# ECEN2350 Digital Logic – Lab 2

## Fall, 2019

Due November 6th, 2019, 11:59pm

10% of Course Grade (100 points)

Successful completion of this lab will require use of iVerilog, GTKWave, Quartus, and the DE10-Lite board. If you are having problems with any of these design tools, contact your instructor as soon as possible.

Everyone is responsible for turning in their own project. You may work with others to complete the lab.

This lab will require you to code and compile the design using iVerilog and Quartus, simulate the design using both GTKWave and/or textual output, and submit a working project and a lab report. The goal of Lab 2 is to extend your design knowledge to synchronous design techniques, and to create a testbench that simulates this synchronous design.

Upon completion of this lab, you will be familiar with synchronous design coding techniques, creating modulo-10 counters, and moving between binary and binary coded decimal (BCD) representations of data.

**Lab Description:**

Lab2 consists of one design that displays the day of the year and also the corresponding month and date. Detailed requirements are found in a later section of this document.

You should read carefully the remainder of this document before beginning to design.

**Grading (total 100% + optional extra credit)**

1. Successful completion and submission of base design 60%
2. Successful completion and submission of testbench 20%
3. Lab report. 20%
4. Extra credit:
    a. Selectable clock speed 10%
    b. Leap year control 10%

If you are unable to complete the base design, submit what you have for partial credit.

Your lab report should fully explain the final state of your project.

**Note:** The lab report is mandatory. No credit will be given if the lab report is not submitted. Document as much of the lab as you were able to complete. You may not submit a lab report that you did not write yourself.

**Submission guidelines:**

1. You must submit your project via Canvas.
2. Submit all files including the final report in a single .zip file. Make sure the folder names you use do not contain spaces or special characters.
3. Verify that the .zip file you submit contains all files required to rebuild your project. This includes all verilog source and testbench files, and at a minimum the Quartus .qpf and .qsf files.
4. If your submitted project will not build successfully, you will lose credit. Test the contents in your .zip file before submitting for credit
5. Late submission of the lab will result in a 20% reduction in your project grade.

**Guidelines for lab report:**

a) Your lab report should consist of the following sections:
    a. Problem statement
    b. Theory of operation
    c. Block diagram of your design
    d. Hierarchy of source files
    e. Description of testbench operation and output
    f. Summary of project success, what works and what doesn't

**DE10-Lite Definitions:**

Per the DE10-Lite User Manual, the slide switches are defined as DOWN or Logic 0 when the switches are nearest the edge of the DE10-Lite board, and UP or Logic 1 when the switches are pushed toward the center of the DE10-Lite board.

With the DE10-Lite board positioned with the 7 segment displays and slide switches at the bottom of the board:

HEX5 is the leftmost 7 segment display, HEX0 is the rightmost 7 segment display.
SW[9] is the leftmost slide switch, SW[0] is the rightmost slide switch.
LEDR[9] is the leftmost LED, LEDR[0] is the rightmost LED.
KEY[0] is the pushbutton switch closest to the VGA video connector.

**Detailed Requirements for Lab2 (Read carefully, get clarifications early)**

1. A binary coded decimal (BCD) counter will count between **1** and **99** inclusive representing the day of the year, with the output displayed on HEX5 and HEX4 as BCD digits. The most significant digit must be blanked instead of displaying zero.

2. The update frequency for the display will be 1 Hz. Use the 10Mhz clock provided by the DE10-Lite board – this clock is named ADC_CLK_10 – and divide down to a1 Hz display clock. Connect the display clock to LEDR[1] so you can see the LED blink at the 1 Hz rate.

3. The value of KEY[0] and will be toggled and latched each time the pushbutton is pressed. KEY[0] will be used as an active low RESET signal for your design. Use LEDR[0] to display the latched value of KEY[0]. Note: RESET will be active after each new board download, so don't forget to press KEY[0] to start your circuit.

4. Using the value of the day of the year, display the corresponding **BCD** coded date in MMDD format. HEX2 will contain the month (HEX3 will be blank), and HEX1 and HEX0 will display the date (DD). The most significant digit of DD must be blanked instead of displaying zero.

   For example, if the day of the year is displaying as 15, this corresponds to January 15$^{th}$ (MM = (blank)1, DD = 15). If the day of the year is displaying as 90, this corresponds to March 31$^{st}$ (MM = (blank)3, DD = 31).

**Extra Credit**

1. Create a second display clock running at 5 Hz. Using a latched signal based on KEY[1], select between display clock frequencies. The selected display clock should drive LEDR[1]. The default display clock frequency after board programming should be 1 Hz.
2. Using SW[9], modify circuit behavior to count based on a leap year when SW[9] is ON.

**Testbench Requirements**

1. You must create a testbench that demonstrates aspects of the design. At a minimum:

    a. Demonstrate the day of the year counter counting through the entire 1 to 99 range.
    b. Show the effect of reset.
    c. Show the output of the clock divider to demonstrate successful clock division.
    d. Show update of the day of the year hex displays.
    e. If you included the extra credit circuits, capture the behavior.
2. Your simulation output can be .vcd file(s), text file(s), or a combination of both.

