Blake Johnson
ECE 156a
R 5-8pm

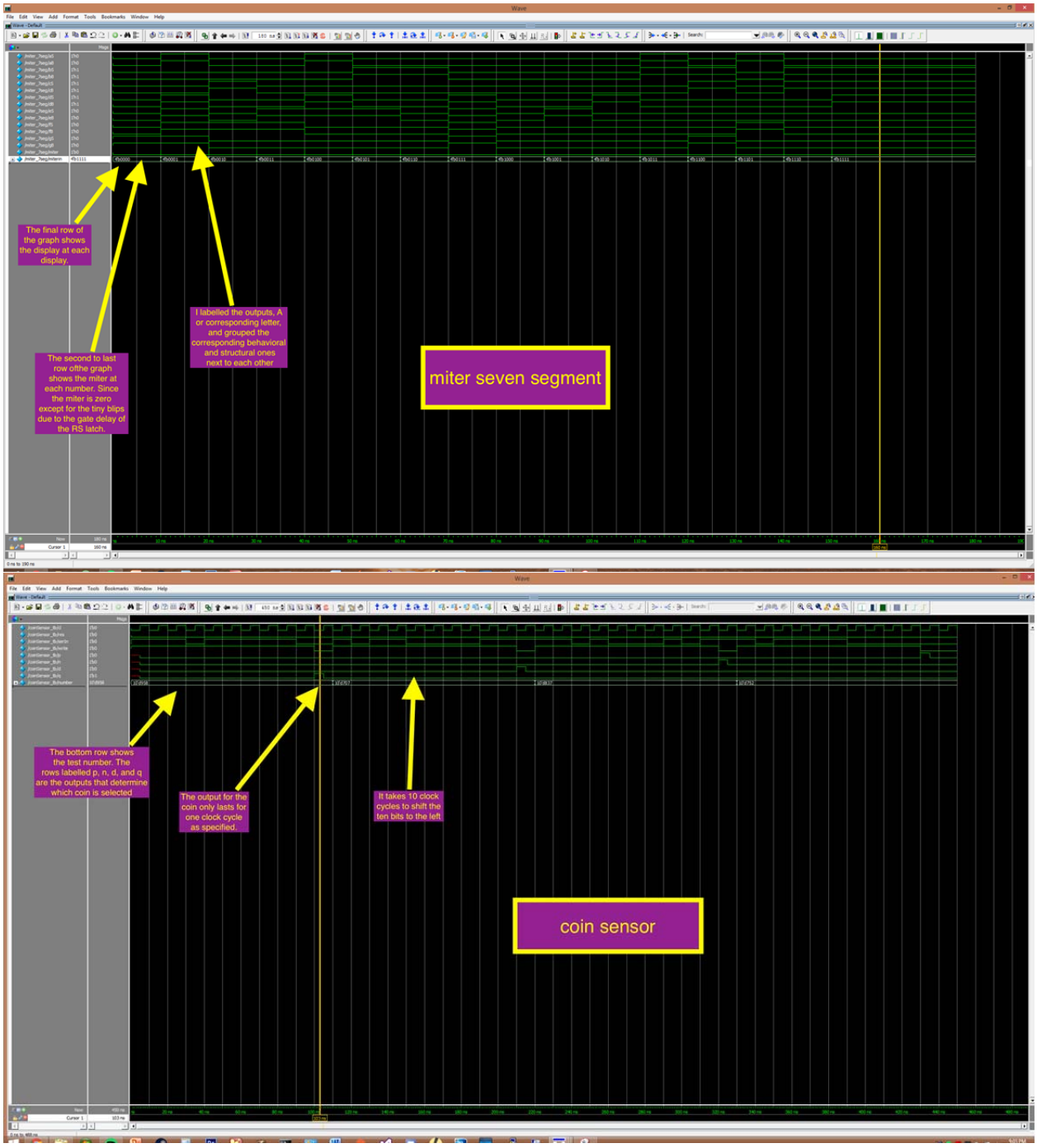
Homework 4

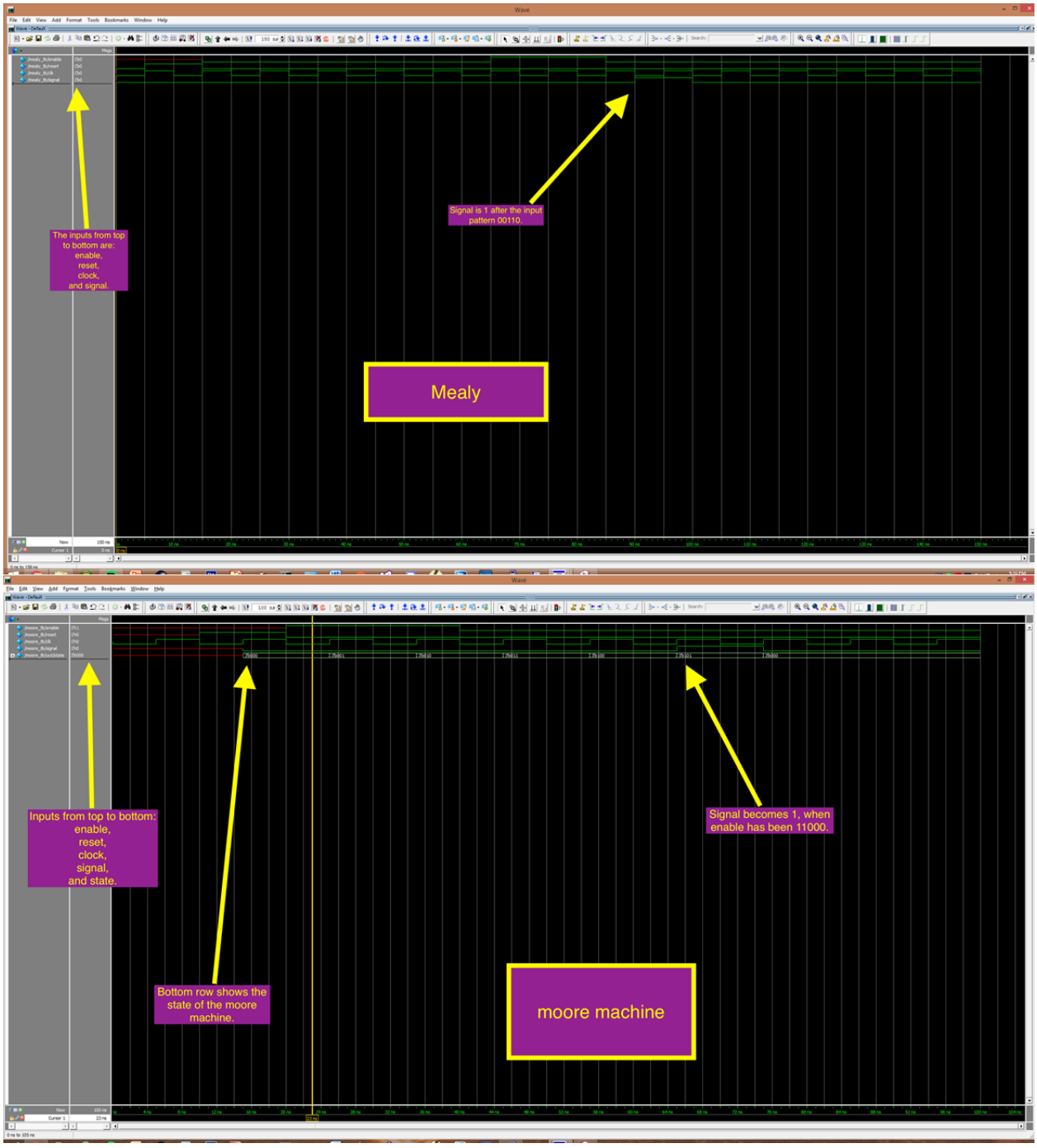
Intro:
This lab taught me how to compare two theoretically identical modules using xor gates and and gates. This lab also taught me how to create mealy and moore machines using different styles, as well as how to implement a coin scanner, that takes in the diameter of the coin and determines the type of coin. In addition to creating this module, we learned how to create functions and send information to the display.

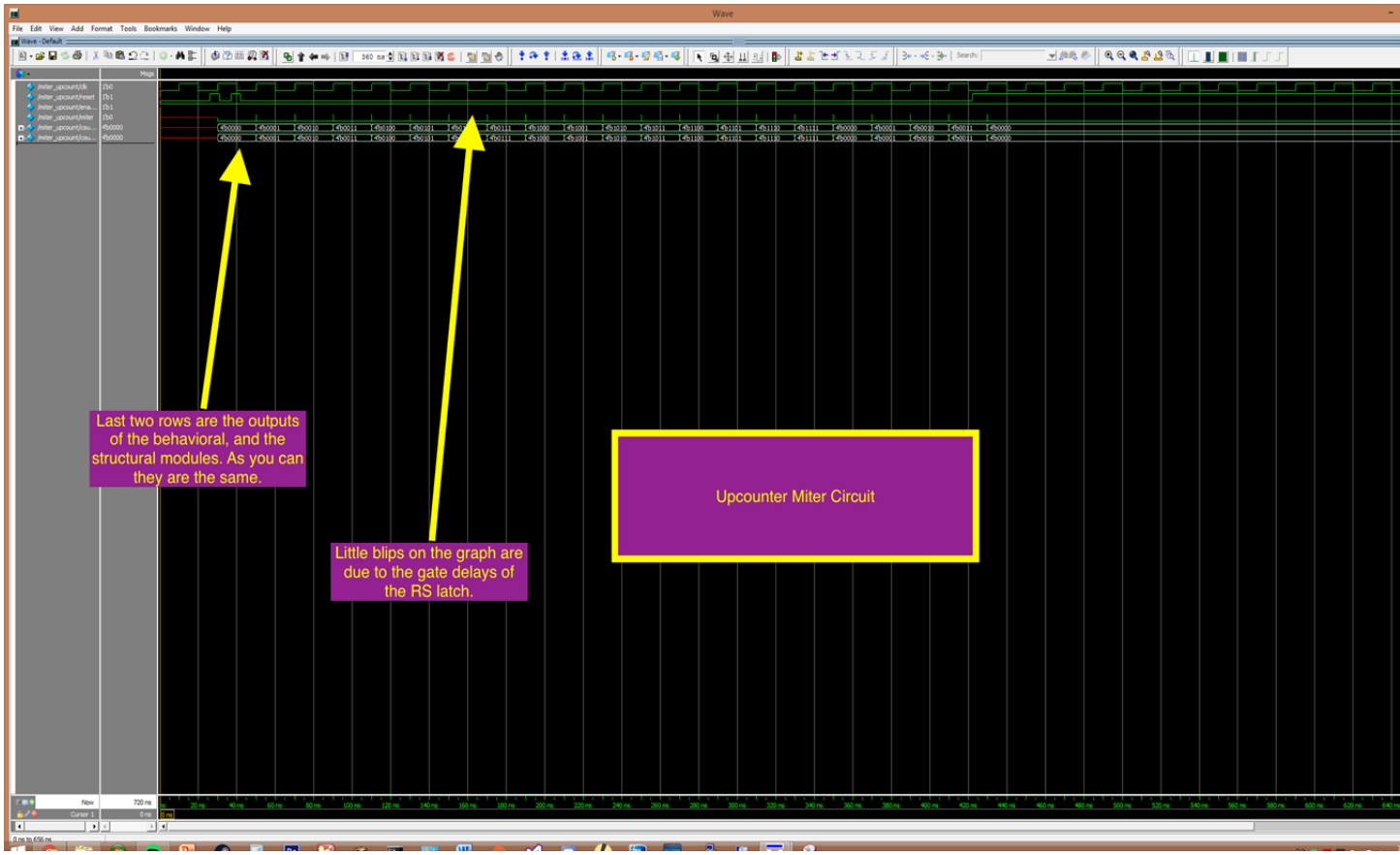
Procedure:
Starting off with the miter made sure that both the structural and behavioral implementations of the 7 segment decoder and the upcounter. Using xor gates to combine the corresponding bits of each module's output and then using or gates to combine all the different bit comparisons created a logic design that would output one if there was even the slightest bit of variation between the behavioral and structural models. After this, I made both the mealy and moore machines from homework 1. The machines were pretty straightforward and I used case statements with conditional statements to make the state machine's connections. Next were the coin sensor and its testbench which was by far the most difficult part of the lab. I started off by creating a moore machine for quarters but as I was trying to make the machine's connections which were somewhat confusing, I reread the instructions and realized I missed the part about the error flexibility. If I added this flexibility using a moore machine it would be near impossible due to how tedious and human error prone that project would be. Instead I used logical shift left to load the 10 bit number into a register from which I used the built in Verilog conversions to switch to decimal and then compare using the decimal number in my logic. This worked once I figured out all the bugs in my code. The testbench was the next hurdle. To test my design I wanted to use randomization because it would help cover more cases. I created a random function based on the outline given by the handout and used it with different parameters to implement a controlled random variation to my testing. From there I tested the functionality of all the types of coins and insured that everything worked.

Tests:
The mealy and moore machine tests consisted of the number they were trying to identify so the only real struggle was with aligning the clock and making sure I had the right amount/order of 1's and 0's. The miter tests were a little tricker because I actually found an error in my code that I fixed before I tuned in my assignments from the previous weeks but I didn't have the most updated version on my computer at home. Once I figured this out, the tests ran smoothly. The coin sensor test proved to be a challenge due to the wide variety of possible test cases, but the random function helped cover a lot of them.

Conclusion:
The lab consisted of many different aspects of Verilog with an emphasis on state machines. The coin sensor was a complex state machine, and due to this complexity, its implementation had to be tackled in a different manner than the simple five bit mealy and moore state machines. The miters showed how it was easy to compare a large number of bits quickly using specific combinations of bitwise operators.


The final row of the graph shows the display at each display.
The second to last row ofthe graph shows the miter at each number. Since the miter is zero except for the tiny blips due to the gate delay of the RS latch.
I labelled the outputs, A or corresponding letter, and grouped the corresponding behavioral and structural ones next to each other
miter seven segment
The bottom row shows the test number. The rows labelled p, n, d, and q are the outputs that determine which coin is selected
The output for the coin only lasts for one clock cycle as specified.
It takes 10 clock cycles to shift the ten bits to the left
coin sensor


The inputs from top to bottom are: enable, reset, clock, and signal.
Signal is 1 after the input pattern 00110.
Mealy
Inputs from top to bottom: enable, reset, clock, signal, and state.
Bottom row shows the state of the moore machine.
Signal becomes 1, when enable has been 11000.
moore machine


Last two rows are the outputs of the behavioral, and the structural modules. As you can they are the same.
Little blips on the graph are due to the gate delays of the RS latch.
Upcounter Miter Circuit