ECEN 2350: Digital Logic

Assignment #6

In this lab, you'll be using the Boolean Board to output to the 7-segment display.

You can find a reference to the Boolean Board's 7-segment display here: `https://www.realdigital.org/doc/586fb4c3326dcd493a5774b2a6050f41`

Documentation for setting up Xilinx Vivado for the Boolean Board is here: `https://www.realdigital.org/doc/c4ceeb20d229e5f3d4e32f3a74e343e9`

1. [10 points.] Write a verilog module named `hexEncode` that takes a 4-bit binary number as input, and outputs 8 bits that encode the number as a hexidecimal number for displaying on the Boolean Board's 7-segment display. Shown below is the layout for the Boolean Board's 7-segment display.

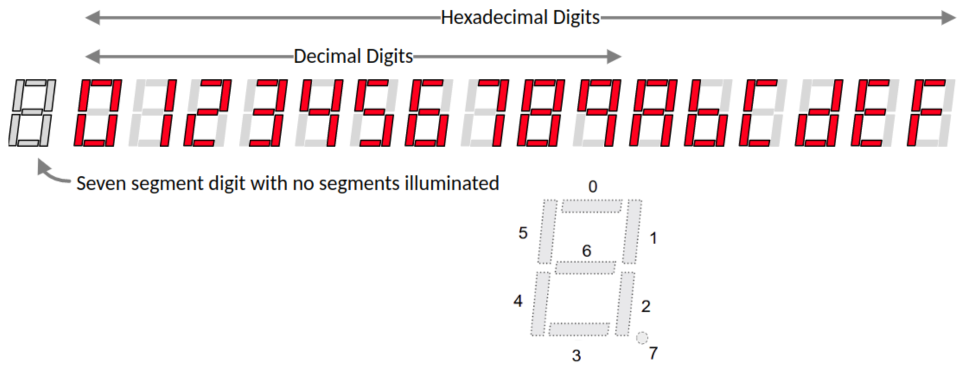

Hexadecimal Digits
Decimal Digits
0 1 2 3 4 5 6 7 8 9 A b C d E F
Seven segment digit with no segments illuminated
0
5
6
1
4
2
3
7

Figure 1: Source: `https://www.realdigital.org/`

There are 7-segments of each digit (and one for the decimal point), and each segment can be turned on or off independently. Note that on the Boolean Board, these segments are **active low**, meaning to turn it on, you set the segment to a 0. Setting a segment to 1 would turn it off.

For instance, to make the display show a `4`, you would set:

- segments 0, 3, and 4 to a 1 (turning them off),

- segments 1, 2, 5, and 6 to a 0 (turning it on),
- segment 7 to a 1 (turning off the decimal point).

If you encoded this to binary (little endian, starting with bit 7), you would get the number $8'b10011001$, or $0x99$. Thus, if your module takes as input the number 4 ($4'b0100$), it should output $8'b10011001$.

Shown above is the hex displays for the digits 0-9 and A-F for reference.

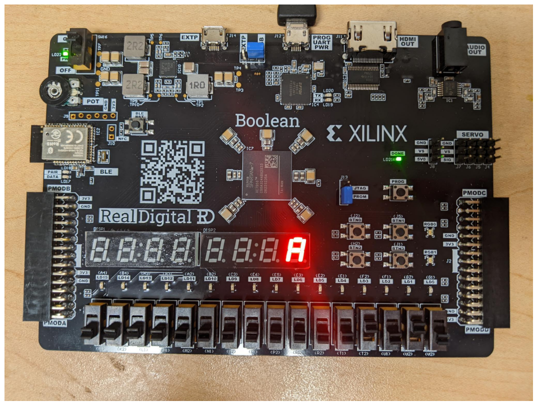
Your verilog module should have the following declaration:

```
module hexEncode(input [3:0] bin, output [7:0] hex);
```

We encourage you to test your code with the provided testbench (`hexEncode-tb.v` in Canvas). This testbench will print out "Incorrect ..." if you have a case that does not work properly, and you can run this in the EDA Playground.

For this part, submit a file called `hexEncode.v` that just contains your `hexEncode` verilog module.

2. [5 points.] Take your code from Part 1, and use it to take the right-most 4 switches from the Boolean Board, and display it as a hexadecimal number on the right-most 7-segment display. For instance, your Boolean Board should look like this for the provided switch configuration shown (0xa):



Turn in a verilog for this part named `lab6-q2.v` that contains your top-level verilog file and any modules needed to run it (e.g. include the hexEncode from Part 1).

3. [10 points.] Your solution from Part 2 shows only one hexadecimal digit[1]. In this part, you'll use the **buttons** on the Boolean board to select between which of the 4 7-segment displays will be output at a time.

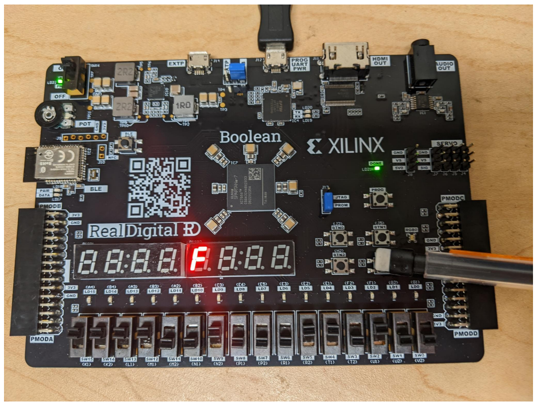
   This will allow you to "display" the full 16-bit switch value, converted to hexadecimal, on the right-most 7-segment module, one nibble at a time.

   If BTN3 is pressed, you should display the hex nibble equivalent to the binary number from the left-most 4 switches (SW15-SW12) on the left-most 7-segment display of the right unit.

   If BTN2 is pressed, you should display the hex nibble of 4 switches SW11-SW8 to the second 7-segment on the right.

   If BTN1 is pressed, you should display the hex nibble of the switches SW7-SW4 on the third 7-segment, and finally if BTN0 is pressed, show the hex nibble of SW3-SW0 on the right-most 7-segment (as in Part 1)

   For instance, if you had the following configuration and pressed BTN3, it should show this on the display:

   Here's a video showing an example of what the result should look like, when the switches are in the configuration 0001001000110100 (hex $0x1234$): `https://youtu.be/dI9g5vHKqA8`

   Note: You will have to handle the case where more than one button is pressed at once. In that case, you should prioritize the highest button (e.g. BTN3 has priority over BTN2). Check what happens in your design when you press more than one button at once to see that it outputs to the expected display.

   Turn in your verilog in a file named `lab6-q3.v`

[1] a so-called "nibble", as it is half of a byte