Abstract:

In this lab we worked with two 7-segment displays to display values from 0-9 and A-F. The lab had three parts. Part 1 and 2 had us implement the values stated above on one of the 7 segment displays by using switches to cycle through them in two different methods, part 1 was to implement it using logic equations, and part 2 was to implement the same thing using binary values. Lastly, for part 3 we had to make a counter that counted up to a specific double digit value that utilized both displays and once it reached that value it would reset back to zero.

Introduction:

The primary objective of this lab was to design and implement a 7-segment LED decoder using VHDL for displaying hexadecimal values based on binary inputs (SW0 to SW3). The 7-segment LED decoder works based on a common-cathode configuration, where the cathode is grounded and segments light up upon receiving a positive voltage signal. Each segment is individually controlled through GPIO pins to display characters by lighting combinations of the 7 segments. To display the numbers from 0 to F, we derived the boolean logic expressions by constructing K-maps from the truth table. The experimental setup involved us in programming the boolean expression in the template VHDL files provided in the eclass and then verifying the code via simulation and then implementing it on the Zybo Z7 FPGA board.

Design Section:

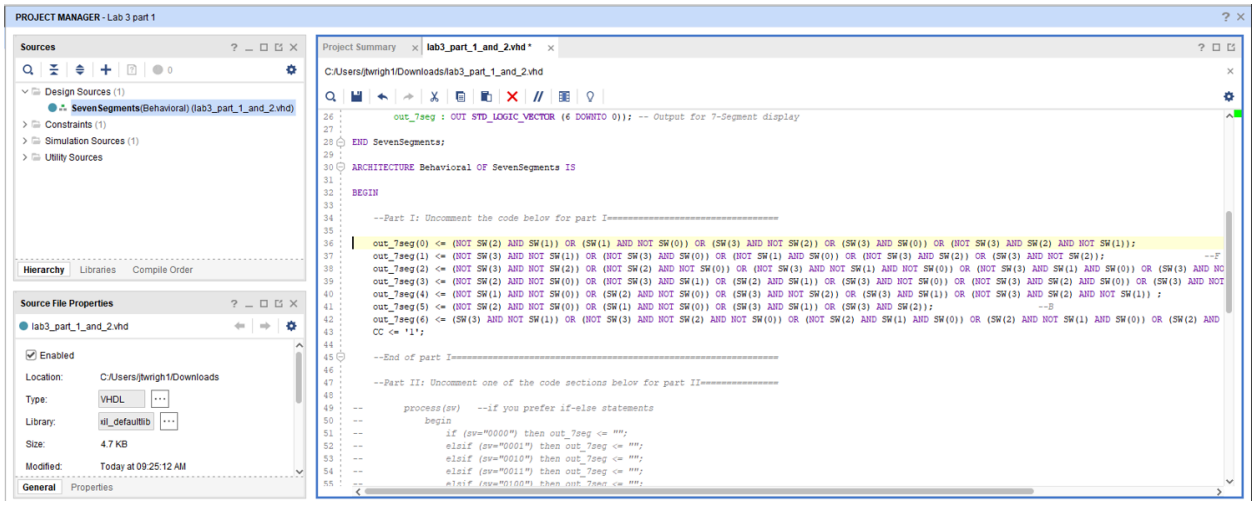

Figure 1 : Code for Part 1

In the lab, we mostly had to program in VHDL. The template VHDL code that was provided to us in the eclass defined the logic for a 7-segment display to display hexadecimal values from 0-F with a 4-bit binary input (SW0 to SW3). The segments of the display are controlled by a defined output (out_7seg(0) to out_7seg(6)) to tell the display which segment of the display to light up. The first part of the lab was to implement the 7-segment display using boolean expressions that were derived by constructing K-maps from the truth table. Those boolean expressions were programmed in the template code using logical operators AND, OR and NOT to implement the

logic for outputting the required signals to the 7-segment display for different input combinations. The code consisted of a selection configuration to select which 7-segment display to route the signals (CC<=1 or 0) used to display the characters. The code for part 1 is shown in figure 1. Figure 2 is one of the circuit designs.

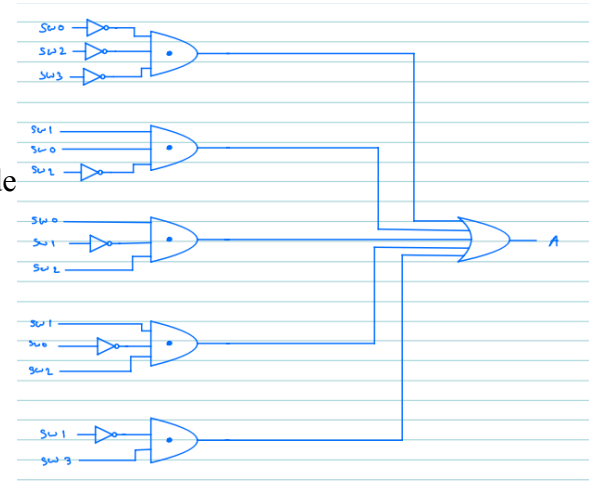

Figure 2: Circuit for segment A

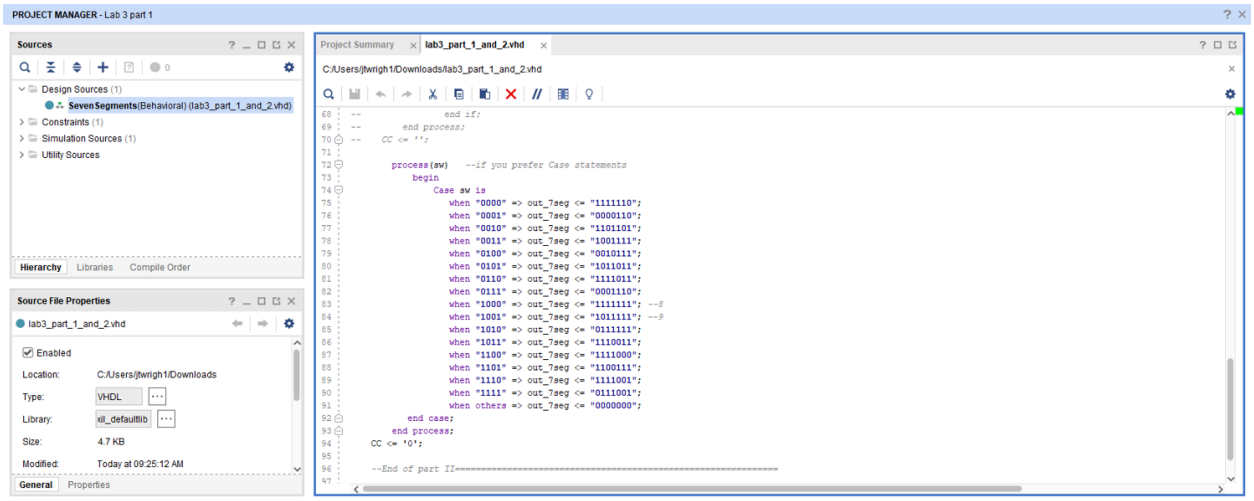

Figure 3: Code for part 2

The second part of the lab was to implement the same thing, without using the boolean expressions. The template that was provided to us used a case statement to control the 7-segment display based on the 4-bit binary input state of the switches. Like in part 1, the code uses a defined output (out_7seg(0) to out_7seg(6)) to assign a 7-bit value to light up the corresponding segment on the display. The code consists of a selection configuration like in part

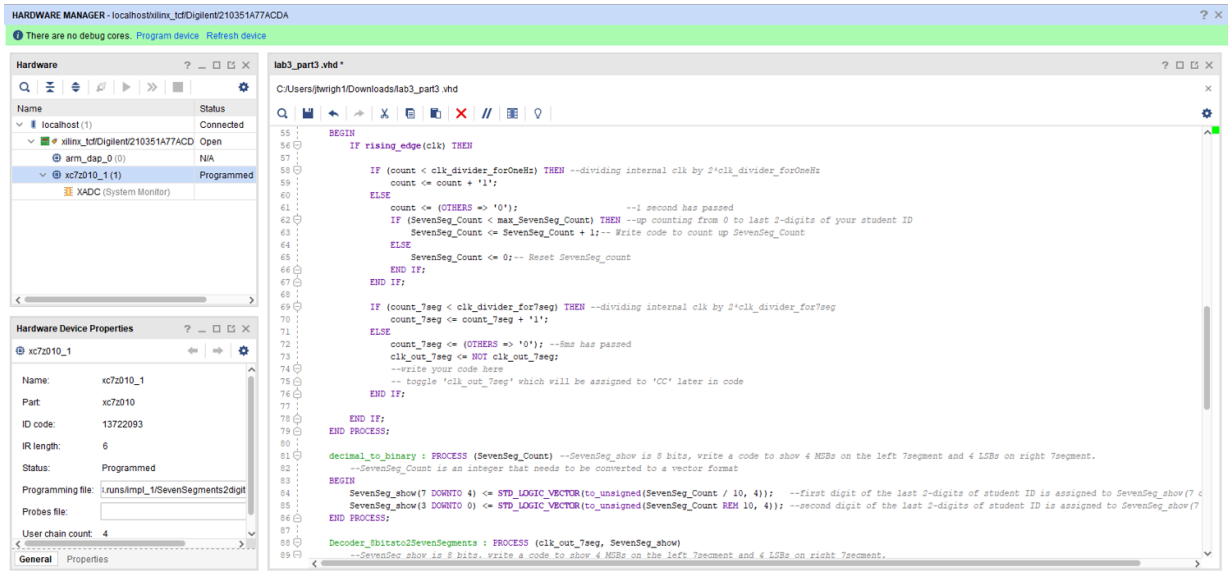

Figure 4.1: Code for part 3

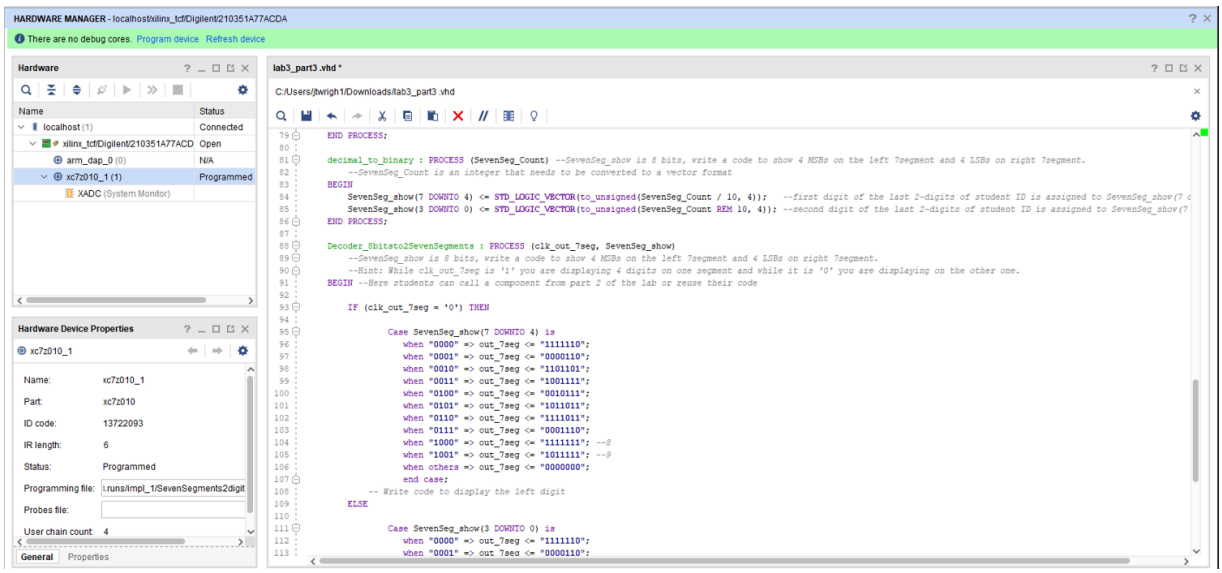

Figure 4.2: Continuation of code for part 3

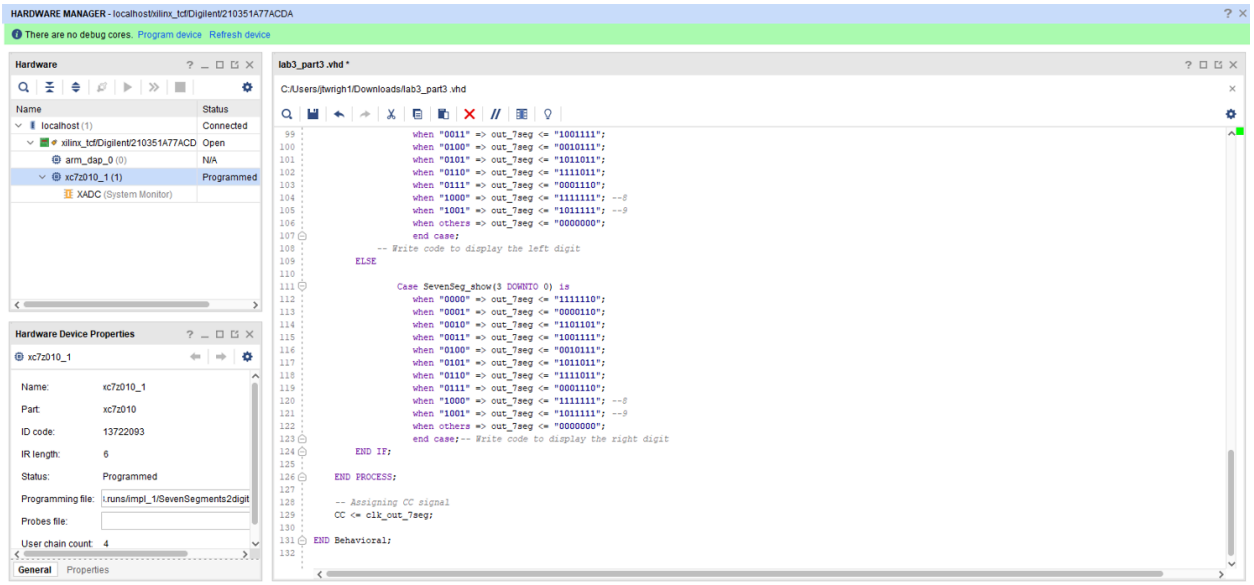

Figure 4.3: Continuation of code for part 3

Finally in part 3 we had to create a counter that counts up to a specified digit. In **Figure 4.1** the first part of the code counts up at each rising edge. If the one's place value is less than 9 then it adds one to the one's place. If the one's place value is = 9 then it adds one to the ten's place value and sets the one's place value to 0. This keeps going on until it hits the specified number where in the bottom half of the code it checks if the count is equal to that number. If it's greater than that number it sets both the one's place value and the tens place value back to zero and it starts counting back up from 0. In **Figure 4.2** it states that when sending the information to the 7 - segment display we send it as one package, thus the values 0-3 are for one display, and 4-7 are for the other display. Finally in **Figure 4.3** this is where the values are being sent to the displays. The board is not powerful enough to display values on both displays at the same time, therefore we use 'clk_out_7seg' to flip between writing to each display. If the value for the clk_out_7seg = 0 then it writes to the first display which is the one place value. Inside the if statement takes the last 4 digits of the 8 digit hexadecimal output and uses that to display the wanted value. If the value is 1 then it uses the else statement to write to the second display which is ten's place value. The if statement takes the first 4 digits of the 8 digit hexadecimal output and uses that to display the wanted value.

Experimental Procedure and Equipment:

Equipment: For this lab we used the Zybo Z7 FPGA board along with a common cathode 7-segment display connected to it. We used the 4 switches on the FPGA board for the 4-bit binary input. We used the lab computers with Xilinx Vivado installed, used to program the FPGA board.

Procedure: We first complete the programming in VHDL in the templates provided in the eclass. After that we simulate the design and check if all the outputs match with the inputs with the truth

table. When the outputs matched with the truth table correctly, we proceed to run implementation and generate a bitstream. After that we turn on the FPGA board, connect it with the computer and program the board. The procedure is the same for all the parts in the lab.

Results:

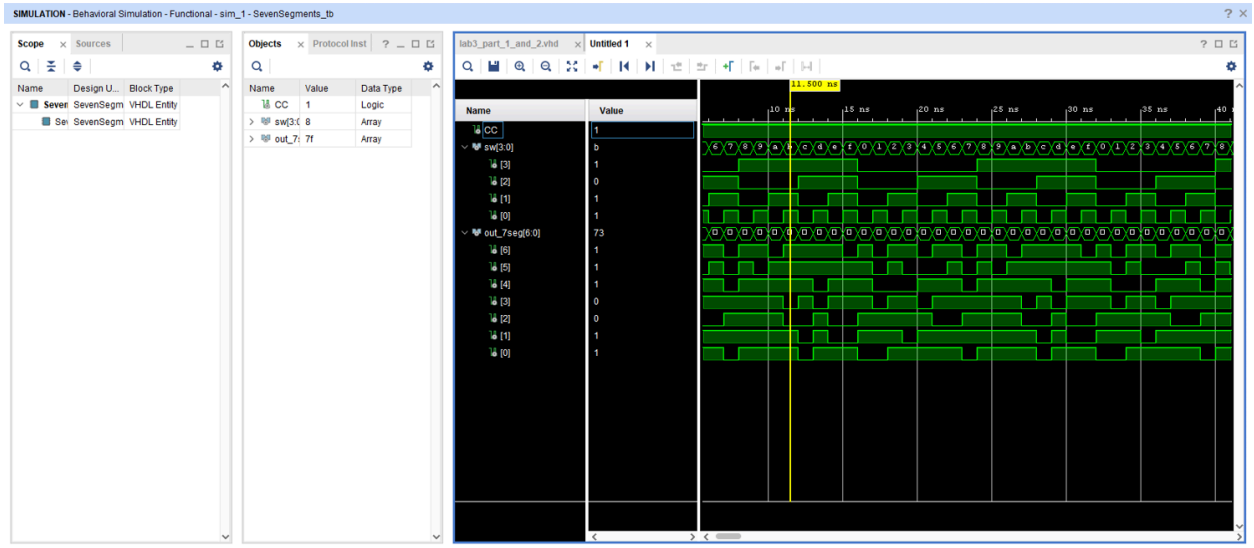

Figure 5: Simulation for part 1

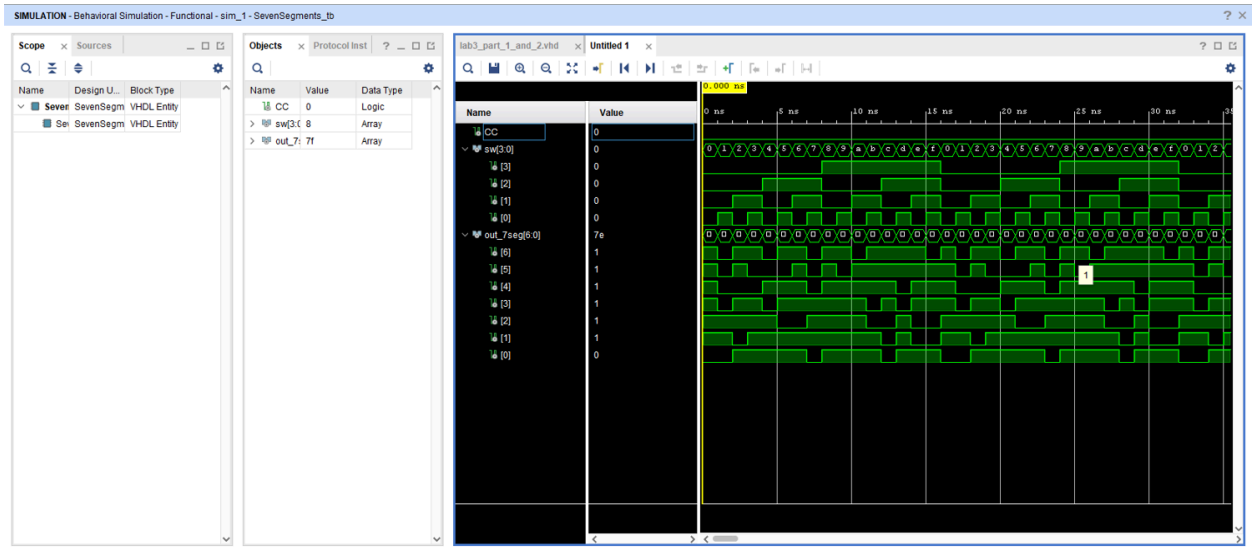

