Blake Johnson
ECE 156a
Homework 3
Section: R 5-8pm

HW3

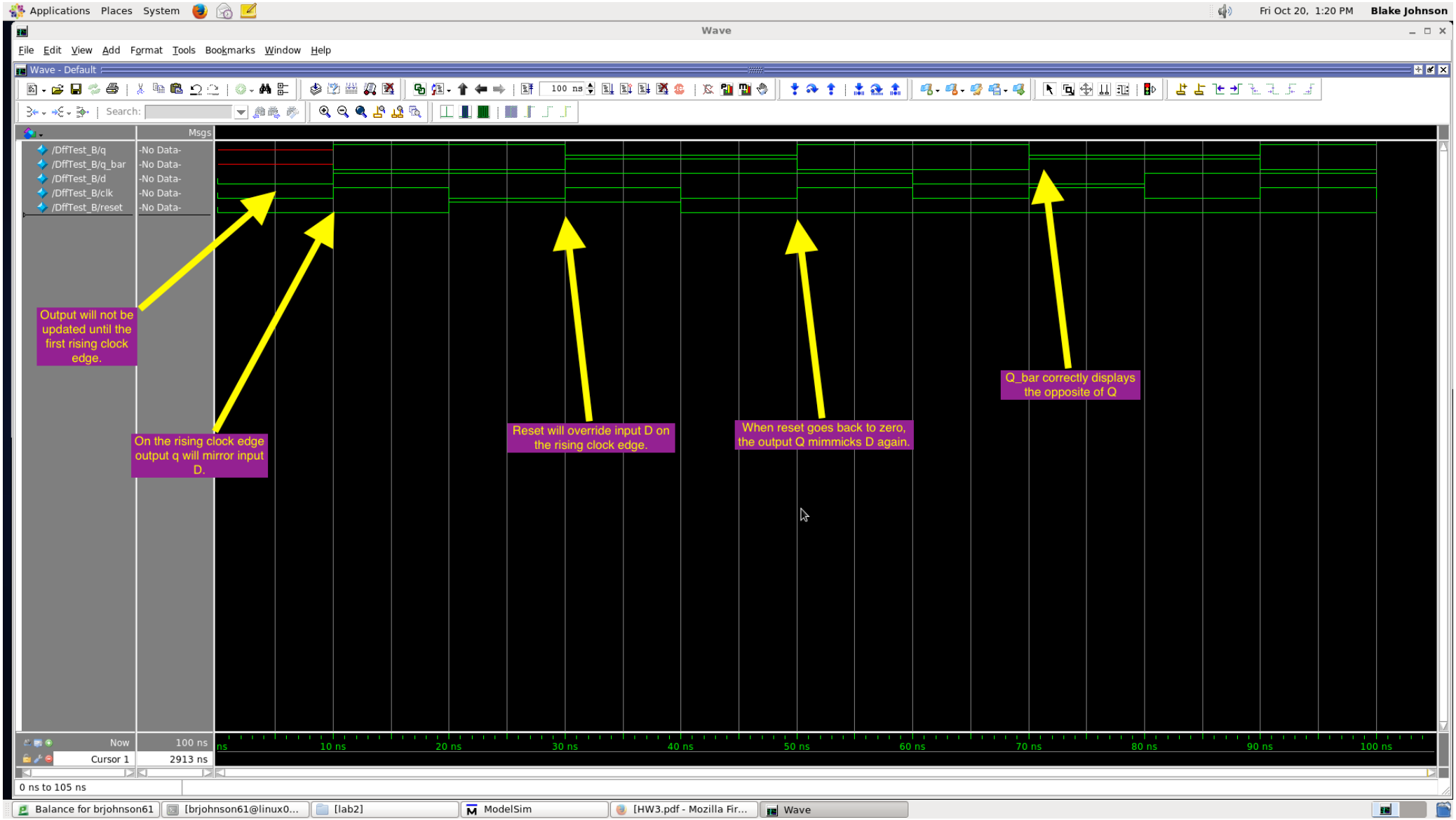
Intro:
In this lab I used behavioral Verilog to model the D flip-flop, seven segment decoder, and up-counter from the previous lab. Overall, working in behavioral Verilog instead of structural Verilog has had many upsides. The code is simpler and shorter, which helps prevent mistakes and confusion.

