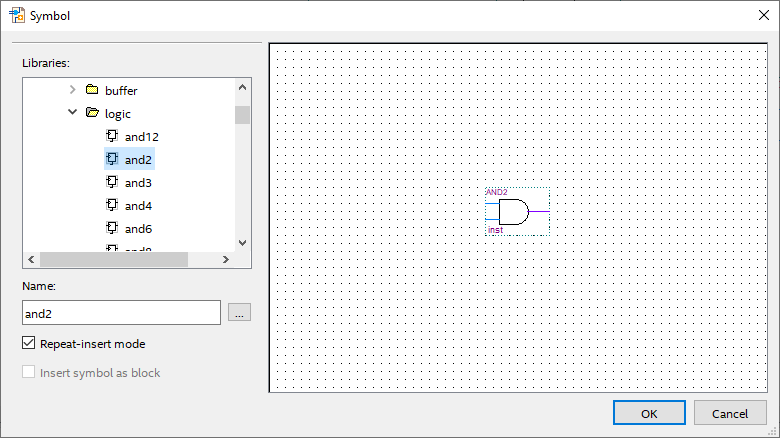
ZHANG\_CHUE

Task 2: “2:1 Mux”

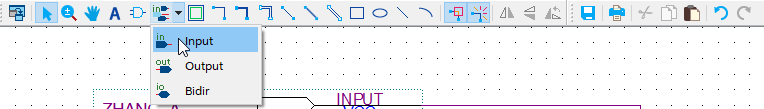


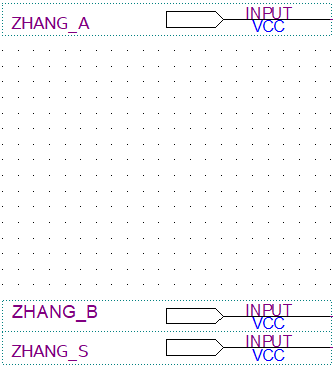


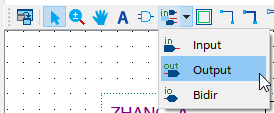
I start by clicking the symbol icon in Quartus and picking out the logic gates. For the 2 to 1 Mux, I will be using two AND-gates, one OR-gate and one NOT-gate.



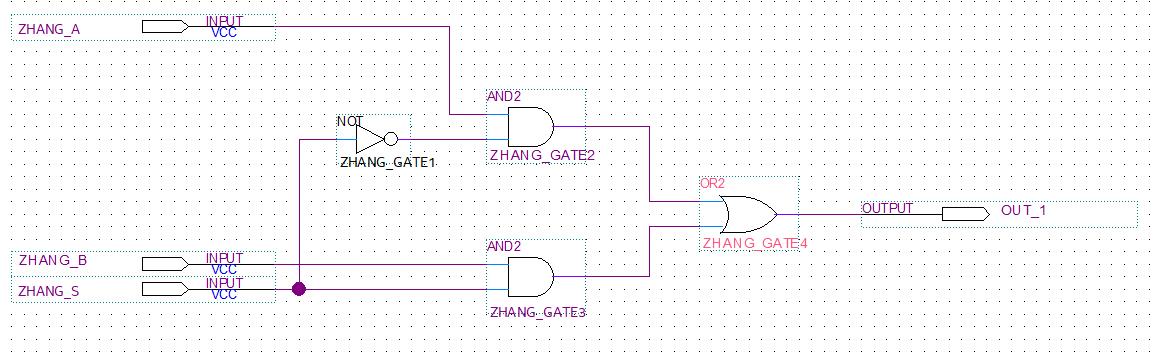
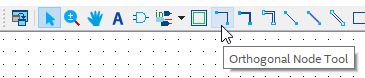
Next, I go to the pins tool dropdown menu and select input. For the MUX, I used 3 inputs