Figure 6: Simulation for part 2

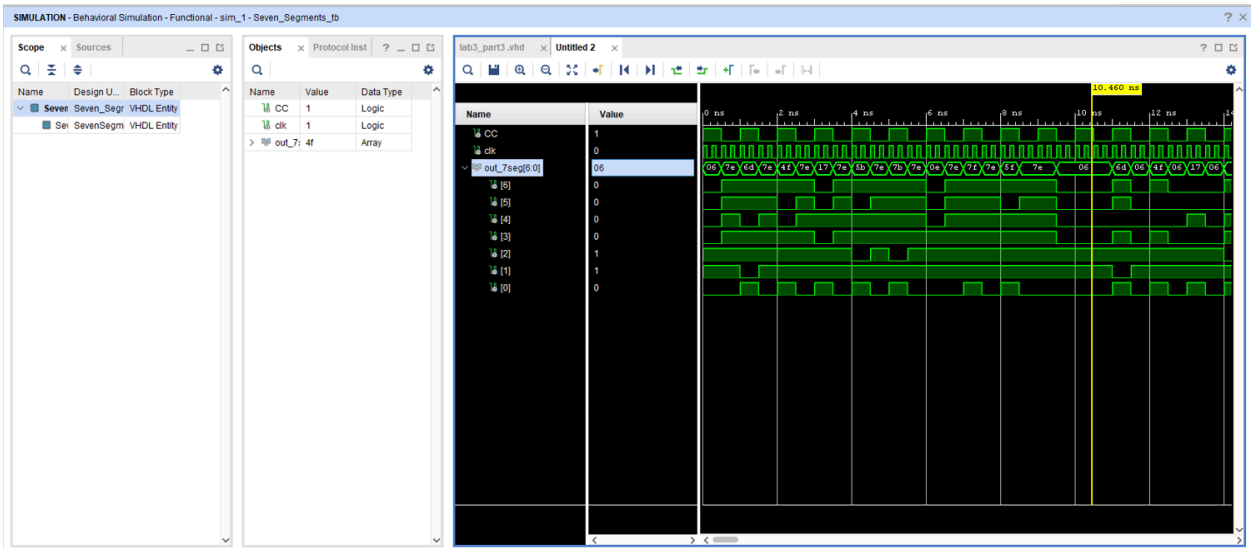

Figure 7: Simulation for part 3

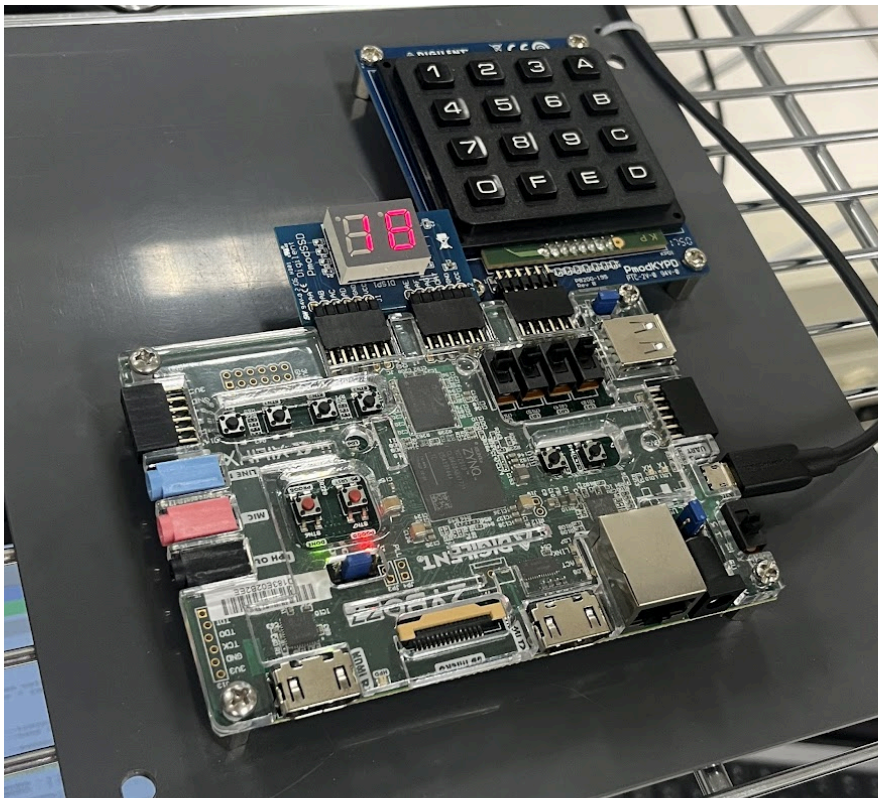

Figure 8: Zybo Z7 FPGA with 7-segment display showing 18 for part 3

Discussion:

This lab was split into 3 parts. Where in part 1 and 2 our goal was to display all hexadecimal values on the 7 segment display provided. In part 3 the goal was to create a counter that counted up to a specified value then reset to count up to that number again. The first 2 parts showed how there are multiple different ways to code and implement logic. Even though both pieces of code did the same thing, part 2's code was a lot

cleaner and easier to understand. A major observation we made was that despite the limitation of the 7 segment display, we were still able to create a 2 digit counter. The board we used could only write to one of the 7 segment displays at once. To get around this issue we cycled through turning each display on and off quickly enough so that the human eye couldn't tell. This is a super clever way of getting around the hardware limitations

Conclusion:

In conclusion we were able to come up with the correct code and implement it to create the graphs and pictures above. The 7 segment displays worked indeed, but coding them to display the right thing was a bit tricky. Coding the counter was moderately difficult to create, but once the code was made it was easy to implement and simulate.

References:

[1] Department of Electrical and Computer Engineering, *General Lab Report Style Guide,* University of Alberta, Fall 2017. [Accessed: November 19, 2024]

[2] Department of Electrical and Computer Engineering, *ECE_210_LAB_3_revised,* University of Alberta, Fall 2014. [Accessed: November 19, 2024]

[3]