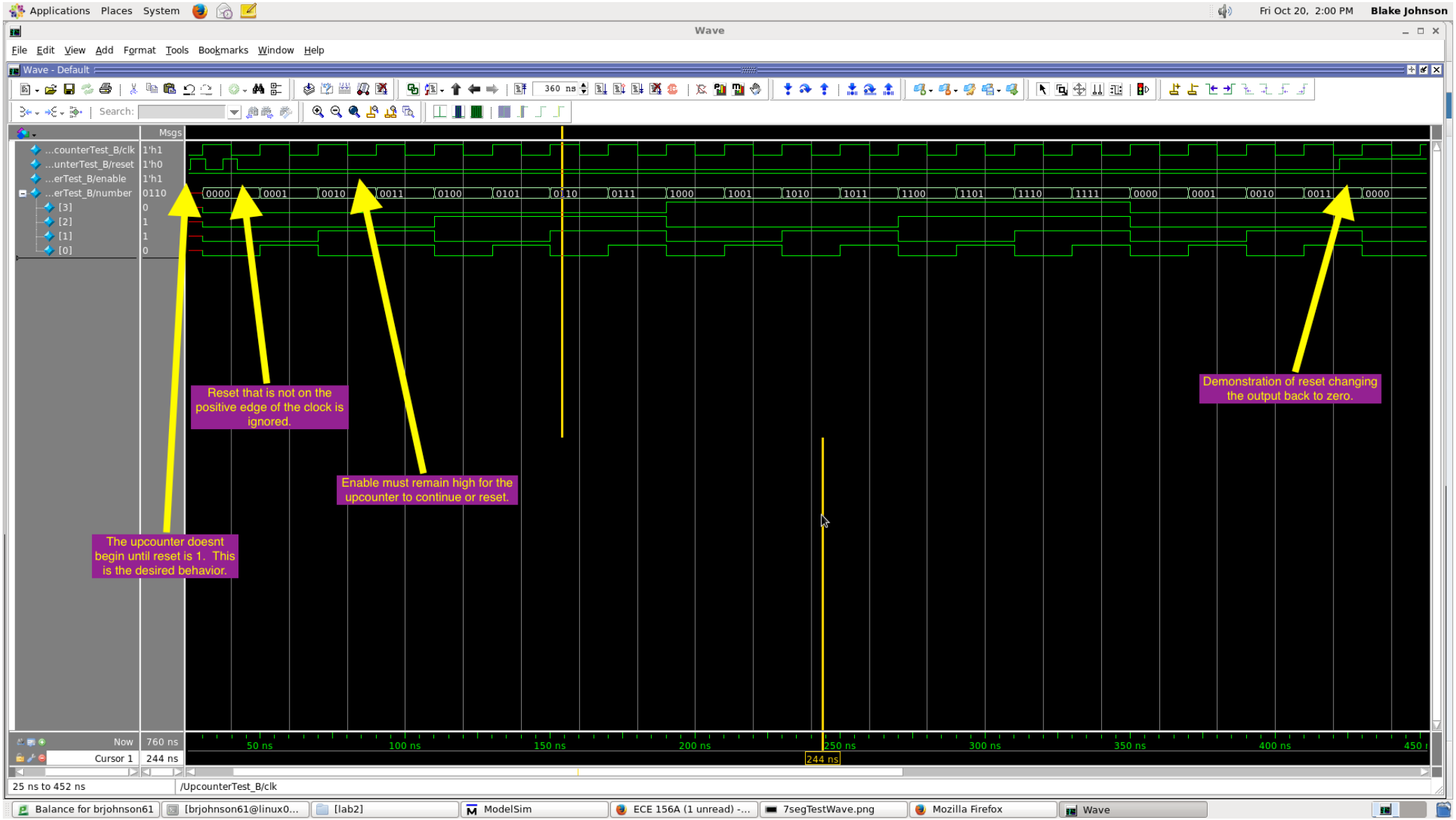
Procedure:
First I started with the D latch and RS latch to get a feel for using behavioral Verilog. Since this lab asked for a holistic design, I did not end up using the RS and D latch modules in my final implementation. After I finished the D flip-flop I implemented the up-counter. Because I used behavioral Verilog, I did not need to keep track of multiple wires and gates simultaneously. I was able to implement this module pretty quickly, however I had to put some care into making sure the module took the edge cases into account. Finally, the seven segment decoder was the last module I implemented. Because we are not supposed to implement large Boolean sum of products assignment statements using assign, I used a case method for each segment of the display. To avoid writing out many combinations, I used the casez feature on Verilog that allows you to put a "?" in place of a don't-care. This implementation worked well after I discovered this.