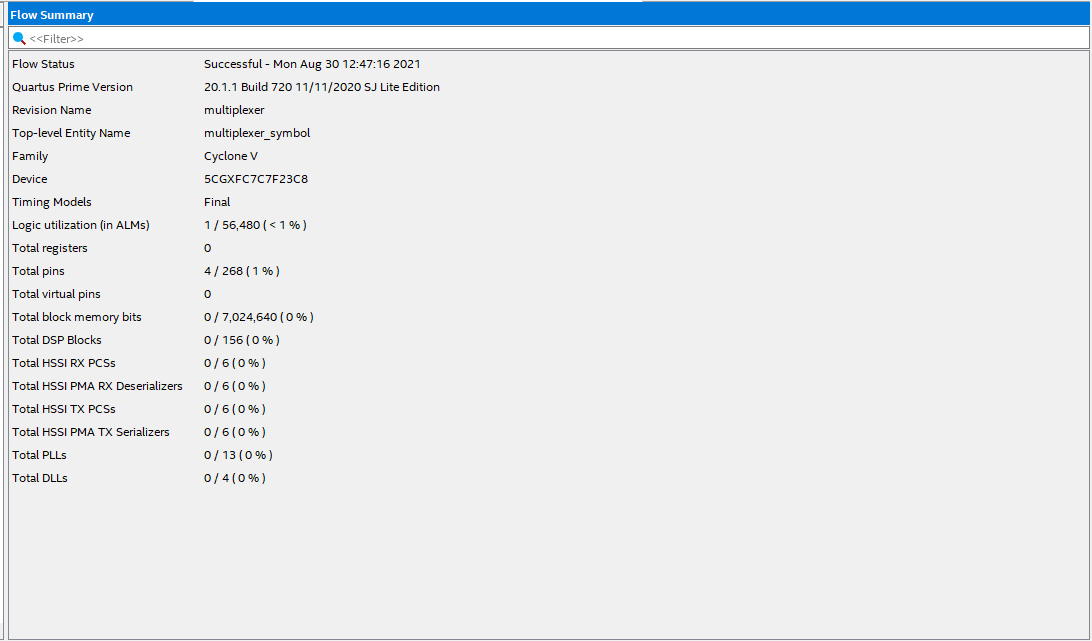
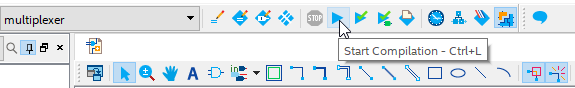
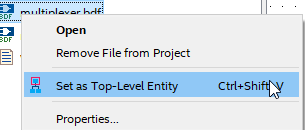


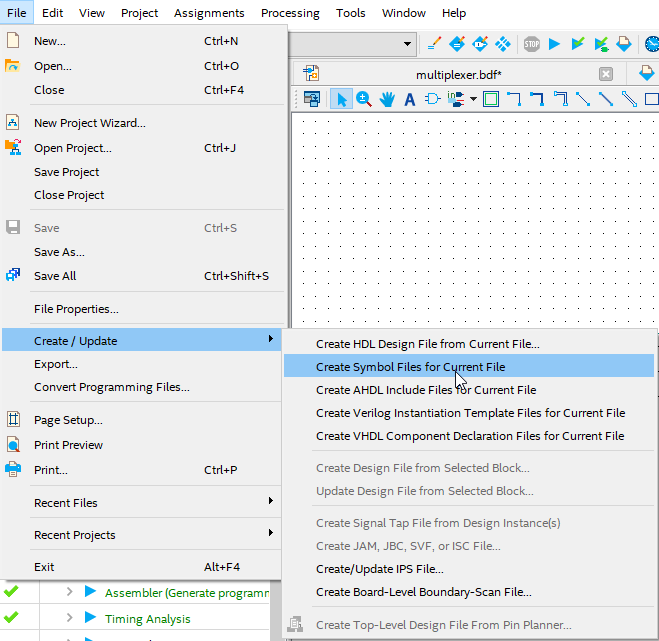
I repeat the same steps again for output Pins, however, I only use one output pin

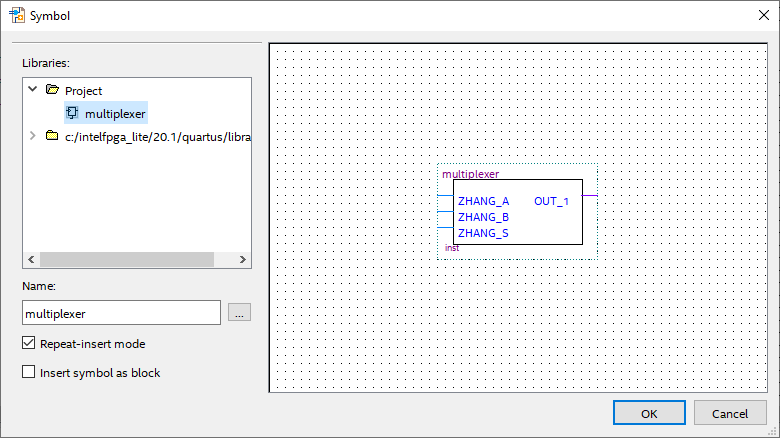
Then I use the Orthogonal Node tool to connect the inputs and outputs to the gates.

