ECEN 449 Lab 1: Using the Vivado

1/30/2020

Jason Gilman

**Introduction:**

In this lab, we were tasked with getting familiar with the Zybo-Z7-10 FPGA board. By implementing the use of LEDs, Switches, and Buttons in the 3 parts of the lab, we were able to see how to use each of these components on the FPGA board. In the lab, a simple switch, up/down counter, and jackpot programs were created.

**Procedure:**

After getting Vivado and the FPGA set up, the first program to be implemented was a simple switch. The code involved linking the switches with the corresponding LEDs, such that when one switch was on, the LED was on. After loading the program onto the FPGA, testing occurred to ensure the logic was correct.

The next step was to implement a simple up/down reset counter, such that the LEDs would display a count from 0 to 15. The input would be from buttons, where one is linked with count-up, another with count-down, and the last with reset, which sets the count back to 0. The module required a cock divider program to slow the detection of pushing a button, because the FPGA clock speed was too fast for a human to be able to make sense of. I slowed the FPGA to about 1Hz, which meant that when the up or down button was pushed, it too about 1 or 2 seconds for the LEDs to update. This was because the updating of the count state was handled on the posedge of the clock. The clock divider module was instantiated within the counter module to take the input of Clock and output a slower clock for the edge detection.

The final program was a jackpot game. This is where the LEDs light up one at a time - in order. The goal of the game is to turn the corresponding switch when the LED is lit up. When the game is won, the LEDs transition to a win state, where all four LEDs flash. The reset input must be pushed to resume the game after being in the win state. This program was implemented by putting each stage of the LEDs into separate states. The combinational and sequential logic was separated to where on the clock edge, the next state becomes the current state. Another clock divider was needed to slow the FPGA clock down. This time the clock was slowed down to about 10Hz, so it makes the game slightly challenging but still makes it possible to win. For the jackpot game, the four LEDs and switches were used, and the button was used for the reset.

**Result/Output:**

The output of the counter was based on the input of the buttons from the FPGA. If the up button is pushed the count increments by one, and is then simultaneously displayed on the LEDs. If the down button is pushed, the count is decremented by one and if the reset button is pushed, the count is set to zero. At all times, the LEDs display the current count.

The output of the jackpot is based on the state that the code is in. There were five states in total. One for each of the LEDs and one win state. If the correct switch is set to on when the LED is on, then the program enters the win state. If you have the incorrect switch on, the game pauses, and waits till the switch is down to resume. You can only enter the winstate if a switch and LED are set to on at the same clock edge.

**Conclusion:**

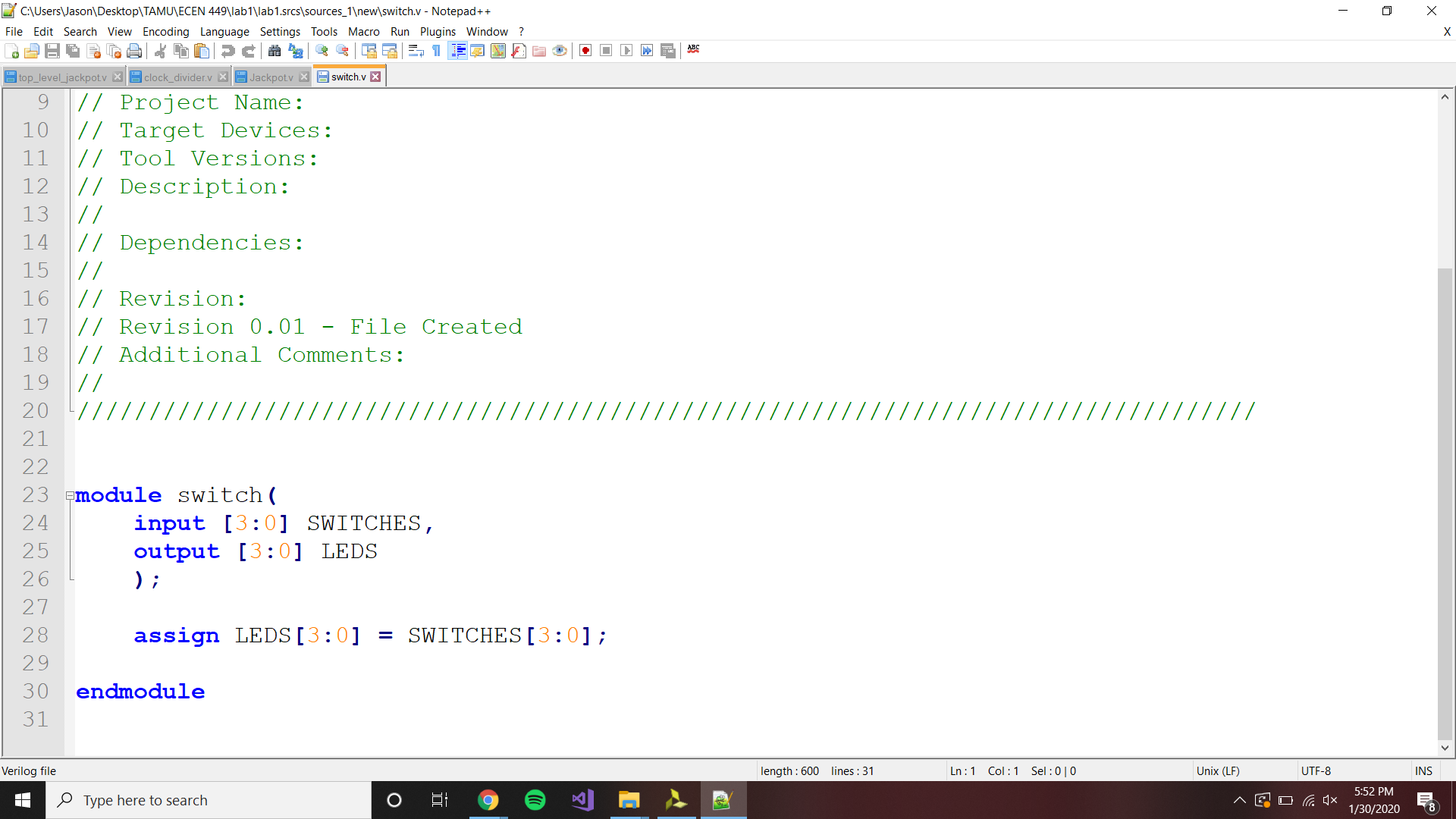
By utilizing state diagrams, the counter and jackpot game were implemented. The counter program utilized more sequential logic to achieve the desired state, because the count was being updated. The jackpot game was a classic example of a state diagram implemented program. There were only 5 states that needed to be implemented to make the full game. Organizing the jackpot game by creating a state diagram aided my thinking to figure out what state needs to follow the current state, and when to enter the win state.

