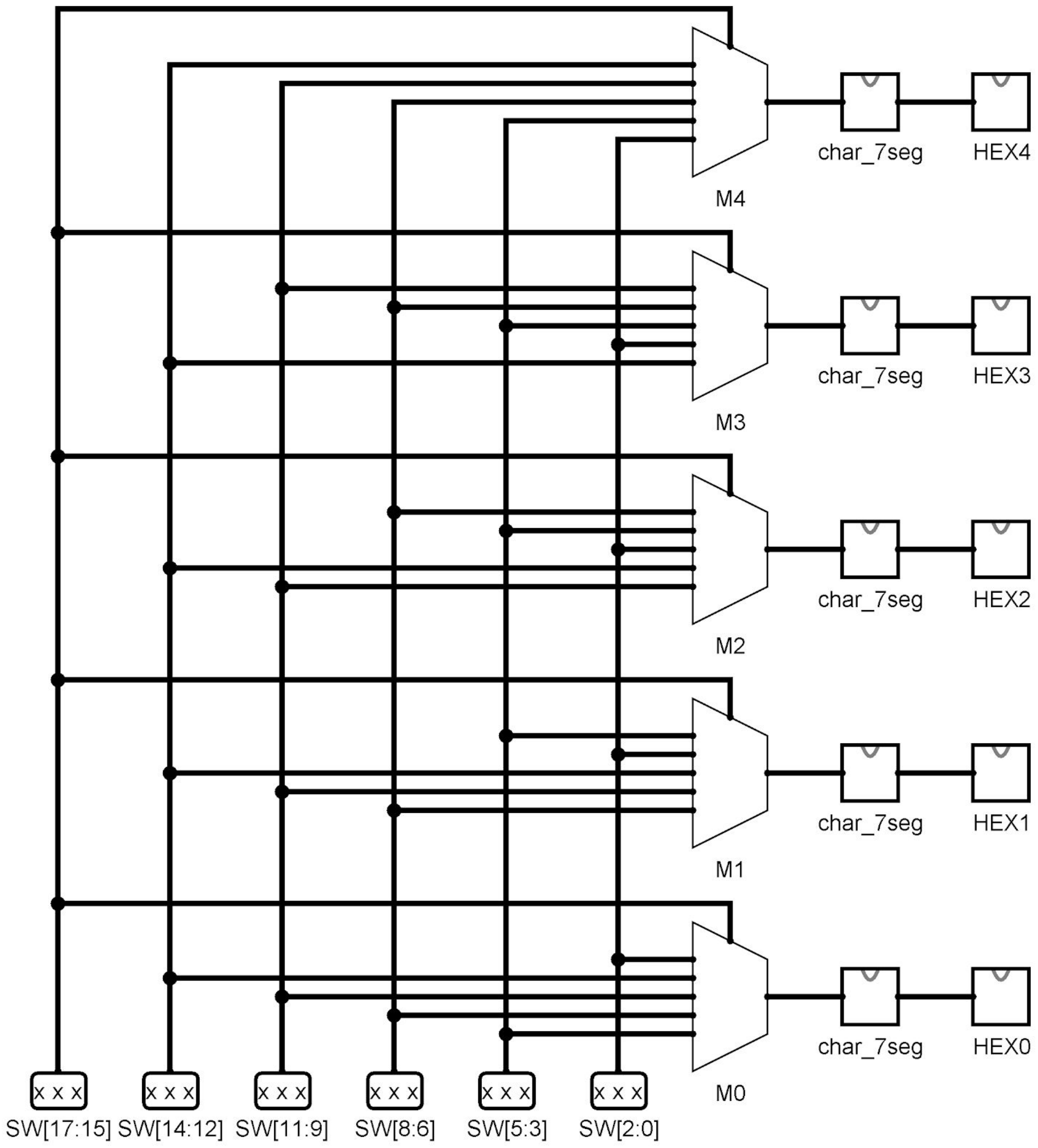
Troubleshooting: After downloading the compiled circuit into the FPGA, we flipped the input switches to cycle through the characters. The characters didn't look right at first; most looked mirrored. We found out the problem was in our declaration of output HEX0. We had written "`[0:6]`" instead of "`[6:0]`". After fixing this, the FPGA worked as expected.