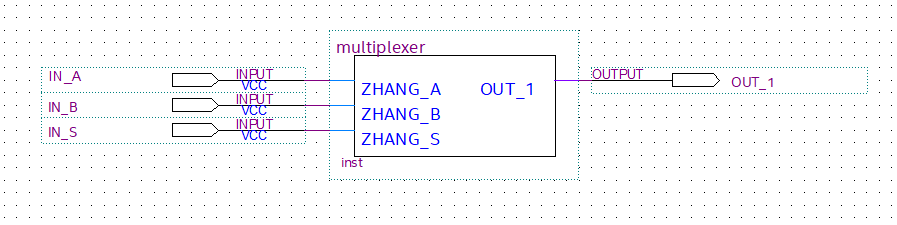


In the figure above, what is shown is a 2:1 MUX where we have 3 inputs. Input A, Input B and a Selector input. The selector is inputted in the NOT-gate which is then inputted in the ZHANG\_GATE2 with ZHANG\_A input. It is also inputted into the ZHANG\_GATE3 with ZHANG\_B input and finally both ZHANG\_GATE2 and ZHANG\_GATE3 and inputted together into ZHANG\_GATE4 (OR-gate) where the result will be outputted.

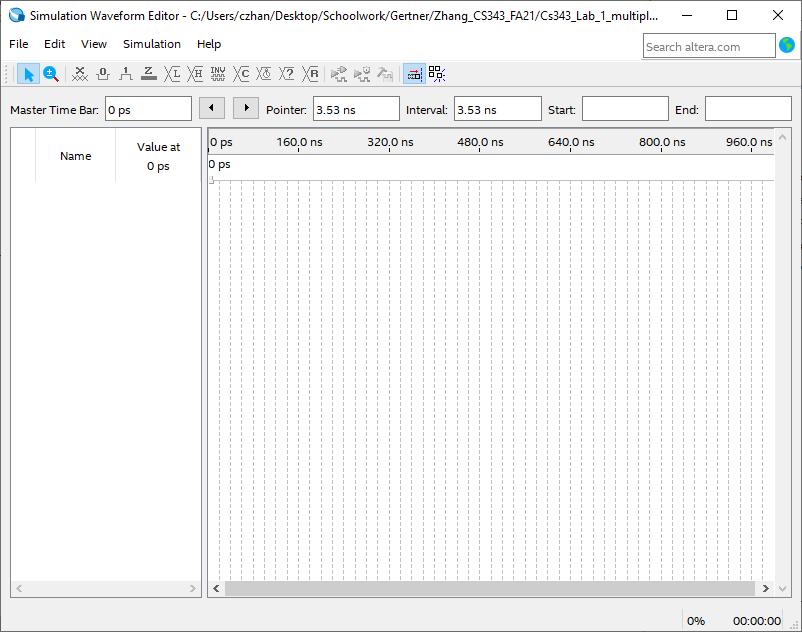
  
In this next few screenshot, what I did was I set the MUX bdf file to top level where I then compile to ensure that everything is working properly.