**Post Lab Questions:**

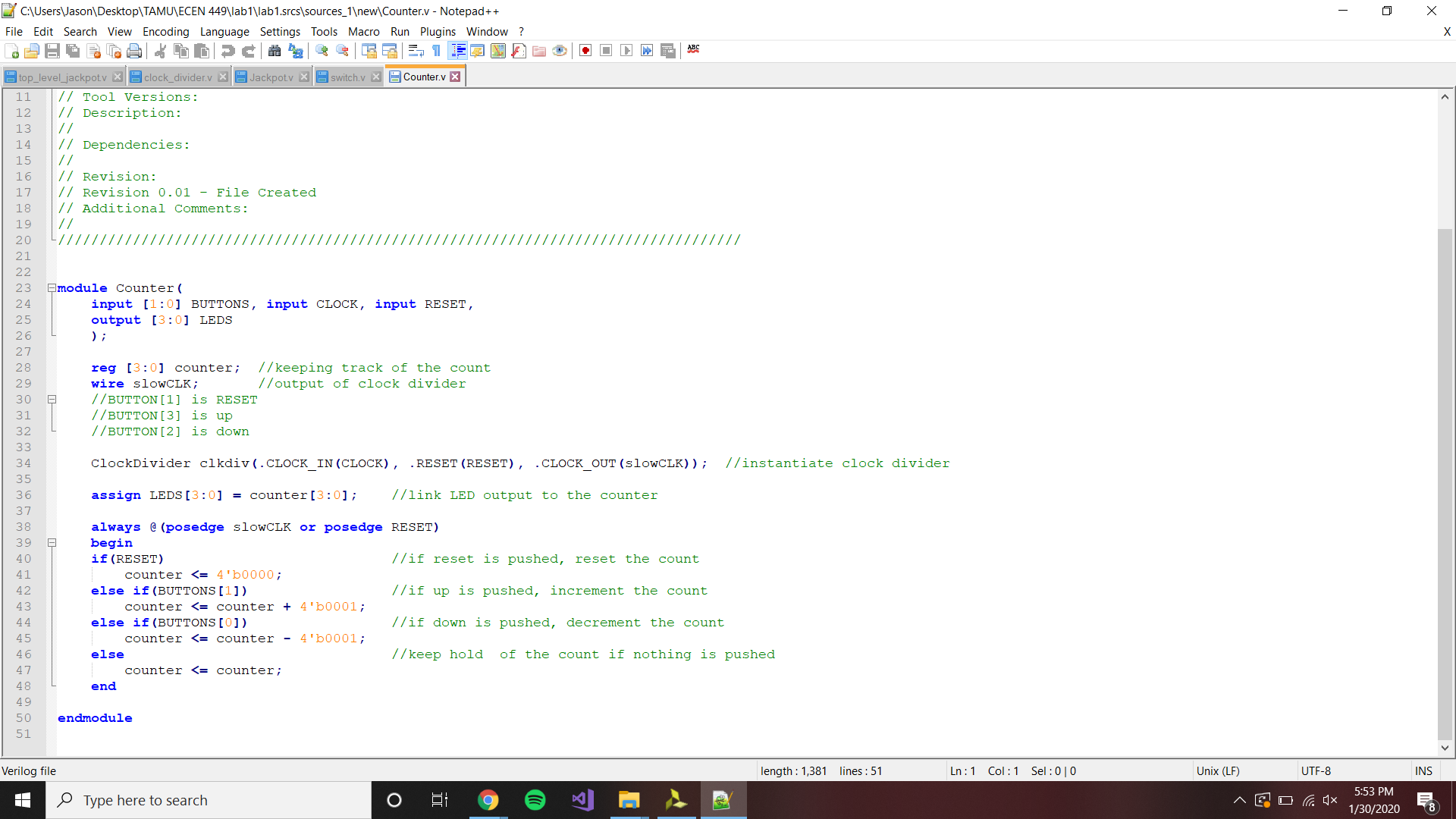
1. The four buttons are wired to pins K18 P16 K19 Y16 respectively. The buttons are pull down, because you need to specify specifically that they need to be pull up. The default setting is pull down.
2. An edge detection circuit outputs a signal whenever a certain edge is detected. This could have been used in the jackpot game to detect which edge the switch and or the LED was on. Because to win the game the switcha and corresponding LED needed to be on at the same edge, a sort of edge detection was needed to make the program work properly.