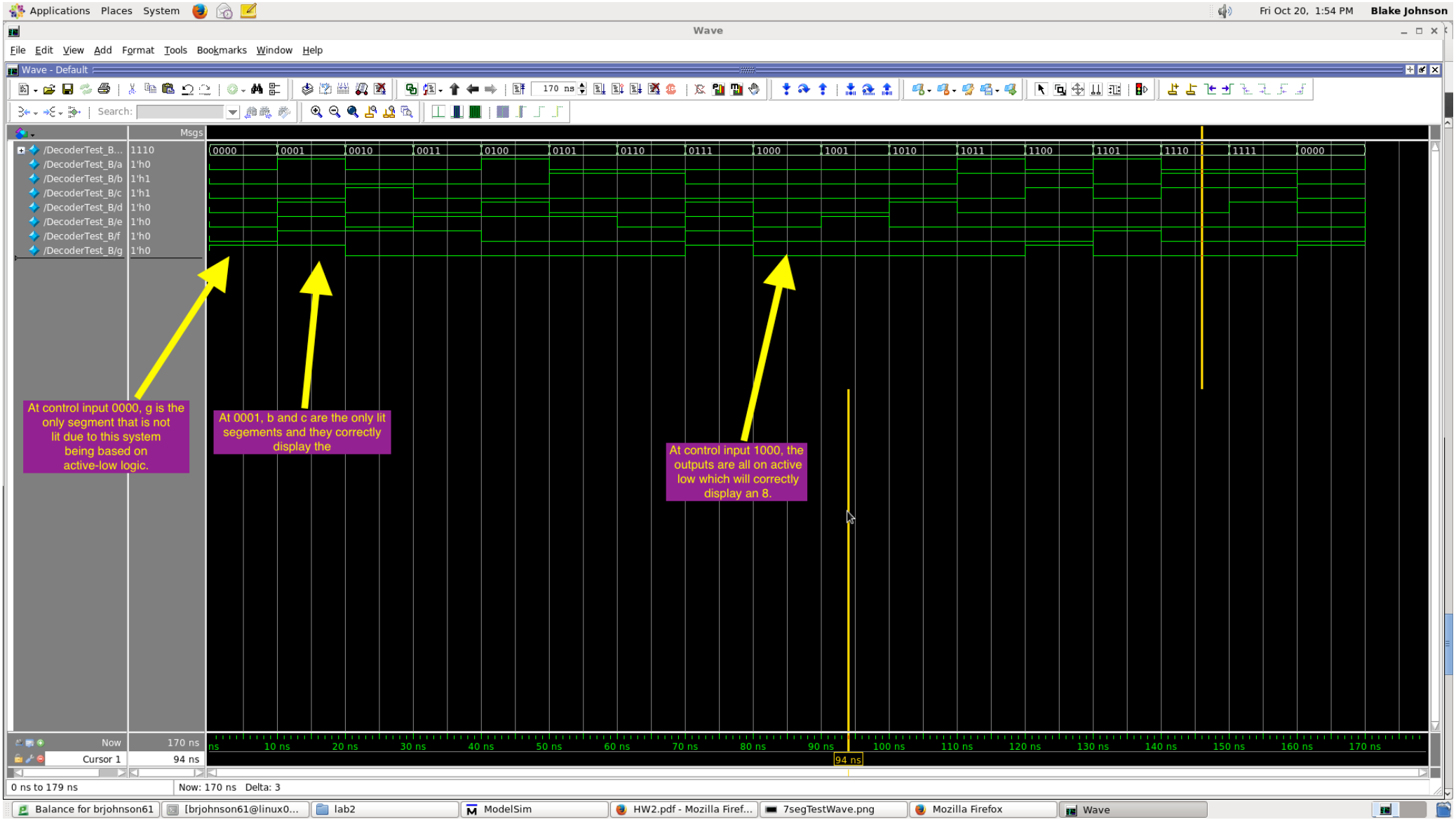
Testing:
Since this lab implements the same modules that are supposed to behave identically, I used the same test files as lab 1. However, I added a few extra test cases on each one to demonstrate a more complete module.