Conclusion: We've learned that in order for the circuit to successfully compile, we need to be sure that the code has the correct numerization with the proper algorithm. As well as how we have the potential output almost anything on the board.

Part 5

I. **Problem:** Construct a circuit that can can be programmed to display a sequence of characters over five 7-segment displays, and have the ability to rotate the word around the five 7-segment displays.
II. **Conceptual Design:** Five spans of three switches, starting with SW[2:0] and ending with SW[14:12], control what characters are displayed on the 7-segment displays. The three leftmost switches, SW[17:15], control the rotation of the characters. The output is visualized on the five rightmost 7-segment displays HEX0-HEX4.

Lab1
Brian Bowman
Raul Muniz



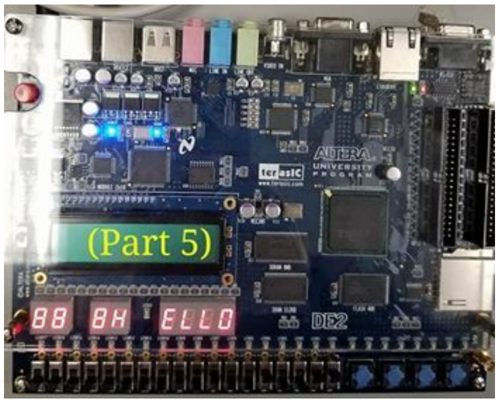
III. Verilog Design:

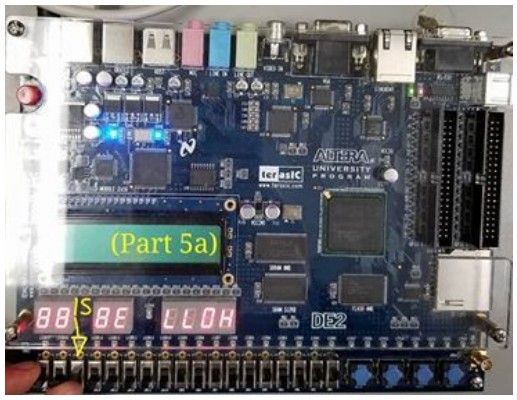
```
module lab1part5 (SW, HEX0, HEX1, HEX2, HEX3, HEX4);
     input [17:0] SW;
     output [6:0] HEX0, HEX1, HEX2, HEX3, HEX4;
```

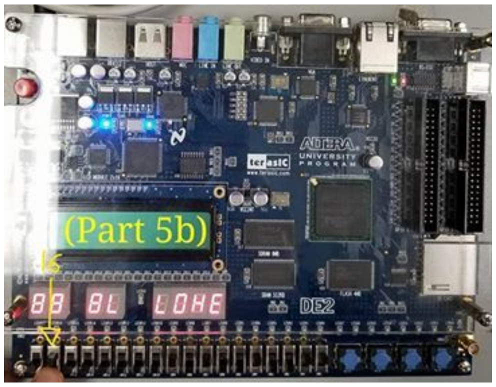
```
	wire [2:0] m0, m1, m2, m3, m4;

	mux_3bit_5to1 M0 (SW[17:15], SW[2:0], SW[14:12], SW[11:9], 
SW[8:6], SW[5:3], m0);
	char_7seg H0 (m0, HEX0);

	mux_3bit_5to1 M1 (SW[17:15], SW[5:3], SW[2:0], SW[14:12], 
SW[11:9], SW[8:6], m1);
	char_7seg H1 (m1, HEX1);

	mux_3bit_5to1 M2 (SW[17:15], SW[8:6], SW[5:3], SW[2:0], 
SW[14:12], SW[11:9], m2);
	char_7seg H2 (m2, HEX2);

	mux_3bit_5to1 M3 (SW[17:15], SW[11:9], SW[8:6], SW[5:3], SW[2:0], 
SW[14:12], m3);
	char_7seg H3 (m3, HEX3);

	mux_3bit_5to1 M4 (SW[17:15], SW[14:12], SW[11:9], SW[8:6], 
SW[5:3], SW[2:0], m4);
	char_7seg H4 (m4, HEX4);
endmodule
```