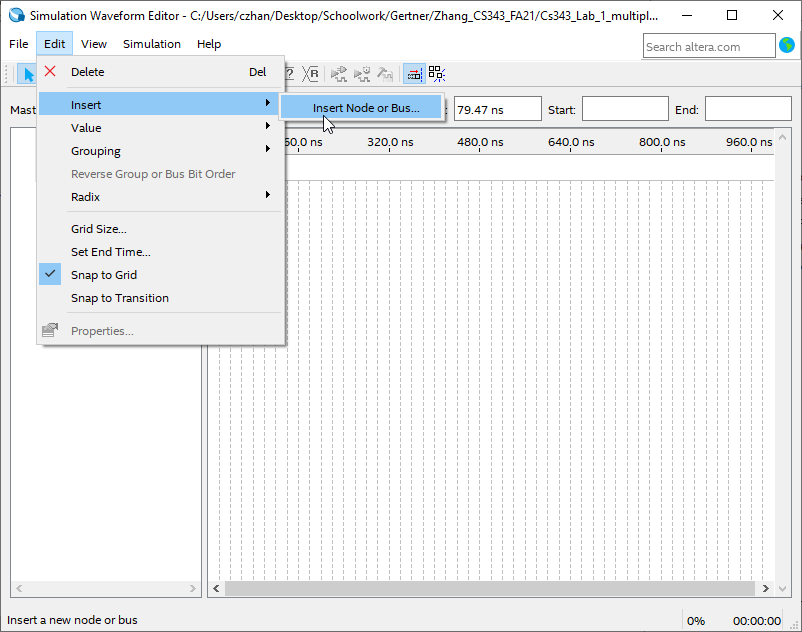




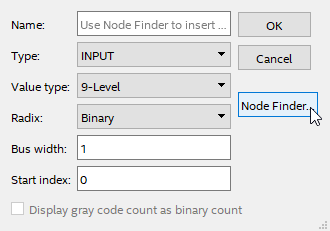
Next, I create a symbol file for the current file and in another block design file, I go into the symbols tool in the project dropdown, I can see the design for my 2:1 MUX. I press OK and put it in the new block design file with 3 inputs and 1 output under the names of IN\_A, IN\_B, IN\_S, OUT\_1 Respectively.



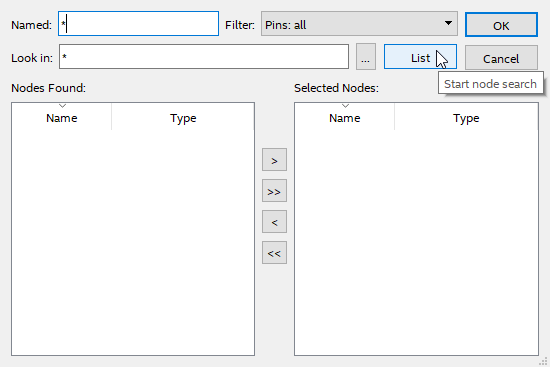
I make a new “University Program VWF” file

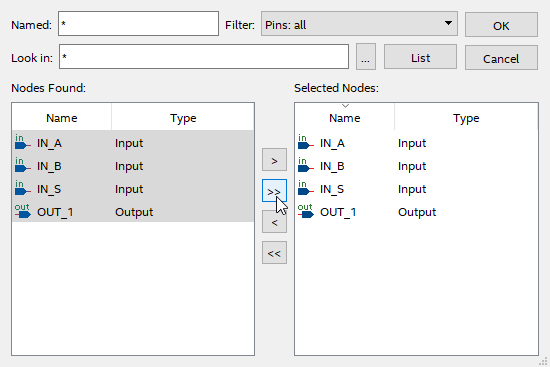


I press Edit -> Insert -> Insert Node or Bus and a pop up should show up.

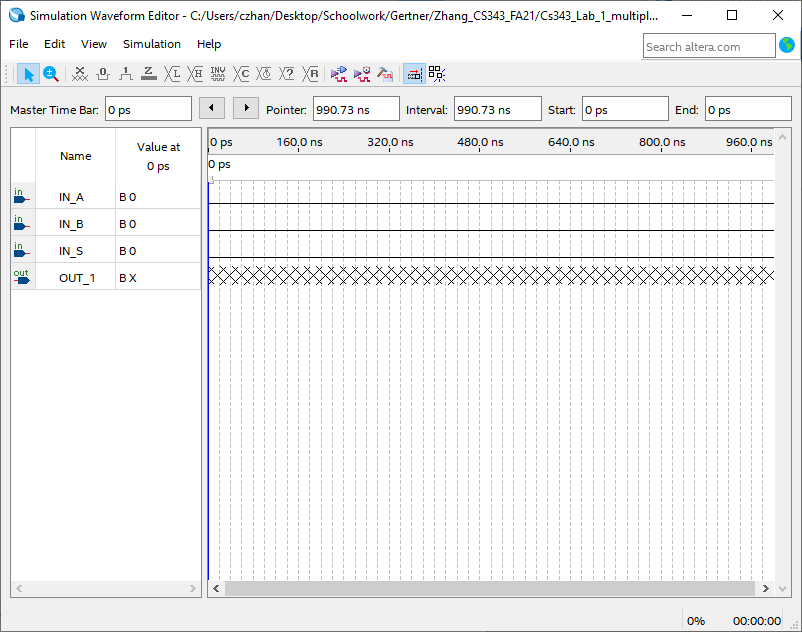


In the popup, press Node Finder and another popup should show up

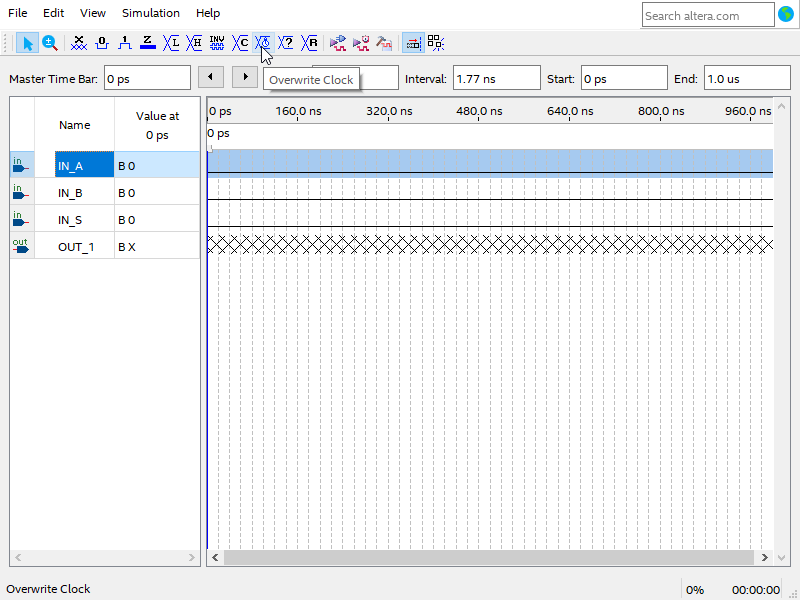




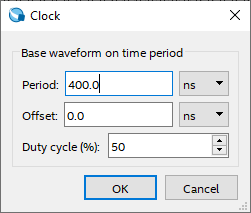
In the new popup, press list and a list on the left with all inputs and outputs should show up, then press the >> symbol to select all nodes and press OK.

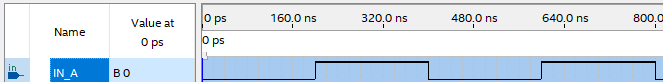


This is what you should see and next I will start manually inputting values for the inputs so that we can generate the appropriate output for the MUX.

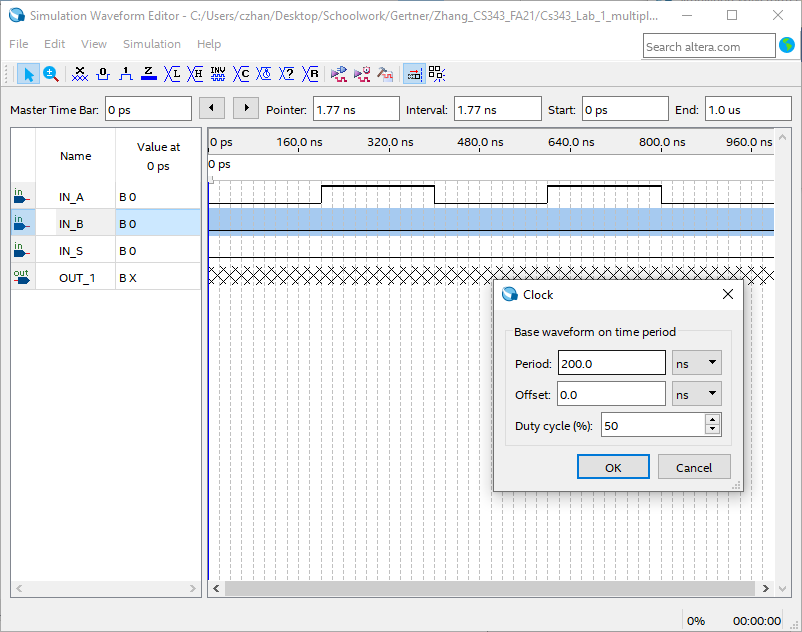


In the same file, I press on IN\_A -> overwrite Clock symbol which allows me to change the values of the input at a set interval. (A POPUP SHOULD SHOW UP)



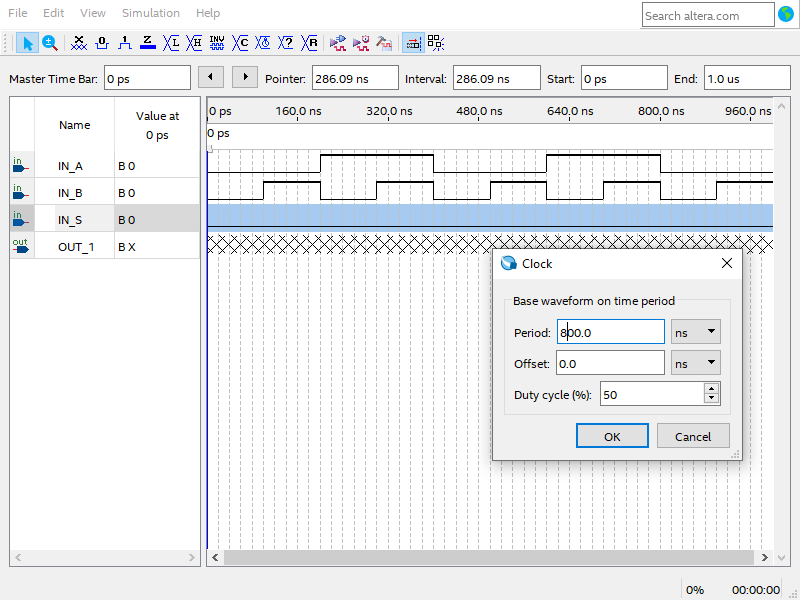


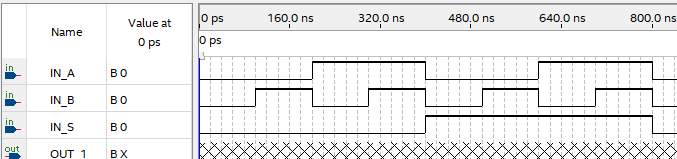
In the popup window, I set the period to 400 ns which makes it so that the value is at 1 at 200ns-400ns and 1 at 600ns-800ns.



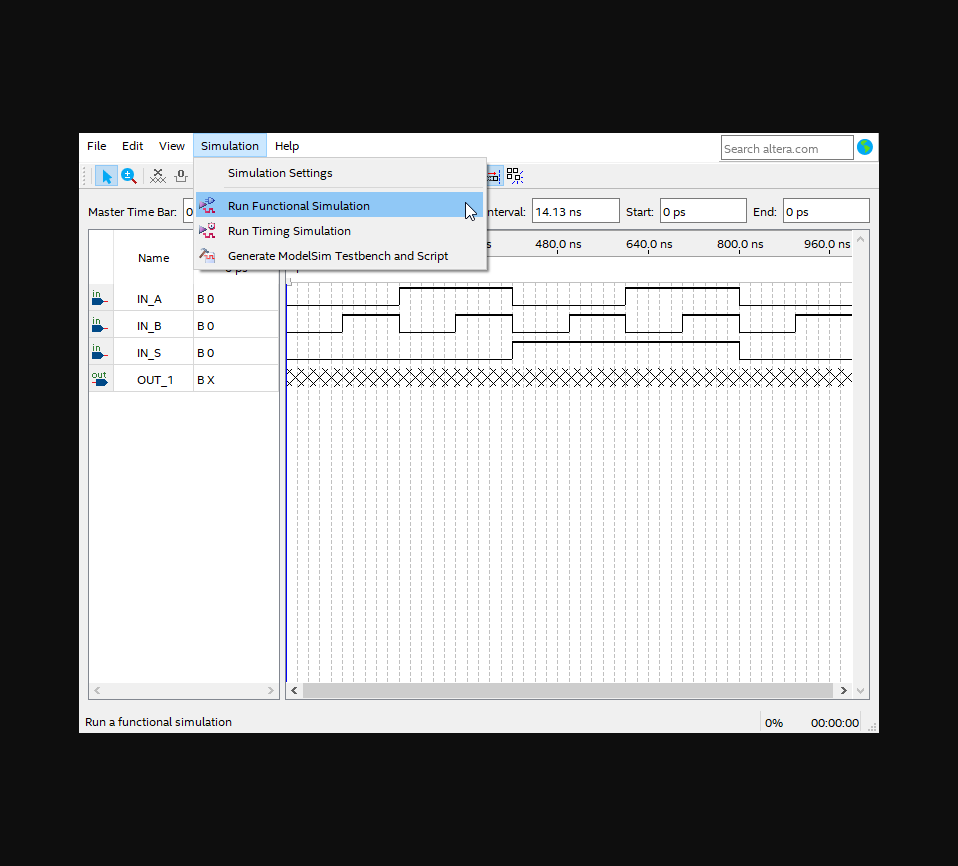


I repeat the same steps as above for IN\_B but instead of 400ns, I do 200ns.

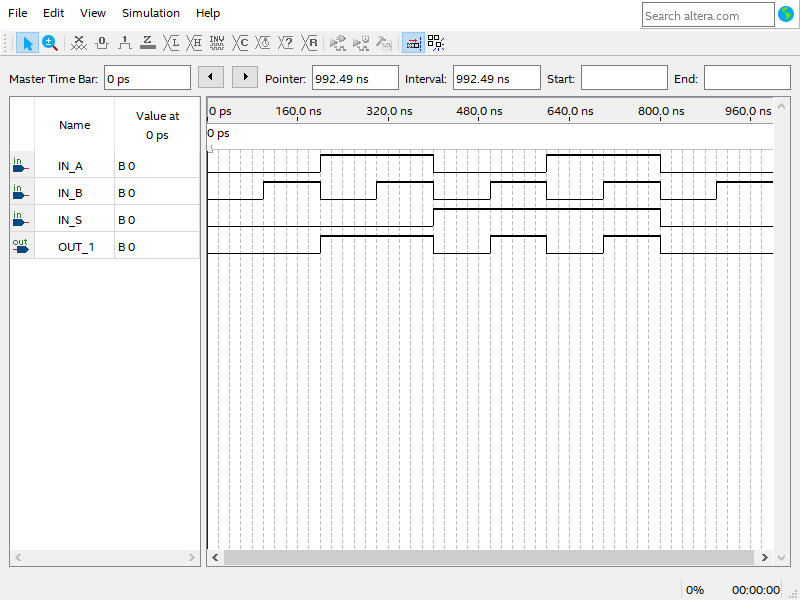


I repeat the same step again for IN\_S but with 800ns.

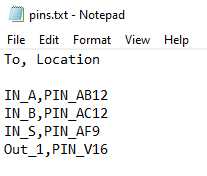
My WVF now looks like this and I can begin the simulation



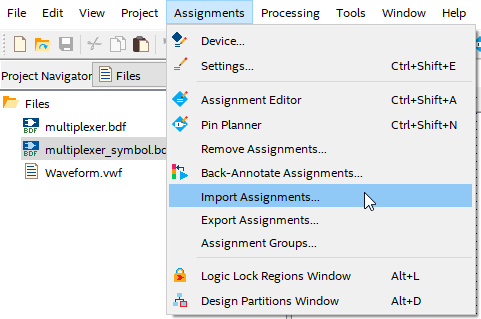
I go to the simulation drop down and press run functional simulation

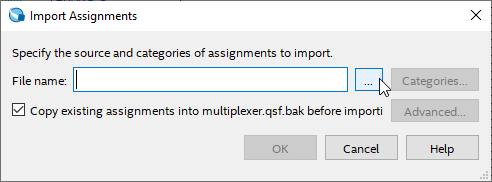


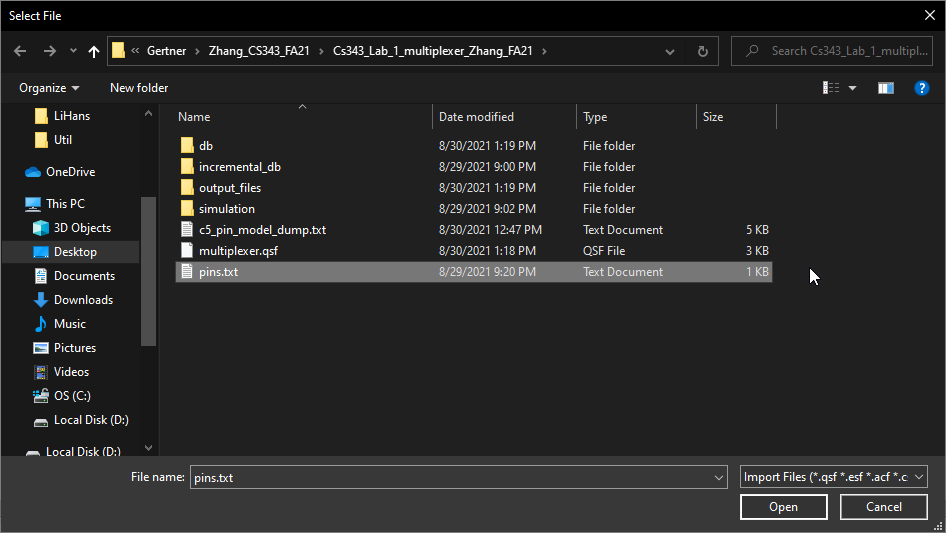
Afterwards, this should be the result. A WVF file with a proper line output instead of X’s.