Conclusion:
This lab implemented the same three modules as before, except this time using behavioral Verilog. Behavioral Verilog has many efficient qualities that speed up writing and simulation time and make for overall more efficient work in the real world. Implementing the same modules in both structural and behavioral Verilog has given me a lot of insight into each style of Verilog. Because I trudged through the tediousness of structural Verilog, I now value the efficiency of behavioral Verilog, but I also understand what going on under the hood and can translate the statements into structural terms if I need to.

Images listed in this order: D flip flop, Up-counter, Decoder


Applications Places System
Fri Oct 20, 1:20 PM Blake Johnson
Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
Search:
Msgs
/DffTest_B/q -No Data-
/DffTest_B/q_bar -No Data-
/DffTest_B/d -No Data-
/DffTest_B/clk -No Data-
/DffTest_B/reset -No Data-
Output will not be updated until the first rising clock edge.
On the rising clock edge output q will mirror input D.
Reset will override input D on the rising clock edge.
When reset goes back to zero, the output Q mimmicks D again.
Q_bar correctly displays the opposite of Q
Now 100 ns
Cursor 1 2913 ns
ns 10 ns 20 ns 30 ns 40 ns 50 ns 60 ns 70 ns 80 ns 90 ns 100 ns
0 ns to 105 ns
Balance for brjohnson61
[brjohnson61@linux0...
[lab2]
ModelSim
[HW3.pdf - Mozilla Fir...
Wave


Applications Places System
Fri Oct 20, 2:00 PM Blake Johnson
Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
Search:
Msgs
...counterTest_B/clk 1'h1
...unterTest_B/reset 1'h0
...erTest_B/enable 1'h1
...erTest_B/number 0110
[3] 0
[2] 1
[1] 1
[0] 0
0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111 0000 0001 0010 0011 0000
Reset that is not on the positive edge of the clock is ignored.
Enable must remain high for the upcounter to continue or reset.
The upcounter doesnt begin until reset is 1. This is the desired behavior.
Demonstration of reset changing the output back to zero.
Now 760 ns
Cursor 1 244 ns
50 ns 100 ns 150 ns 200 ns 250 ns 300 ns 350 ns 400 ns
244 ns
25 ns to 452 ns
/UpcounterTest_B/clk
Balance for brjohnson61
[brjohnson61@linux0...
[lab2]
ModelSim
ECE 156A (1 unread) -...
7segTestWave.png
Mozilla Firefox
Wave


Applications Places System
Fri Oct 20, 1:54 PM Blake Johnson
Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
Search:
Msgs
/DecoderTest_B... 1110
/DecoderTest_B/a 1'h0
/DecoderTest_B/b 1'h1
/DecoderTest_B/c 1'h1
/DecoderTest_B/d 1'h0
/DecoderTest_B/e 1'h0
/DecoderTest_B/f 1'h0
/DecoderTest_B/g 1'h0
0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111 0000
At control input 0000, g is the only segment that is not lit due to this system being based on active-low logic.
At 0001, b and c are the only lit segements and they correctly display the
At control input 1000, the outputs are all on active low which will correctly display an 8.
Now 170 ns
Cursor 1 94 ns
ns 10 ns 20 ns 30 ns 40 ns 50 ns 60 ns 70 ns 80 ns 90 ns 100 ns 110 ns 120 ns 130 ns 140 ns 150 ns 160 ns 170 ns
94 ns
0 ns to 179 ns
Now: 170 ns Delta: 3
Balance for brjohnson61
[brjohnson61@linux0...
lab2
ModelSim
HW2.pdf - Mozilla Firef...
7segTestWave.png
Mozilla Firefox
Wave