**Background Information**

1. January contains 31 days.
2. February contains 28 days, except for a leap year, when February contains 29 days.
3. March contains 31 days.
4. Binary coded decimal means that the only allowable digits are 0 – 9.

**HINTS:**

1. Latching a pushbutton is this easy:

```
always @ (negedge pushbutton_in)
    latch_out <=  ~latch_out;
```

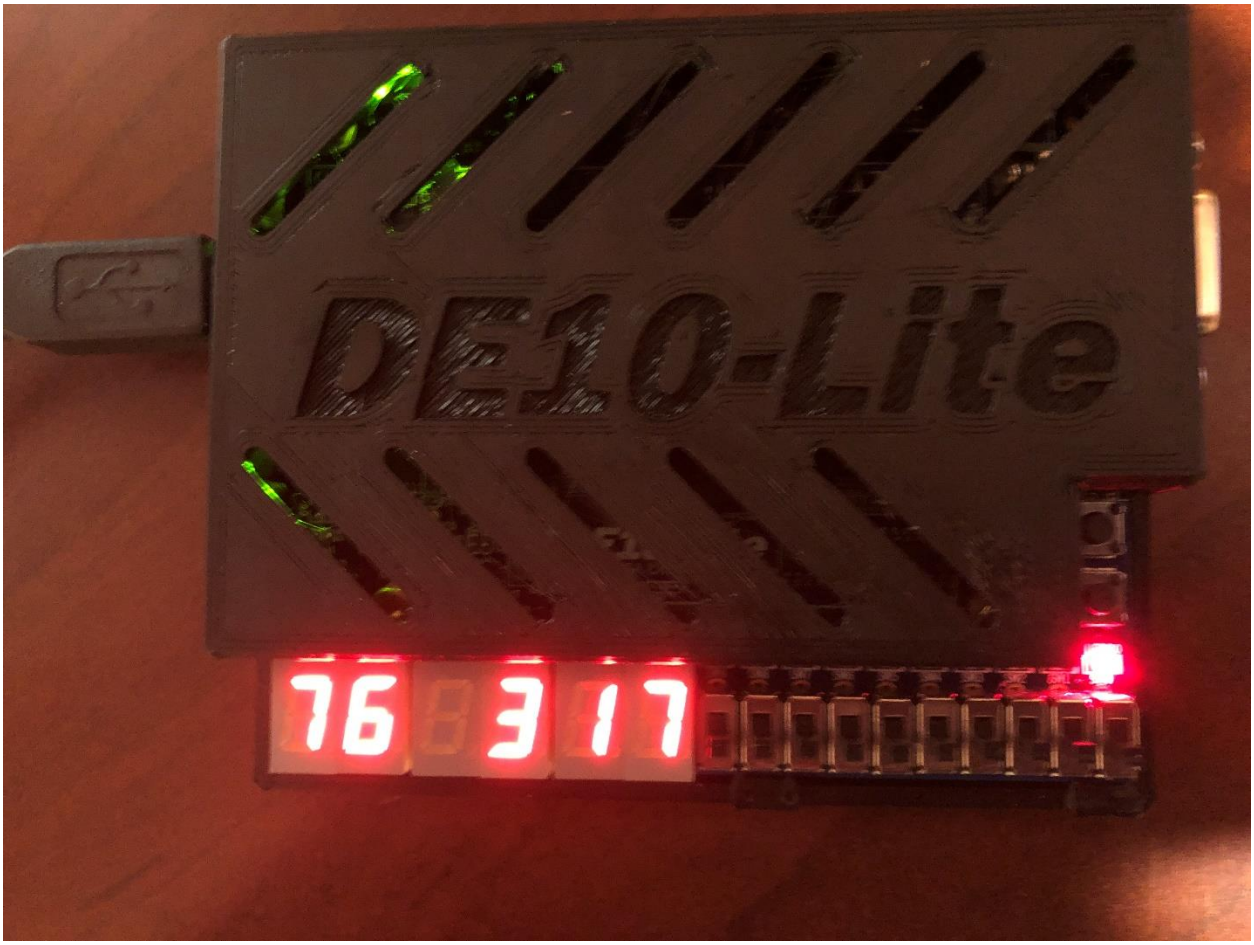
2. You can likely reuse your seven segment decoder from Lab1.
3. Create your BCD counter using behavioral code, not by copying the diagram in the textbook. Think carefully about counter behavior before beginning to code.
4. To convert from 2 BCD digits to binary, you can use this:

```
wire [6:0] binary_value;
assign binary_value = (BCD_value[7:4] * 10) + ({3'b0, BCD_value[3:0]});
```

5. Since the month value only ranges from 1 to 4, displaying that value is simple. However, since the day of the month ranges from 1 to 31, displaying the value as BCD digits may not be straightforward.
6. You may want to modify your clock divider to a faster frequency in order to speed up simulation time and also limit that amount of information in your text and .vcd files. Instead of a 1 Hz display clock, simulate using a 10 Khz clock instead.

And the most important hint of all…..

It is very tricky to convert from binary back to BCD. It might be simpler to do the conversion in a case statement that by using an algorithm. In my design, I created a special hex display module that took in the binary value of day of the month, and wrote out the values to the two day of the month HEX displays. This eliminated any need to convert binary to BCD




76th day of the year, date is March 17th.