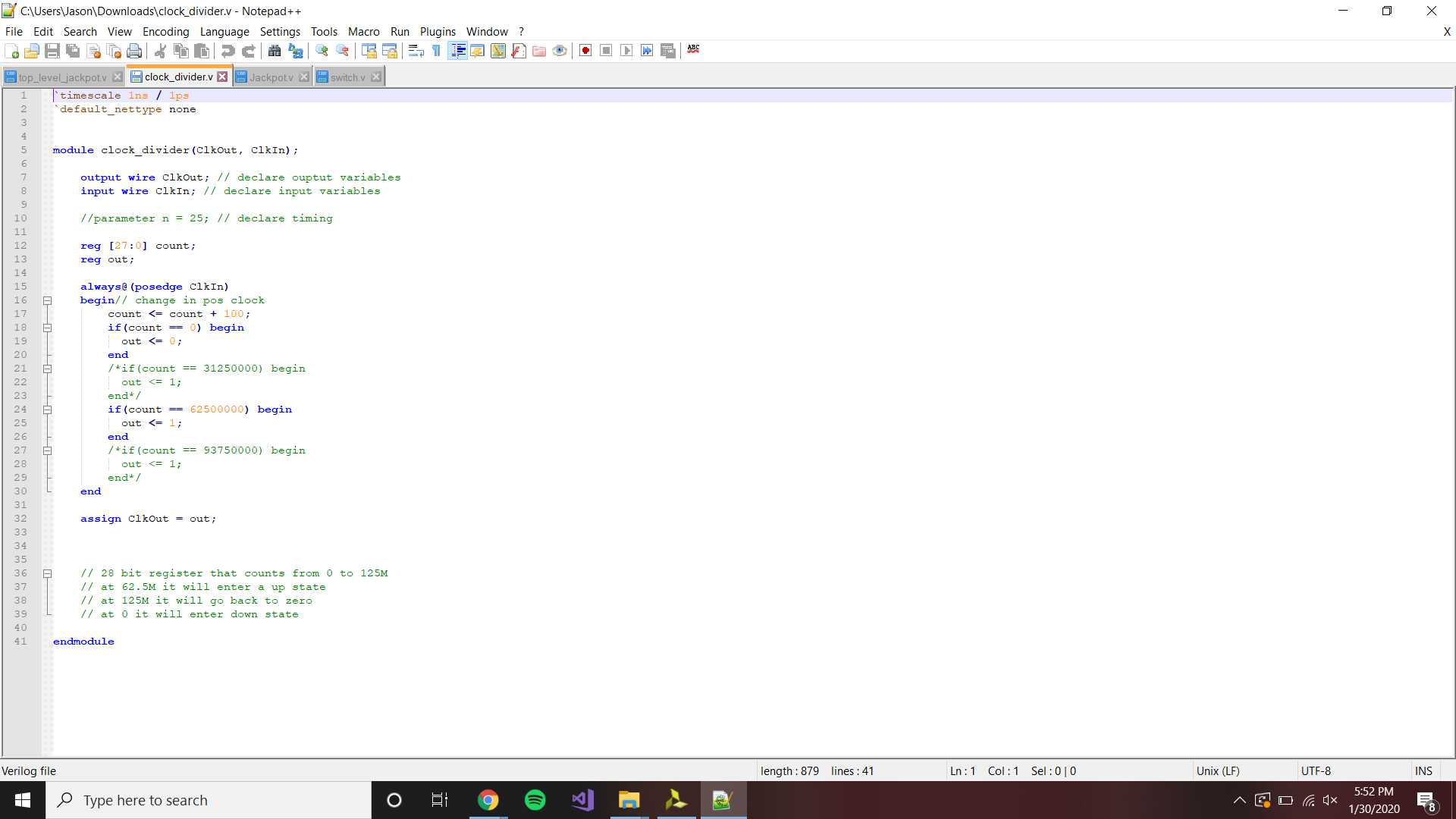
**Appendix:**

Switch:



Counter:





Jackpot:

