**CPE166 Advanced Logic Design**

**Lab 2 Report**

**Student Name: Andy Cha**

**Date: 3-06-2020**

**California State University, Sacramento**

**TABLE OF CONTENTS**

**Section Page**

Title Page …………………………………………………………………................................ 1

Table Of Contents ………………………………………………………………….................... 2

1. Introduction …………………………………………………………………......................... 3

2. Part 1: 8-bit Carry Select Adder Design ……………………………………………………. 4

Part 1.1: 1 Half Adder Source Code, Testbench & Simulation Waveform………………………. 4

Part 1.2: Full Adder Source Code, Testbench & Simulation Waveform………………………... 5

Part 1.3: 8-Bit Carry Select Adder Code, Testbench & Simulation Waveform………………….. 6

Part 1.4: Multiplexer(MUX) Code, Testbench & Simulation Waveform………………………….. 7

Part 1.5: Multiplexer(MUXB) Code, Testbench & Simulation Waveform……………………….. 8

Part 1.6: 8-bit Carry Select Adder Code, Testbench & Simulation waveform……………………. 9

Part 1: 8-bit Carry Select Adder Result Discussion………………………………………………. 11

3. Part 2: 4 By 4 Binary Sequential Multiplier Design…………………………………………. 12

Part 2.1: Design Purpose………………………………………………………………………... 12

Part 2.2: Design Code………………………………………………………………………….... 13-17

Part 2.2: Design Code & Result Discussion…………………………………………………….. 17

4. Part 3: Character Displays on 8-Digit Multiplexed Seven Segment Displays……………...… 18

Part 3.1: Design Purpose……………………...……………………….……………………….... 19

Part 3.2: Design Code…………………………………………………………………………….. 20

Part 3.2: Design Code & Result Discussion………………………………………………………. 20

5. Conclusion……………………………………………………………………………………….. 21

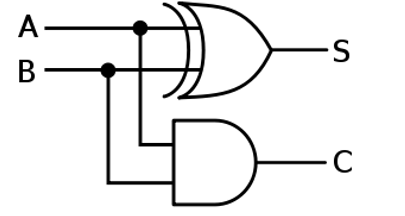
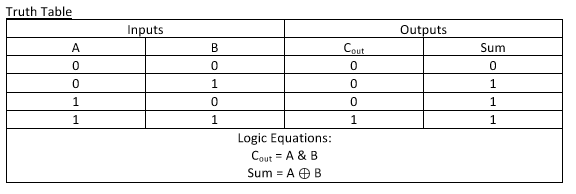
**Introduction**

In this lab we will work with Carry select adder, sequential multiplier, and multiplexed seven segment displays. The purpose of this lab is to give us practice in hierarchical design and design strategy using Verilog. We must be familiar with how carry select adder works, sequential shift/add multiplication algorithm and multiplexed seven segment display. Once we understand how those 3 work, we can simulate the design with waveforms and determine if it is working. Then we will apply our Verilog code into the FPGA setting up the pins allocated to each input and output.

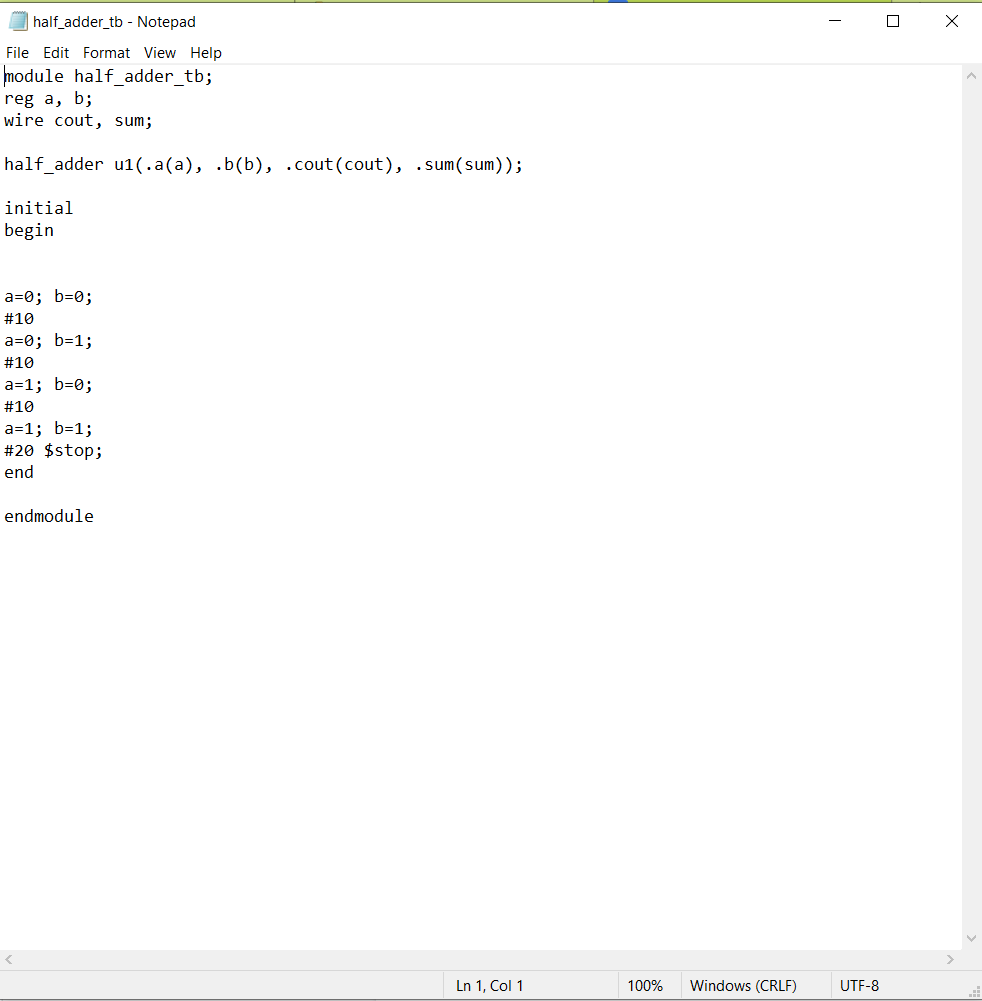
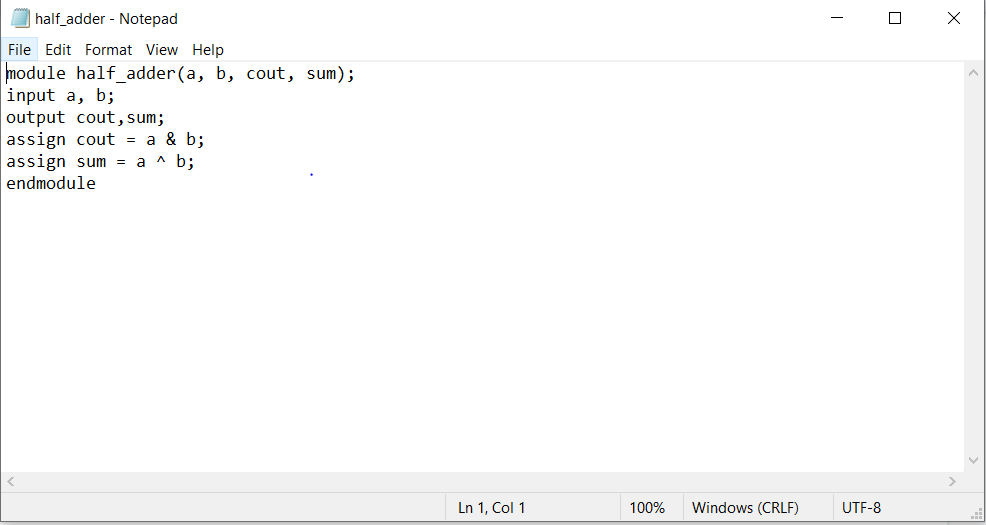
**Part 1: 8-bit Carry Select Adder Design**

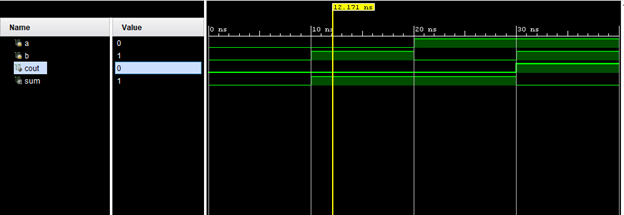
**Part 1.1**

The first part of this lab we will need to create a half adder first. The truth table is given to us along with the design. From this we can create our Verilog code along with the testbench.



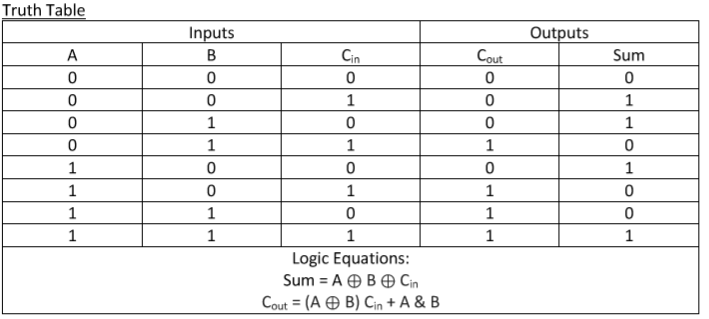
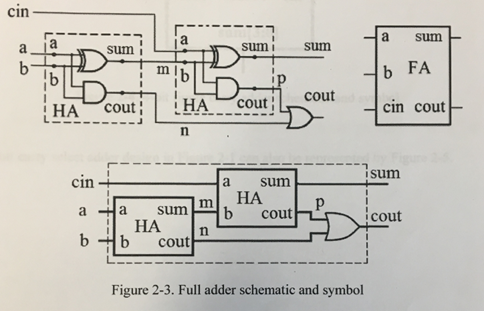
Source Code:



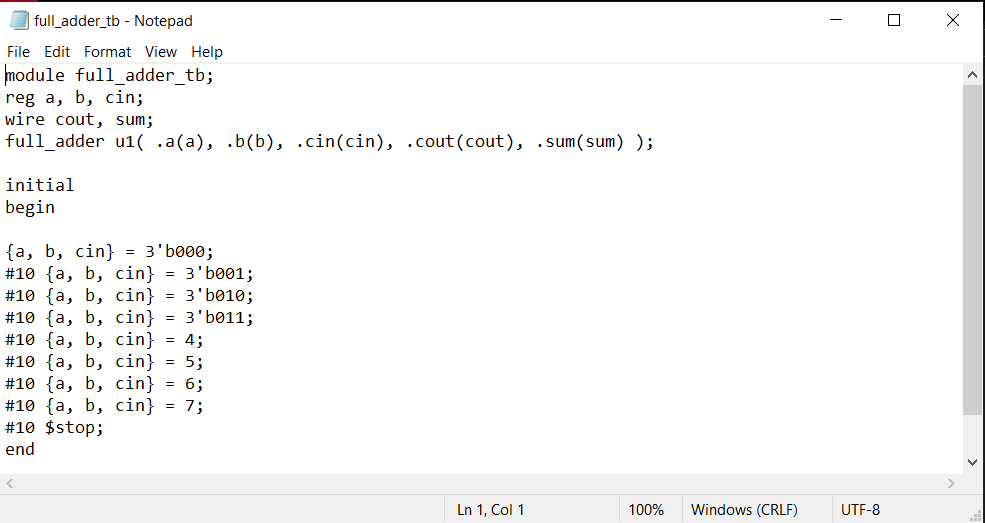
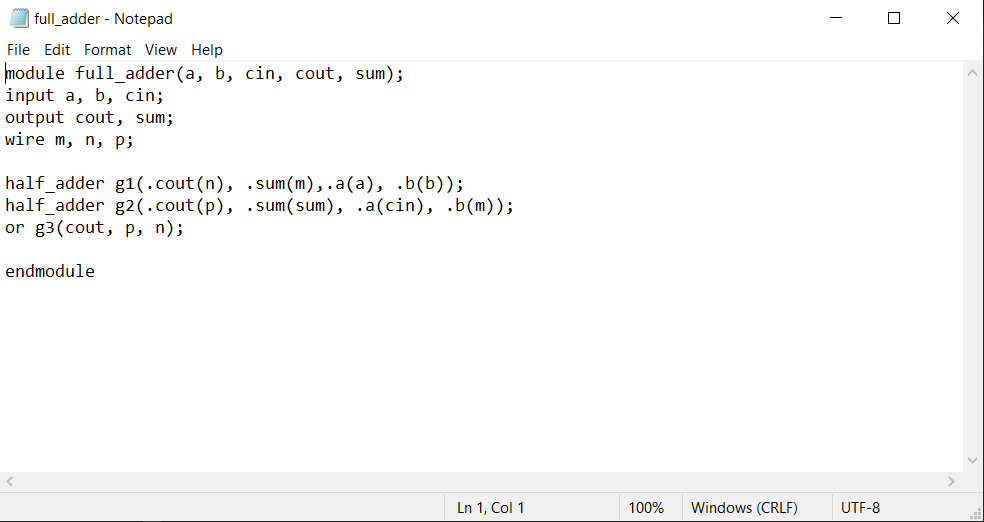


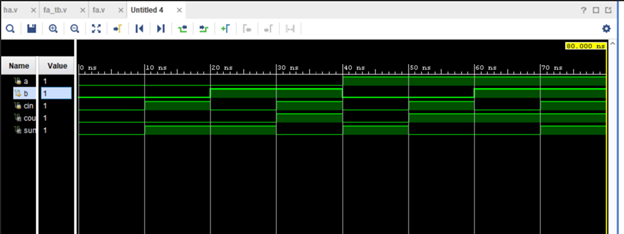
**Part 1.2**

Once we have created the half adder we can use it to create a full adder. A full adder consists of 2 half adders and one OR gate.



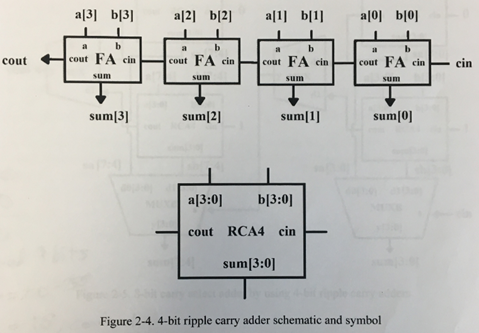
Source Code:



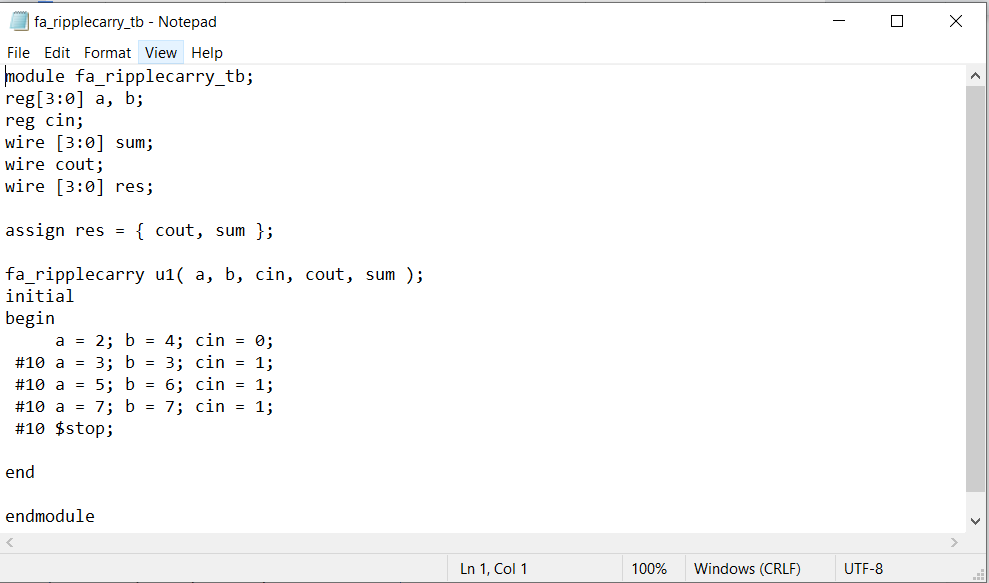
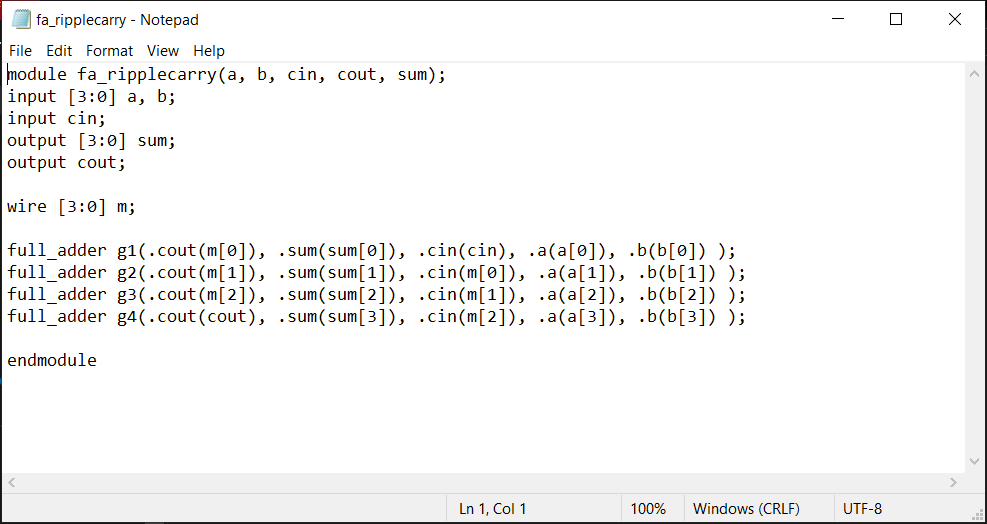


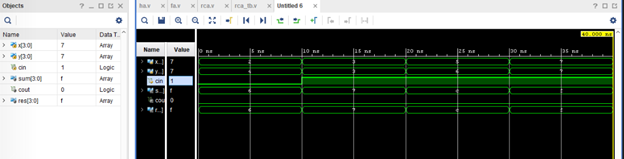
**Part 1.3**

Once we have the full adders completed we have to create a 4-bit ripple carry adder(RCA4). We connect 4 full adders together and move the cout to the next full adder’s cin. We also have to change inputs to [3:0] a, b and create a name for the wires in the middle which will be [3:0] m.

****

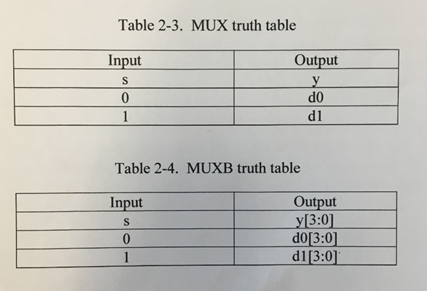
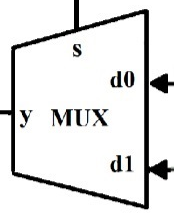
Source Code:



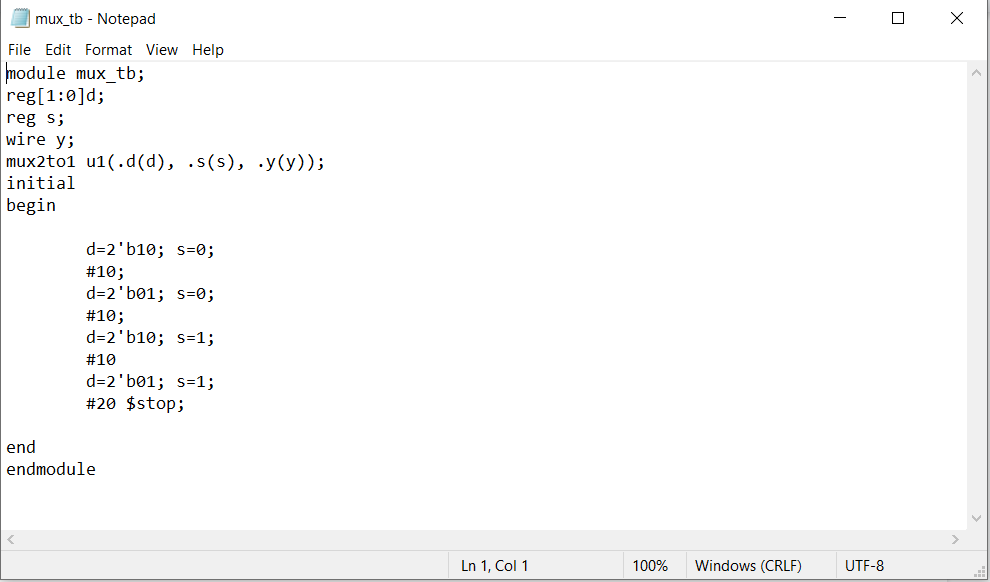
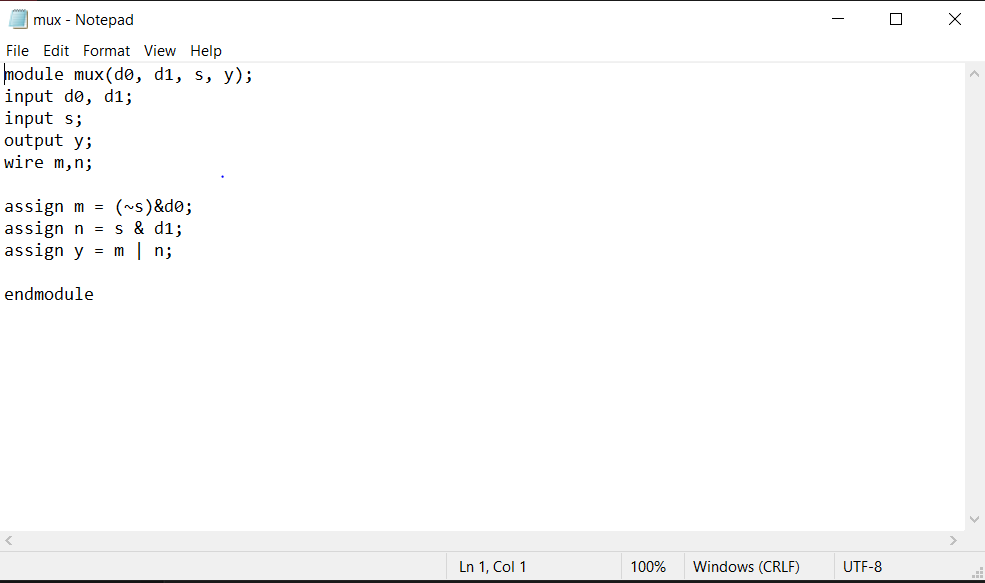
****

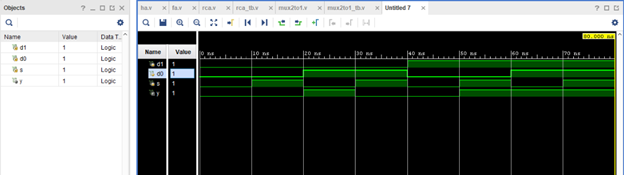
**Part 1.4**

Here we will use the multiplexer that was created in Lab 1.



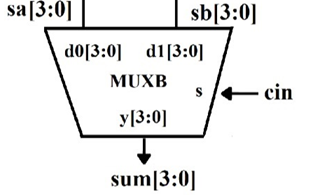
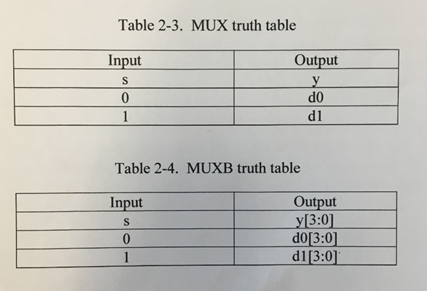
Source Code:



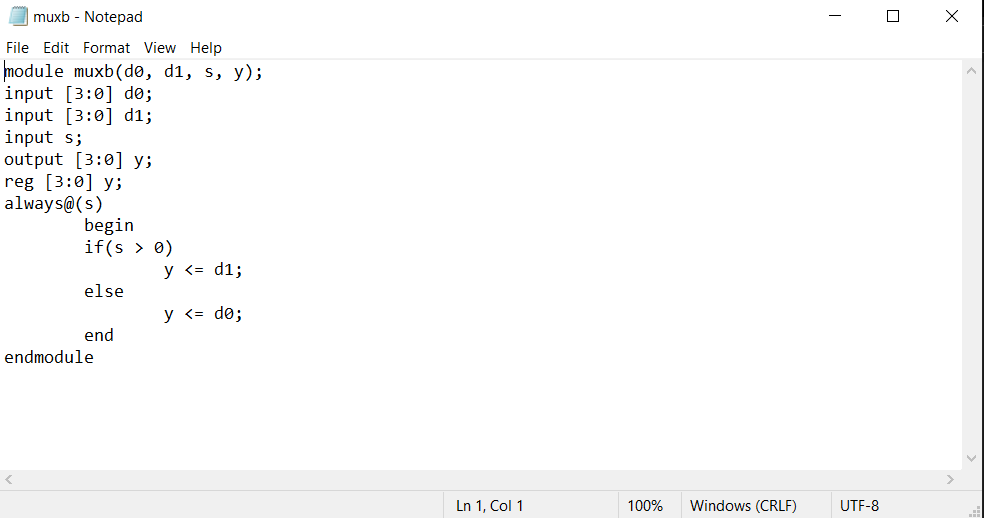


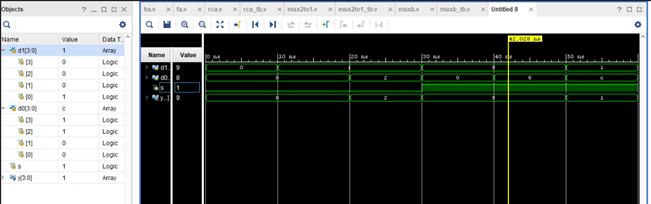
**Part 1.5**

Here We will use the multiplexer that was created from Lab 1. We have to modify it to take 4 bits instead of one bit. We can use the same Testbench as the mux but we need to change the instance name to “muxb” and change the inputs to 4 bits. Then we also need to include a cin.



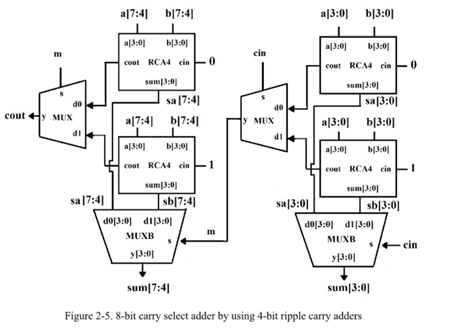
Source Code:



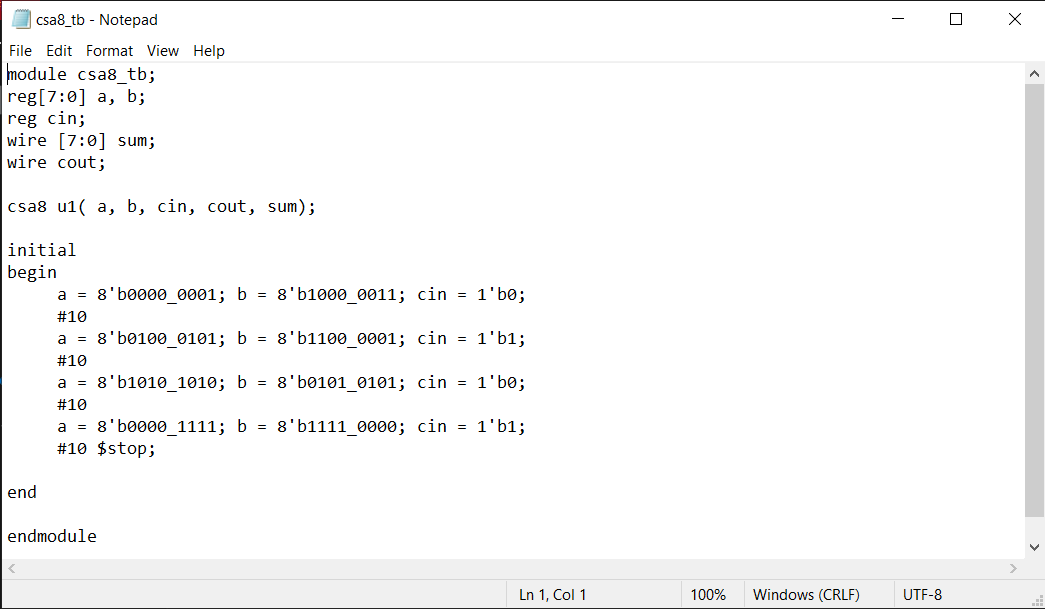
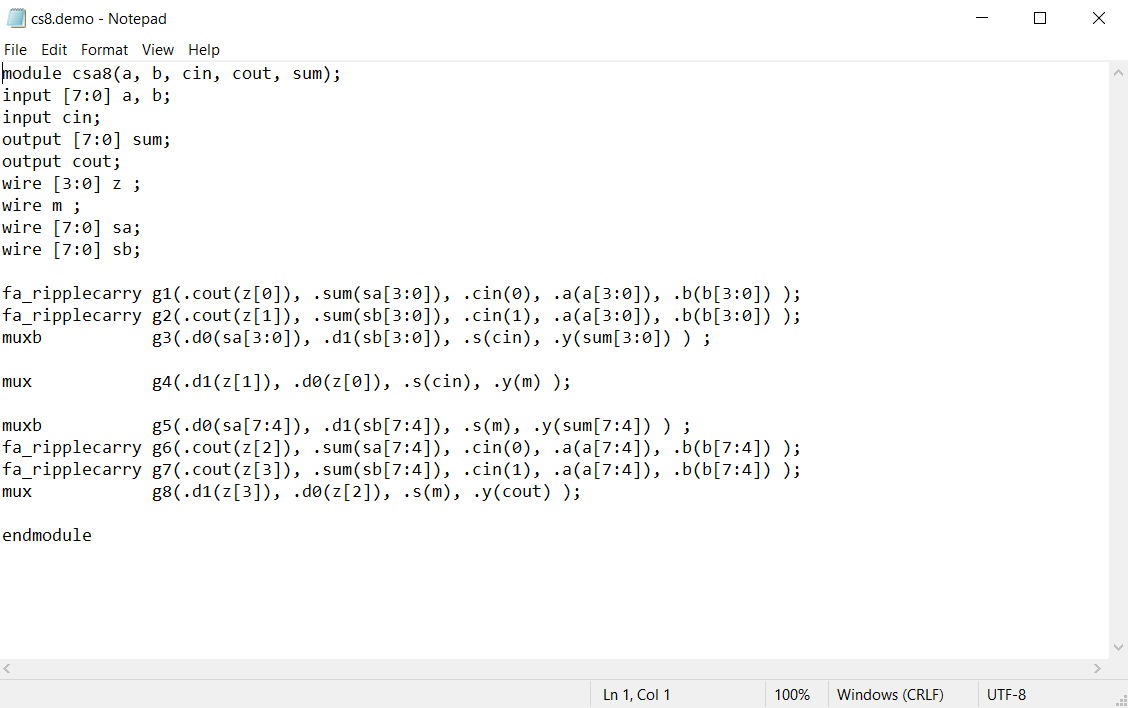


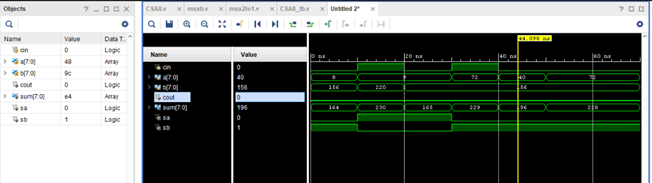
**Part 1.6**

Now we have all the components we need, we can design the carry select adder(CSA8) by connecting 4 RCA4, 2 MUXB, and 2 MUX. In Verilog we can make 8 instances and connect each one with the correct wires.



Source Code:





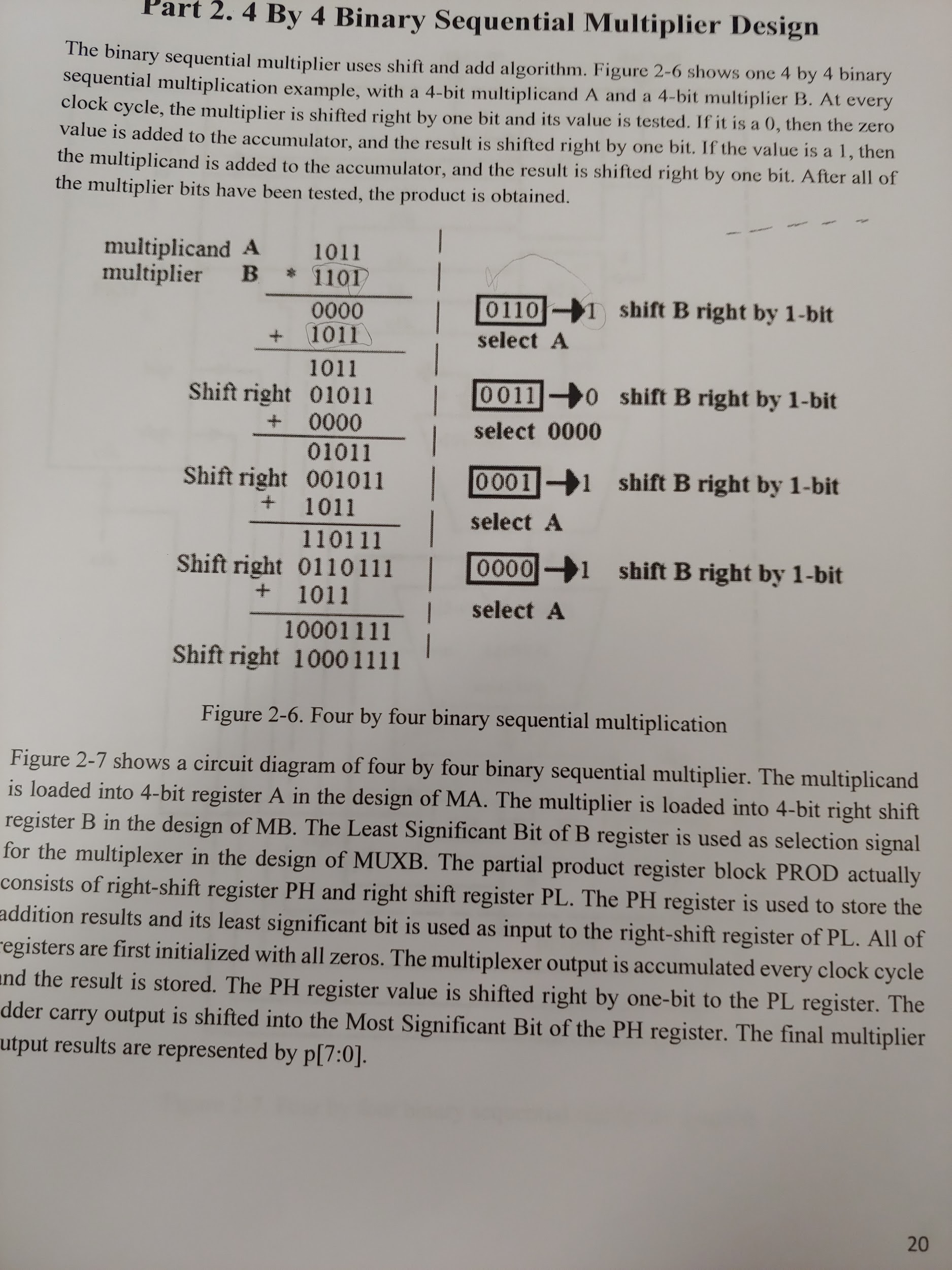
Result Discussion:

The result of this part was that the CSA8 adds two 8-bit numbers properly. It took a lot of time designing all the Verilog files and creating testbenches for each one to determine that there were no issues with it. It was better to check each piece to see if it was right than to test the final product and guess where the error was out of all the files. Some issues I came across were creating the CSA8. Putting all the files together took some time and had to overlook the diagram over and over. The goal of this part was how to use combinational logic and putting it together to create an 8 bit carry adder.

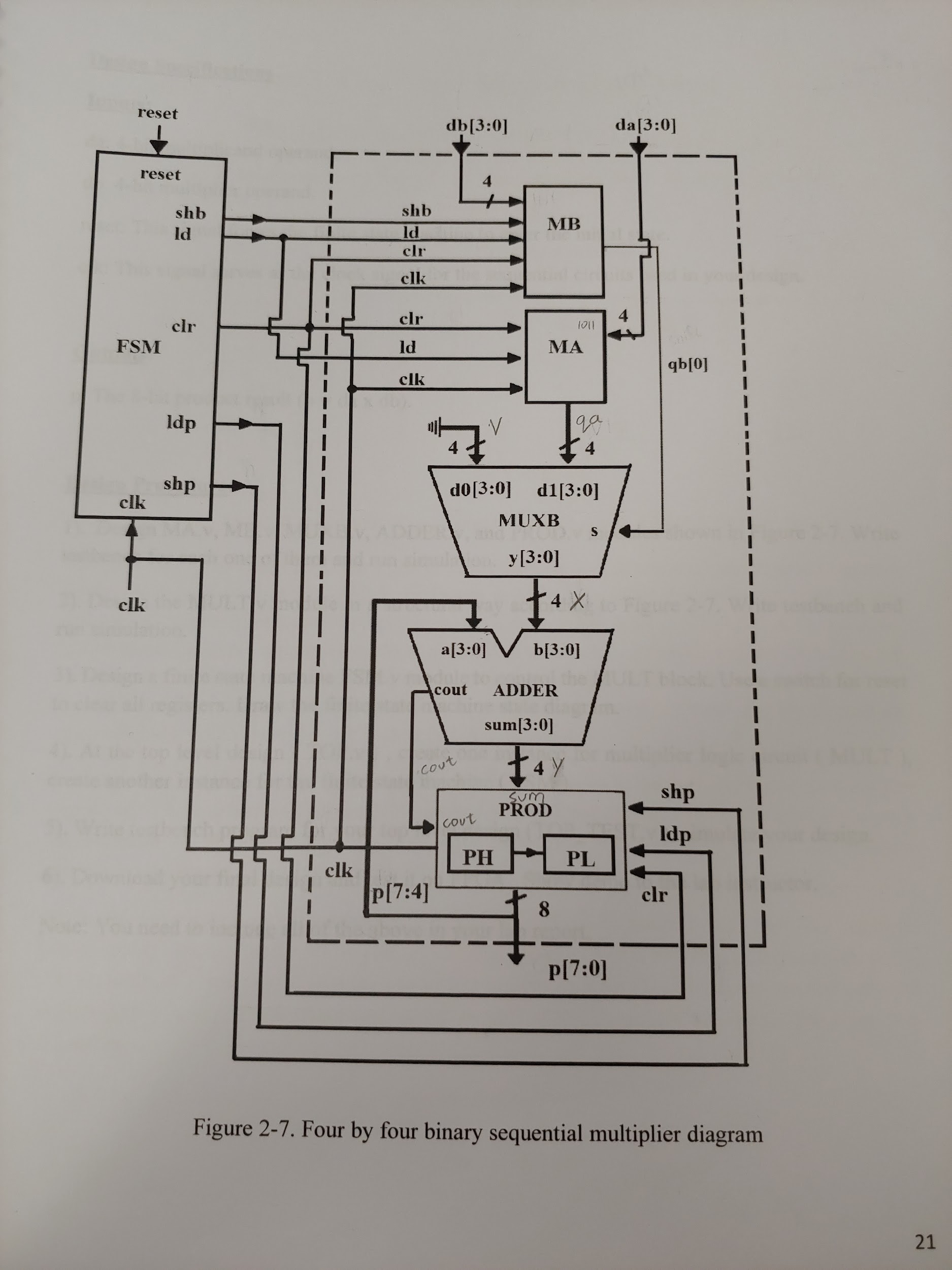
**Part 2: 4 by 4 Binary Sequential Multiplier Design**

**Part 2.1:**

This part of the lab uses a shift and add algorithm. The picture below is an example of a 4 by 4 binary sequential multiplication example. Every clock cycle, the multiplier is shifted to the right by one bit and its value is tested.

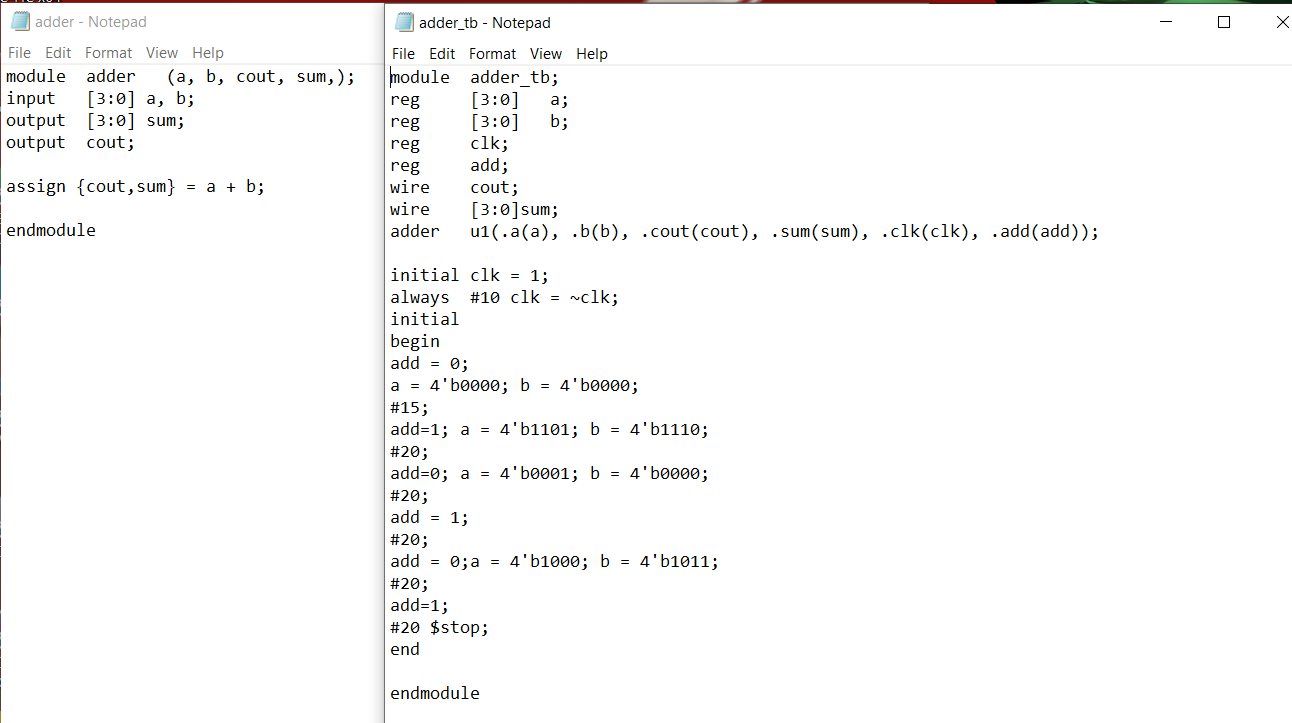


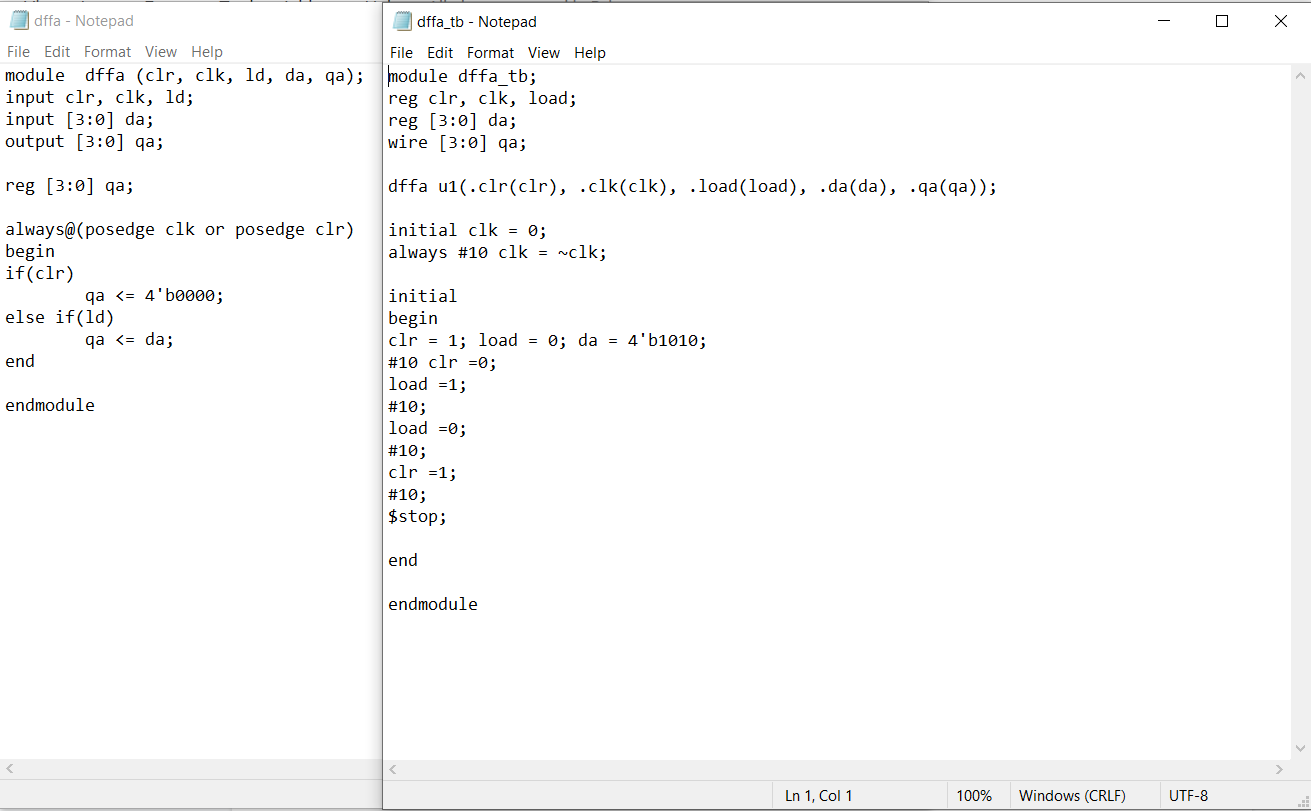
To create this in Verilog we need to design the circuit below. We need a FSM, dffa, dffb, muxb, adder, and a prod circuit. Once we created each one we can wire them accordingly. We will need to create a mult file where we do the circuits within the dotted line. Then we need to create a top level that takes everything and puts it together.

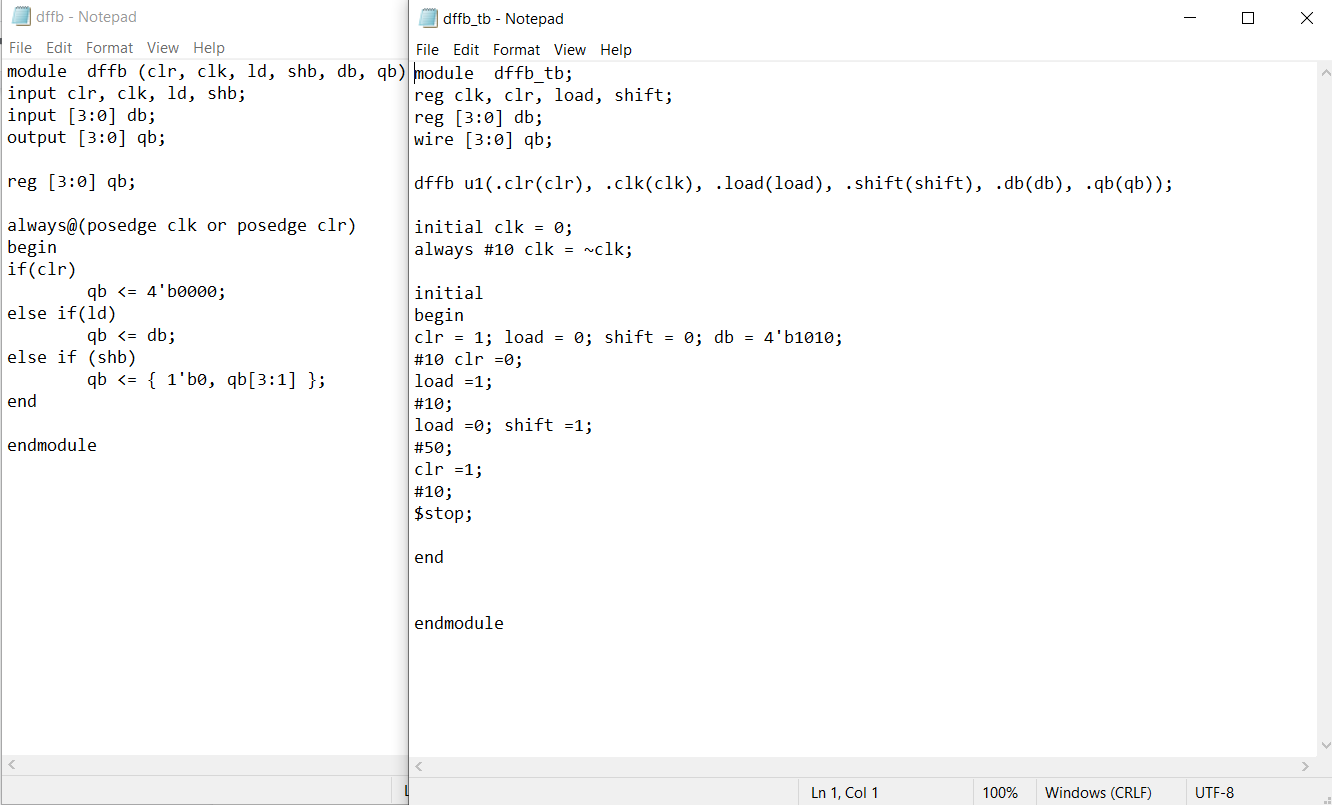


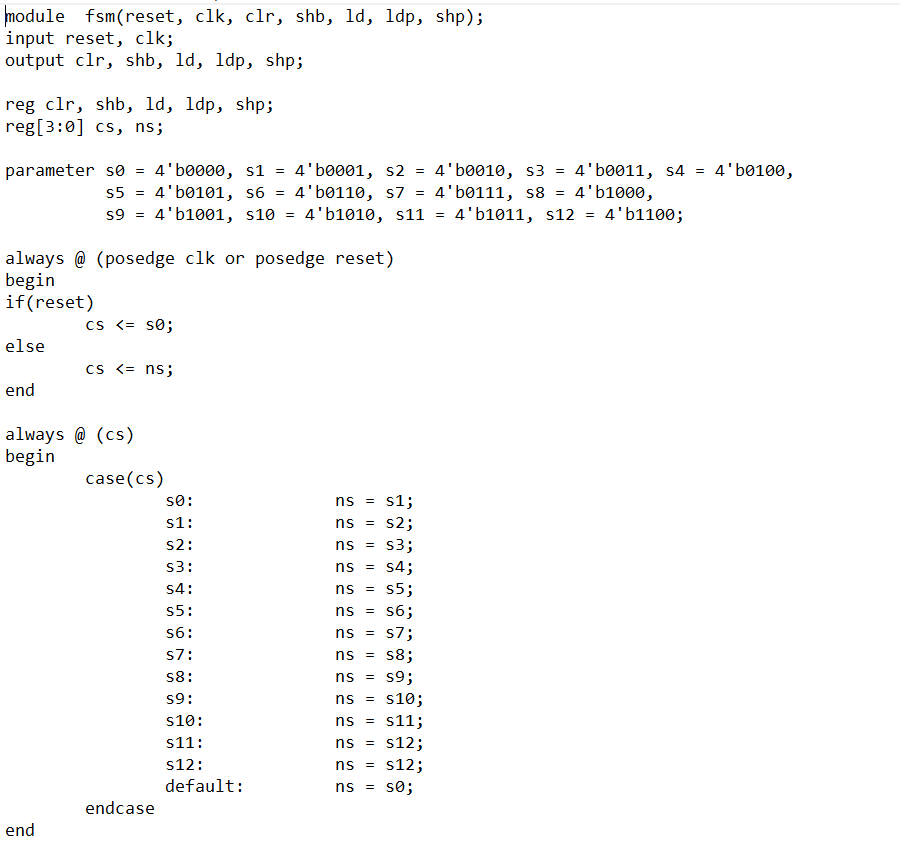
**Part 2.2:**

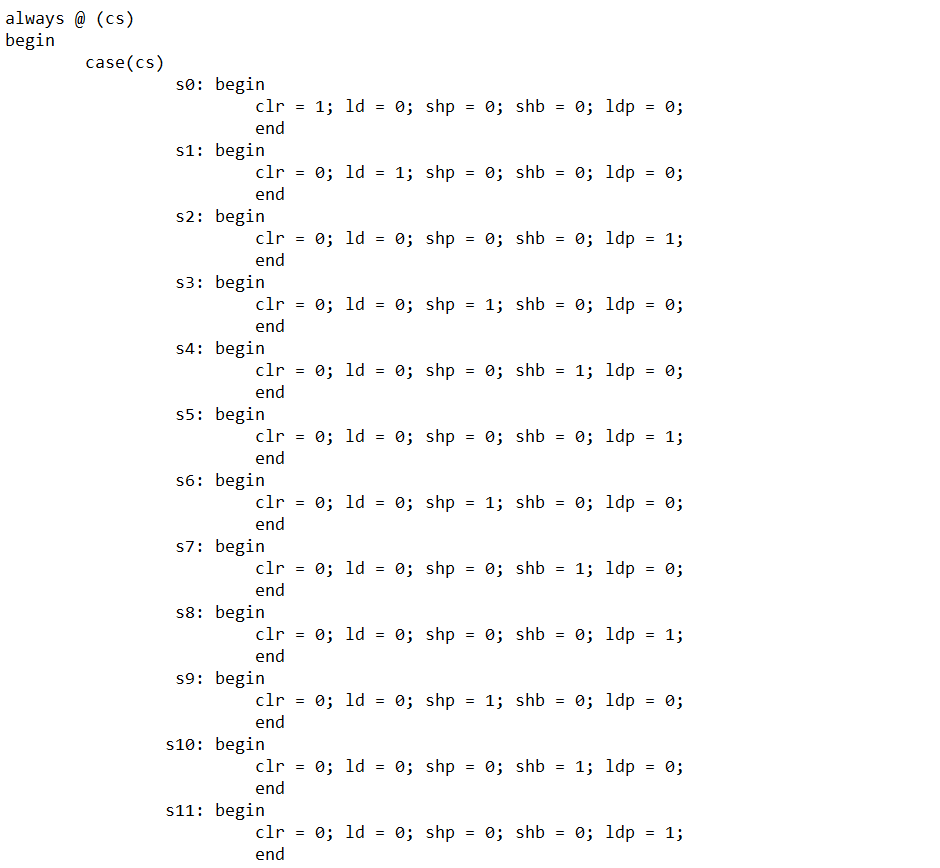
Source Code:

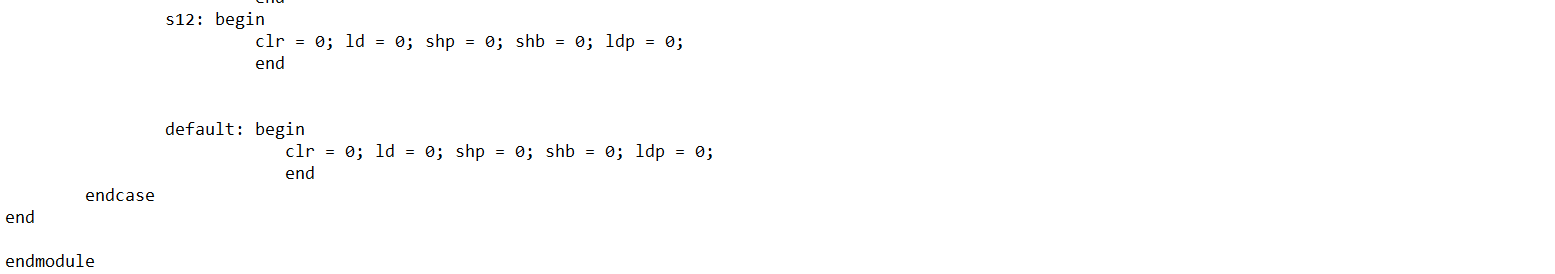


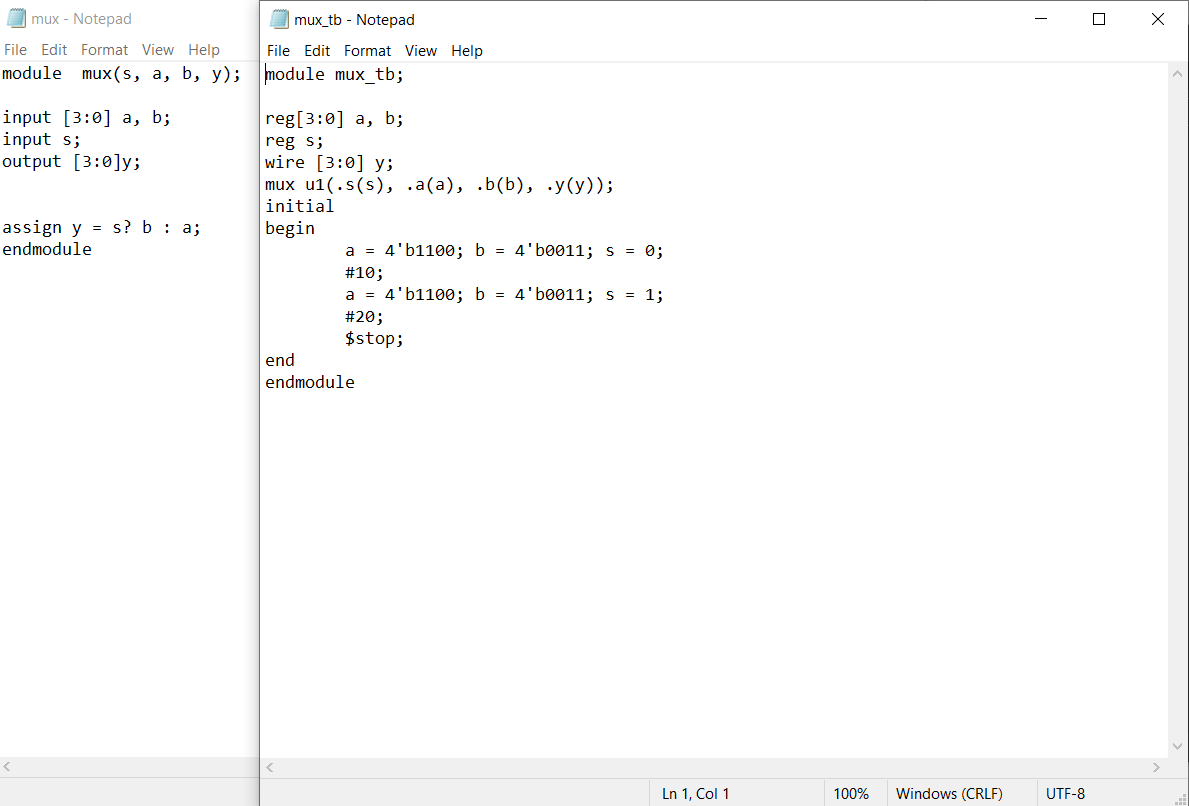
****

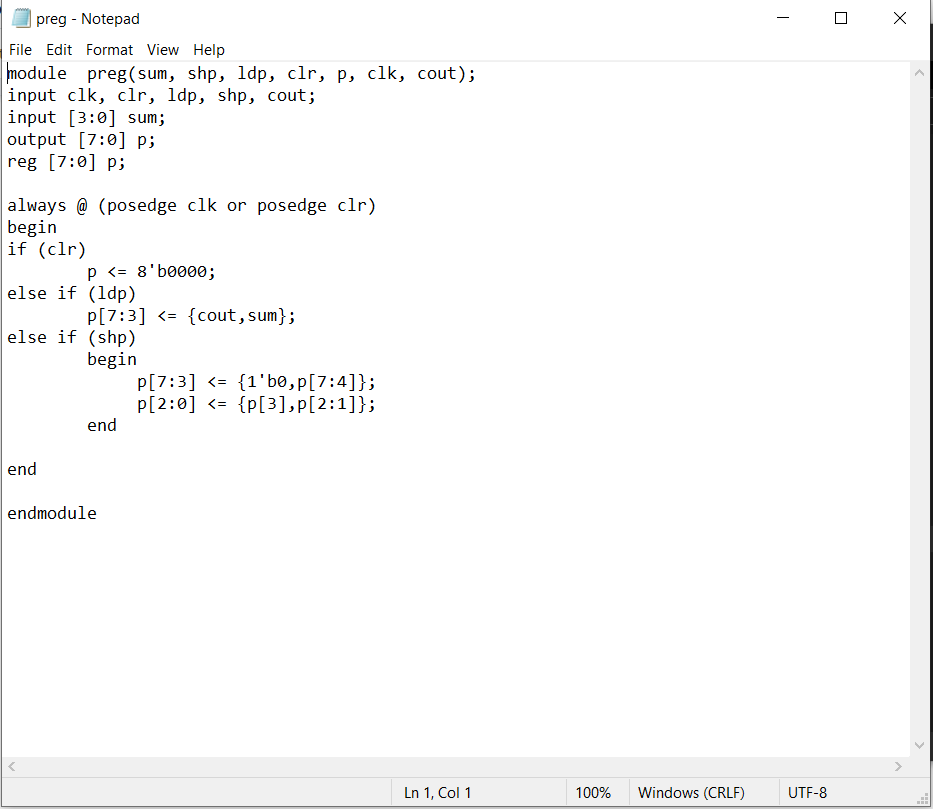
****

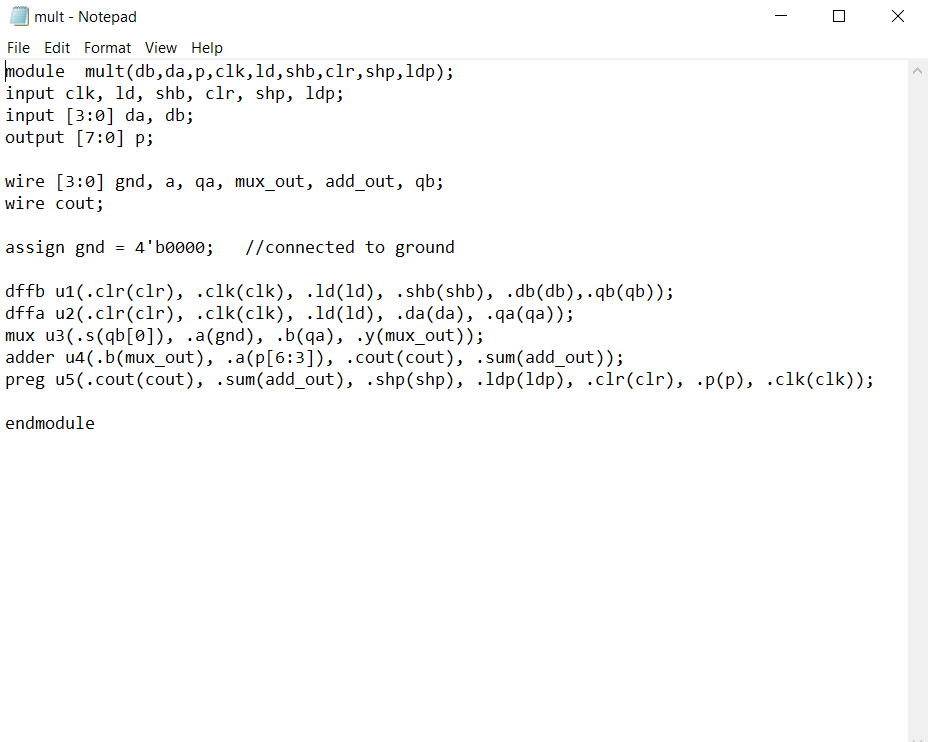
****

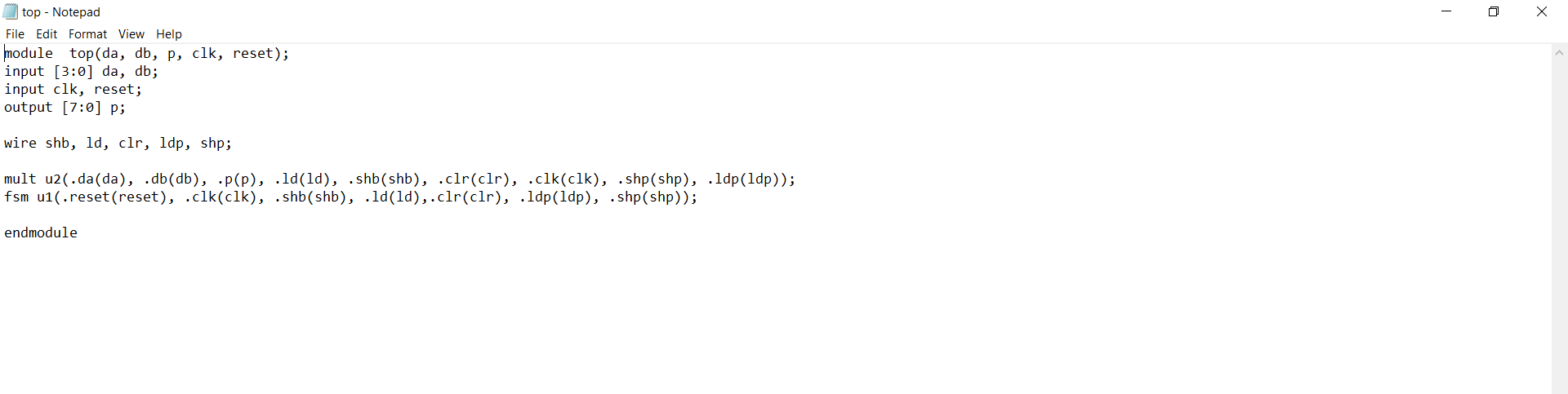
****

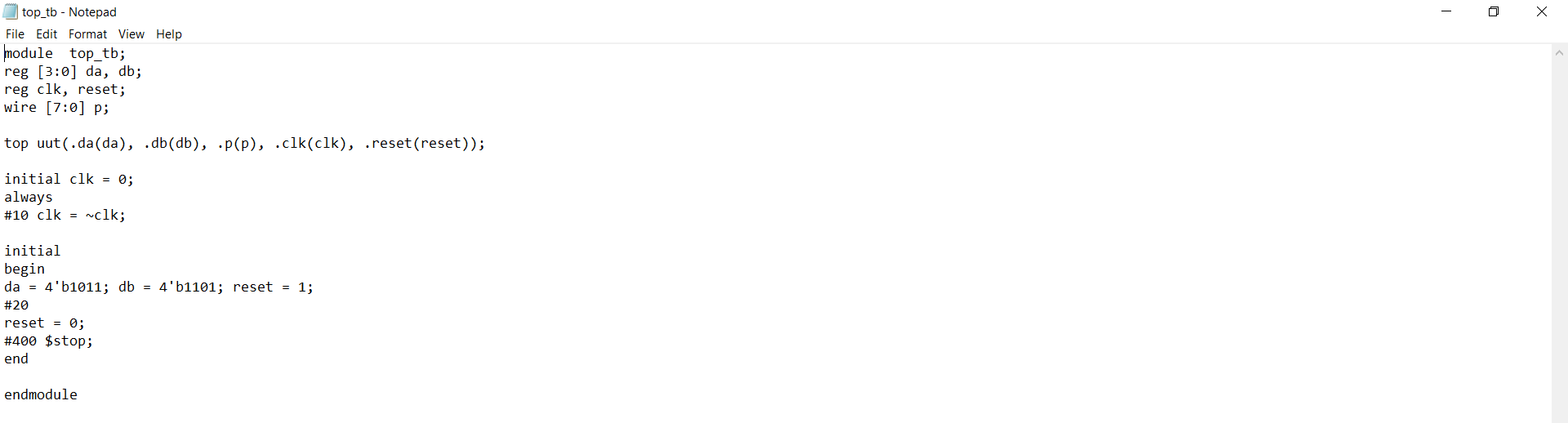
****

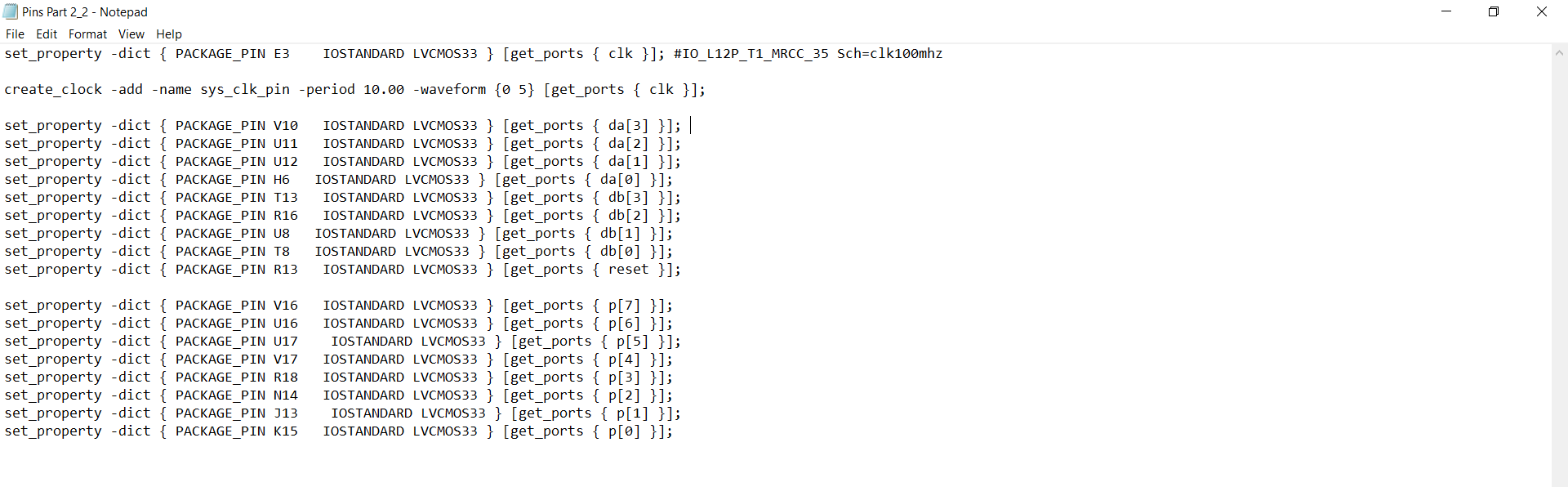
****

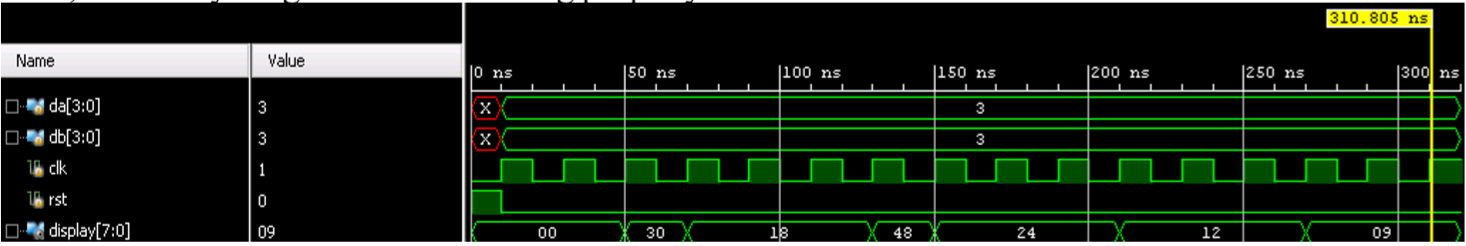












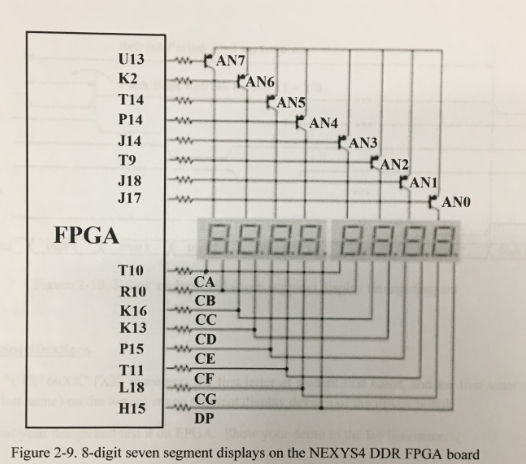
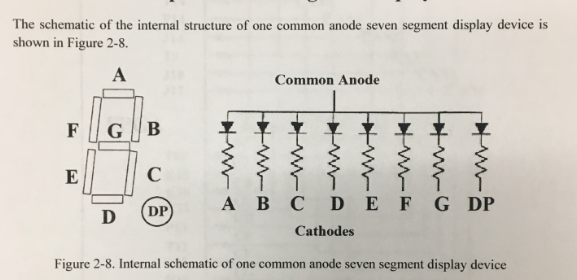
Result Discussion:

It took me a long time to get the 4 binary segmental multiplier multiplied values properly. I had issues mainly with mult and top. I also messed up the wiring a lot. The key findings of this part of the lab was how to put these components together and use sequential logic with the FSM to control all of these components. In this part of the lab, we figured out how to create shift registers and FSM and got complex circuits to work together. We had a multiplier on the FPGA that produces the correct values. The lab taught us a lot about how Verilog, Vivado and the NEXYS board works by setting pins and LEDs.