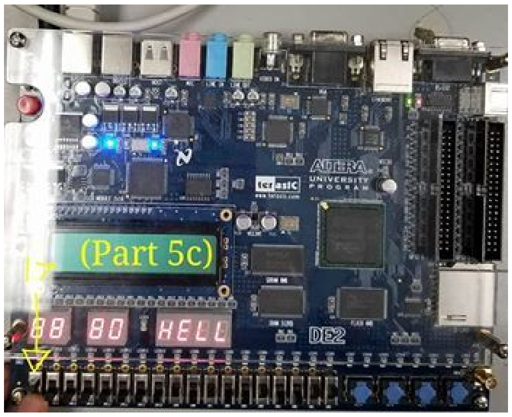
IV. Lab Procedure:

<u>Pictures:</u>









Troubleshooting: Going into the lab, we had the code ready to execute on the board. When we tested it out, instead of the letters (HELLO) on the LEDR shifting left they shifted right. We might have over-thought this minor error that we wasted a good amount of time trying to find the solution. We would comment out the previous order of the verilog code to see what other orders we've experimented with to be sure we don't use the same code again. After a few lines of experimenting code, we came to the conclusion that the code we started off with was correct, but it was just upside down. In other words, 4 to 0, instead of 0 to 4.
Conclusion: In conclusion, we've learned that in order to get the code to execute the way it should, trial and error is always a helpful tool to see what error might be occurring. After finishing the lab, it also occurred to us that we could have turned off the three leftmost hex displays, which we didn't use. This would have made the output neater and easier to see.

ELEE/CMPE 4303 Digital II **Practice Mid-Exam**
Spring 2018

1. Multiple Choice(25 points)

1) 4’b1001 represents the following decimal number_____b) 9_________

a) 8
b) 9
c) 10
d) 11

2) The following Verilog statement can be used to define X as logic OR of (Y,Z);

a) or (X,Y,Z)
b) or (Y,Z,X)
c) OR (X,Y,Z)
d) OR (Y,Z,X)

3) If A=4’b1010, B=8’b10101010, C is defined as wire [11:0], C={A, B}, then C[8:7]=_____b) 2’b01_____

a) 2’b00
b) 2’b01
c) 2’b10
d) 2’b11

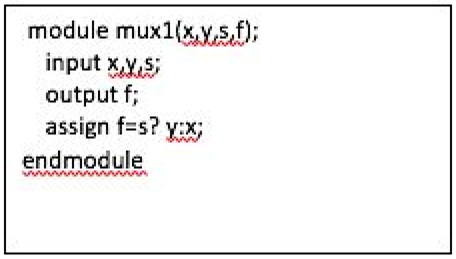
4) To convert a 4-bit SW input to 7-segment code HEX0 display output, we can instantiate a module bcd2leds(hex, leds) which has hex as 4 bit input, leds as 7 bit ouput.
as____________________

a) bcd2leds U1 (HEX0, SW);
b) U1 bcd2leds (HEX0, SW);
c) bcd2leds U1 (SW, HEX0);
d) U1 bcd2leds (SW,HEX0);

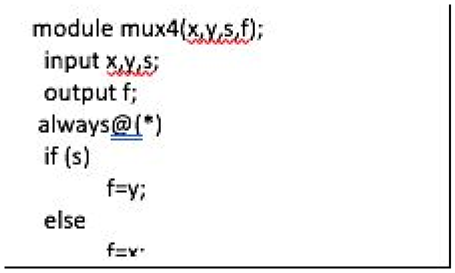
Lab1
Brian Bowman
Raul Muniz

5) Read the following four designs for a 2:1 multiplexer, the correct design(s) is(are)

a) All of I, II, III, IV
b) II only
c) [✓ I, IV only]
d) II and IV only



```
module mux1(x,y,s,f);
  input x,y,s;
  output f;
  assign f=s? y:x;
endmodule
```



```
module mux4(x,y,s,f);
 input x,y,s;
 output f;
always@(*)
 if (s)
        f=y;
 else
        f=x;
```

1) Write a Verilog module for the circuit in Fig.1 using primitive gates. (15 points)

```
module verilog1(x1,x2,x3,x4,f1,f2);
        input x1,x2,x3,x4;
        output f1,f2;
        wire a,b,c,d,e;
        or(a,x1,x2);
        not(b,a);
        or(c,x3,x4);
        and(d,b,c);
        and(e,x3,x4);
        or(f1,d,e);
        and(f2,a,c);
endmodule
```

