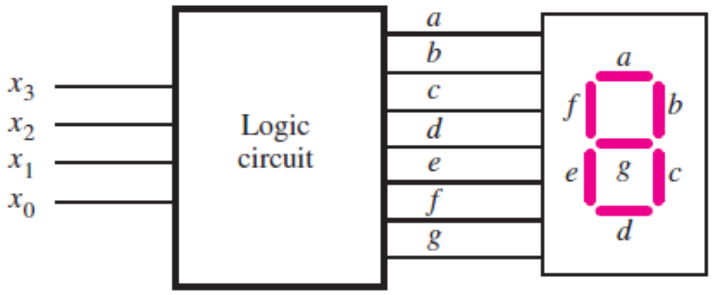
LAB 1 – ECE 3300L

Fall 2024 / Dr. Van Blerkom

The objective of this lab is to display numbers on a 7-segment display. The specific number displayed depends on a four-bit input. The figure below shows a *7-segment decoder* module that has a four-bit input ($x_3x_2x_1x_0$). This decoder produces seven outputs that are used to display a number on a 7-segment display. The truth table shows the values of the output *a* through *g*, for the binary inputs from 0 to 9.



(a) Logic circuit and 7-segment display

	x_3	x_2	x_1	x_0	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

(b) Truth table

To make our Verilog implementation less cluttered, instead of making individual signals *a* through *g* in the code, we will use a 7-bit bus to address the LEDs. Let's call this bus "leds [6:0]":

- leds[0] => LED *a*
- leds[1] => LED *b*
- leds[2] => LED *c*
- *etc…*

Implement a Verilog module to create the truth table above; i.e. takes a 4-bit binary value as input, and produces the proper 7-bit LED output code. You can start from code below:

```
module bin_to_leds(
    input [3:0] b_in,
    output reg [6:0] leds
    );

    always @(b_in)
        case(b_in)
            0: leds = 7'b0111111;
                ... fill in the rest here ...
             default: leds = 7'b0000000;
        endcase
endmodule
```

There is another step we need in order to light up the LEDs on the Digilent board. The Digilent board actually turns on the LEDs when the value is 0, not 1 (this is because these pins are connected to the cathodes of the diodes). So, write another module that takes a 7-bit value as input, and produces a 7-bit value as an output that is the inverse; i.e. every bit of the output should be inverted from the input.

```
module invert7(
    input [6:0] a,
    output [6:0] x
    );

                ... fill in the rest here ...

endmodule
```

Now we will instantiate these two modules and connect them together inside another top-level module. We will also create an 8-bit signal *enb_leds* that will turn on only the last 7-segment display.

```
module seven_seg_out(
    input [3:0] b_in,
    output [6:0] inv_leds,
    output [7:0] enb_leds
    );

    wire [6:0] leds;

    assign enb_leds = 8’b11111110;

    bin_to_leds u1 (.b_in(b_in), .leds(leds));
    invert7 u2 (.a(leds), .x(inv_leds));
endmodule
```

Modify the constraint file in your project so that the inputs of seven_seg_out connect to 4 switches on the board, and the *inv_leds[0]* through *inv_leds[6]* outputs connect to the ports labeled “AC” through “AG”, under the “7 segment display” part of the constraint file.

Also connect the *enb_leds[0]* through *enb_leds[7]* outputs to ports labeled “AN[0]” through “AN[7]”; these outputs enable the selected 7-segment display on the board (there are 8 of them, one for each bit of *enb_leds*).