**Part 3: Character Display on 8-Digit Multiplexed Seven Segment Displays**

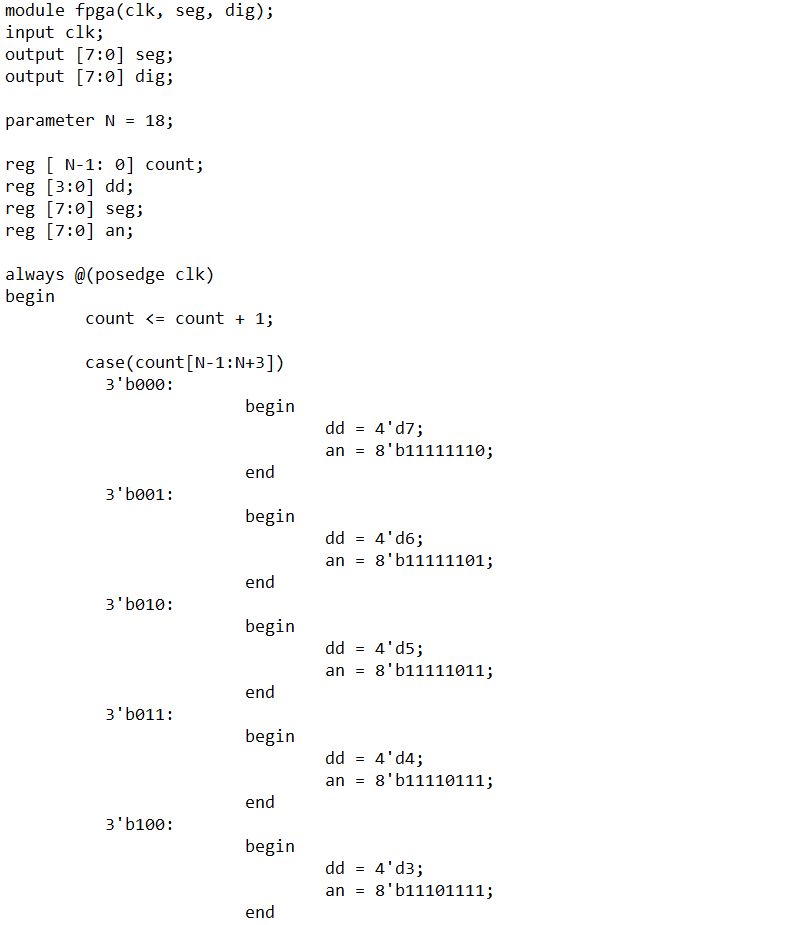
**Part 3.1:**

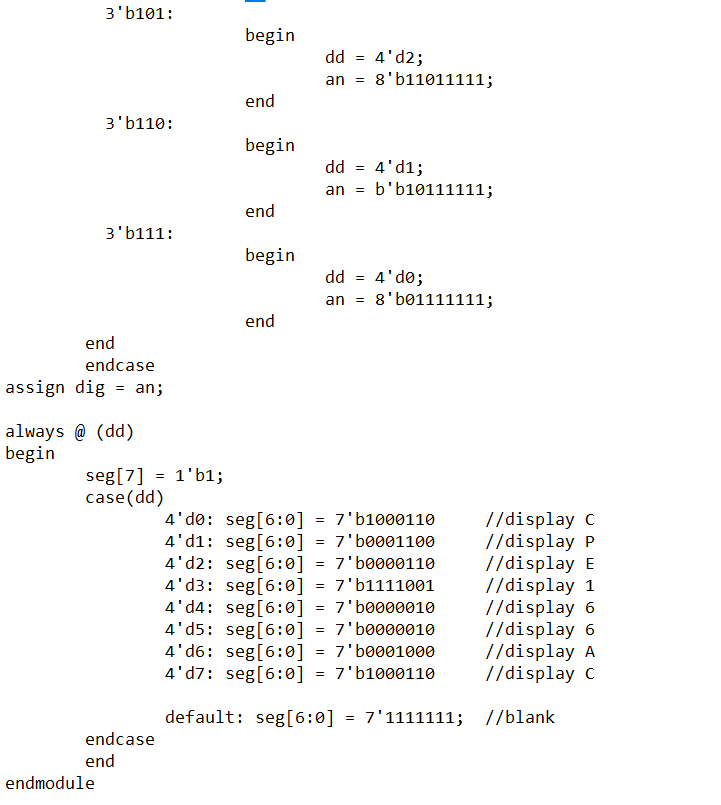
Here the anodes of the 7 segments are tied together as AN7, AN6, AN5, AN4, AN3, AN2, AN1, and AN0. For the cathodes they remain separated as CA, CB, CC, CD, CE, CF, CG, and DP. The FPGA drives the anode signals and corresponding cathode patterns of each digit in continuous fast-paced signals.

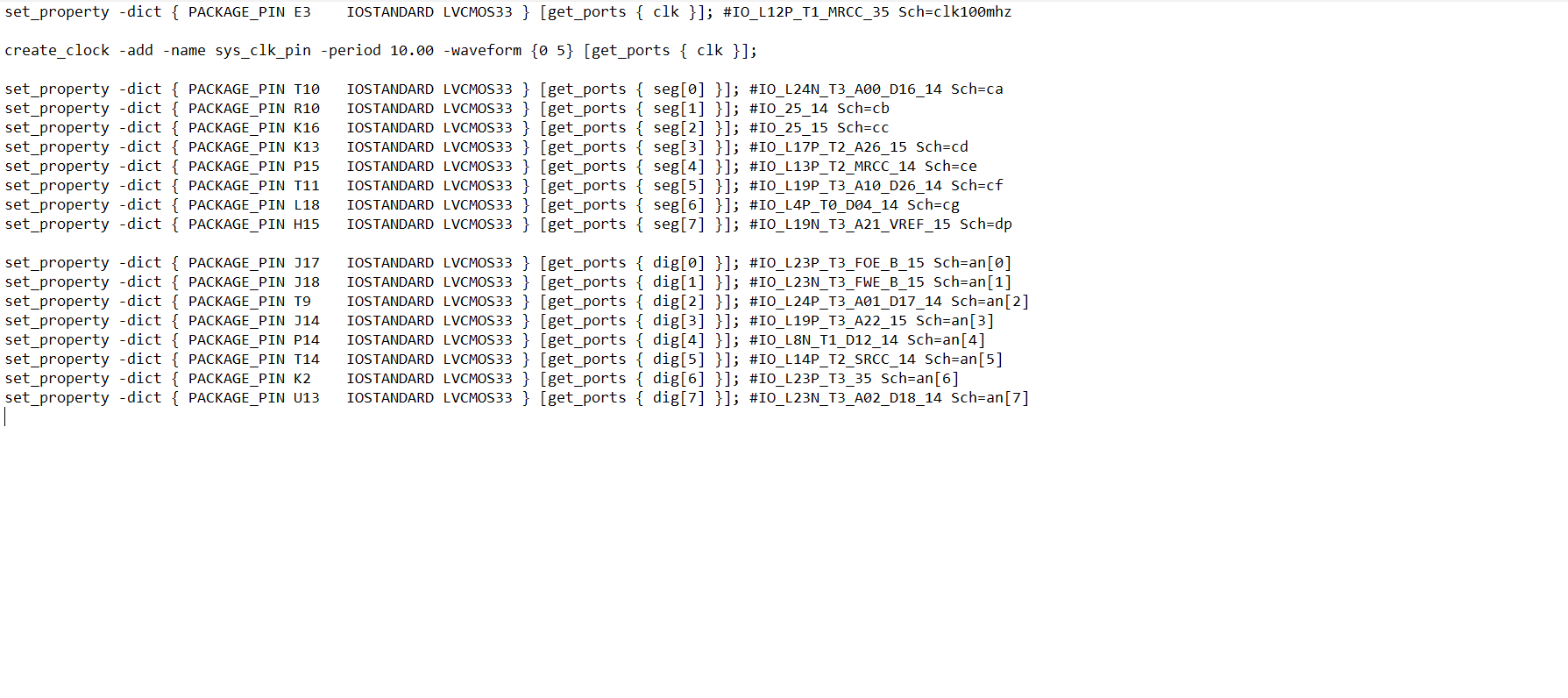


**Part 3.2:**

Source Code:







Result Discussion:

I managed to manipulate the 7-segment display. This lab was pretty easy with no issues. The code was already given to us online. I just had to understand how to configure the pins and manipulate the bytes so that the LED can light my initials. The goal of this part was how to write constraint files as well as how the anodes and diodes work with the seven-segment display to show values. It also helped us get familiar with the Vivado program and FPGA.

**Conclusion**

In conclusion, this lab helped me learn more about Verilog. It taught me how to code with hierarchy design and writing testbenches. I also got a better understanding of how waveforms are read and how to use it to check if our code was correct. The first part was simple. Create each circuit design and connect them at the end. Part 2 then introduced us to sequential design with the FSM. This part was the hardest for me. I struggled getting the right output. Part 3 taught us how to use a multiplexed seven segment display. Overall, This lab really gave me a better understanding of how coding in Verilog works and the steps to creating a circuit. It also helped to use the FPGA and seven segment display to see how the software and hardware interact with each other.