2) Write a Verilog module for the circuit in Fig.1 using continuous assignments.(15 points)

```
module verilog1(x1,x2,x3,x4,f1,f2);
        input x1,x2,x3,x4;
        output f1,f2;
        assign f1=( (~x1|x2) ) & (x3|x4) | (x3&x4);
        assign f2=(x3|x4) & (x1|x2);
endmodule
```

Lab1

Brian Bowman
Raul Muniz

Two submodules partA and partB have been designed as two Verilog files partA.v and partB.v, readily available for use.
The diagram of mycircuit is shown in Fig.2. Write a Verilog file for the top module part of module mycircuit by instantiating modules partA and partB. (20 points)

```
module partA (x,y,d);
input x,y;
output reg [1:0] d;
always @(x,y)
begin
d[0]<=x ^ y;
d[1]<=x & y;
end
endmodule

module partB (w,s);
input [2:0] w;
output s;
assign s=w[0]^w[1]^w[2];
endmodule

module mycircuit(X,f);
input [1:0] X;
output f;
wire b1,b2;
partA a1 (.x(X[0]),.y(X[1]),.d(1)(b1),.d[0](b2));
partB a2 (.w[0](b2),.w[1](b1),.w[2](1’b0),.s(f));
endmodule
```

A half-adder HA is defined by the truth table

Inputs *outputs*

x *y*		*carry* *sum*	

Write a Verilog file for module HA.

Lab1
Brian Bowman
Raul Muniz

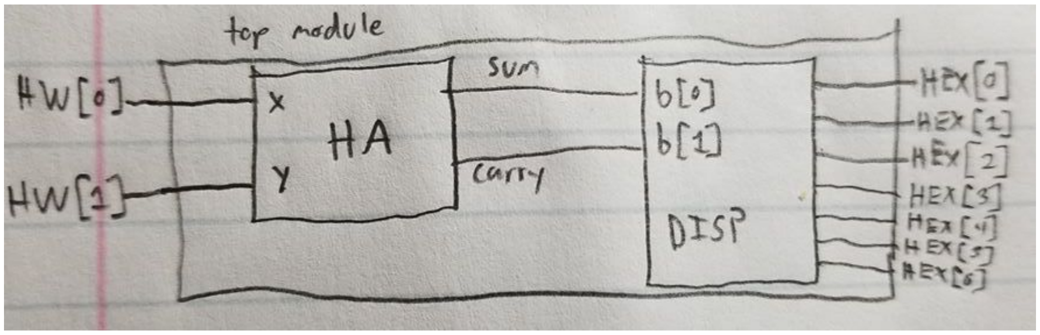
```
module HA (x,y,sum,carry); // Half adder
input x,y;
output sum,carry;
assign sum=x^y;
assign carry=x&y;
endmodule
```

(b) (10 pts) A two bit display DISP on a **low active 7 segment LED** is to display 0,1, 2, 3 for the two bit input 00, 01, 10, 11. Complete the Verilog design for module DISP

```
        module DISP (b, leds);
                input [1:0] b; // two bit input
                output reg[6:0] leds; // corresponding 7 segment display of the two bit number
                always @(b)
                Begin
                case(b)
                2'b00:leds=7'b0000001;
                2'b01:leds=7'b1001111;
                2'b10:leds=7'b0010010;
                2'b11:leds=7'b0000110;
                endcase
                end
                endmodule
```

(c) (10 pts) A top module test is used to demostrate the half-adder and display the result on DE2 board. SW[1:0] are used for the two input bits. HEX0[6:0] are used for the display output. Use sub-modules HA and DISP as defined above.

i. Draw the block diagram for the top module showing all inputs, outputs, and internal connections for the submodules



ii. Write the Verilog design codes for the top module.

```
module top(HW,HEX); //top module
input [1:0] HW;
output [6:0] HEX;
wire [1:0]A;
HA a1 (HW[0],HW[1],A[0],A[1]); //half adder instantaition
DISP a2 (A,HEX); // 7 segment display driver instantiation
endmodule
```