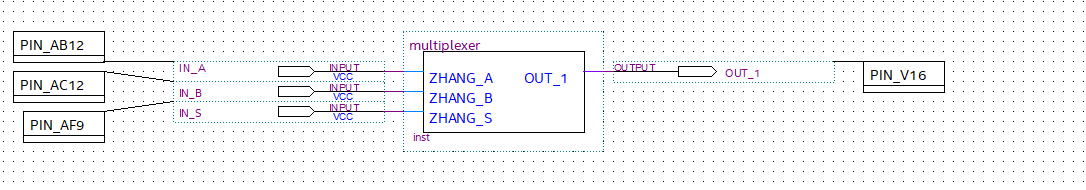
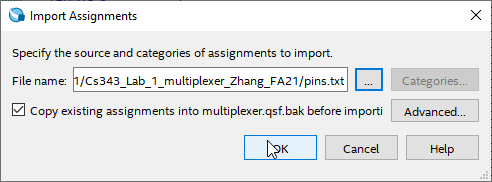


Next I create a text file with the naming convention “To, Location” where the To describes where the pin is assigned to and which pin to assign.



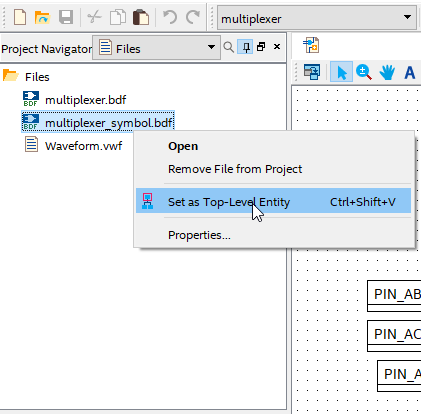




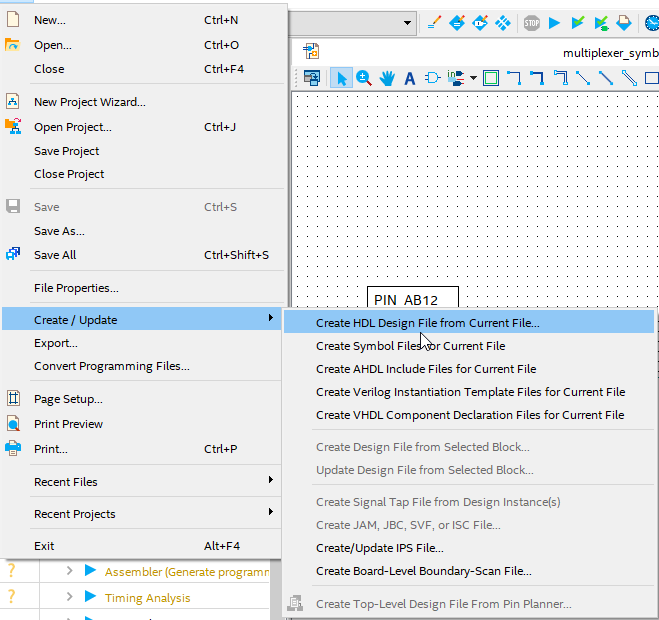


The next step is to go to the assignments and import assignments. Then press the … when the popup shows which should allow you to select a file. Select the pins files that was created before and the output should show in the design file.

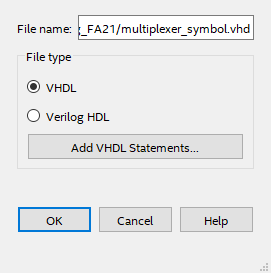
Task 3



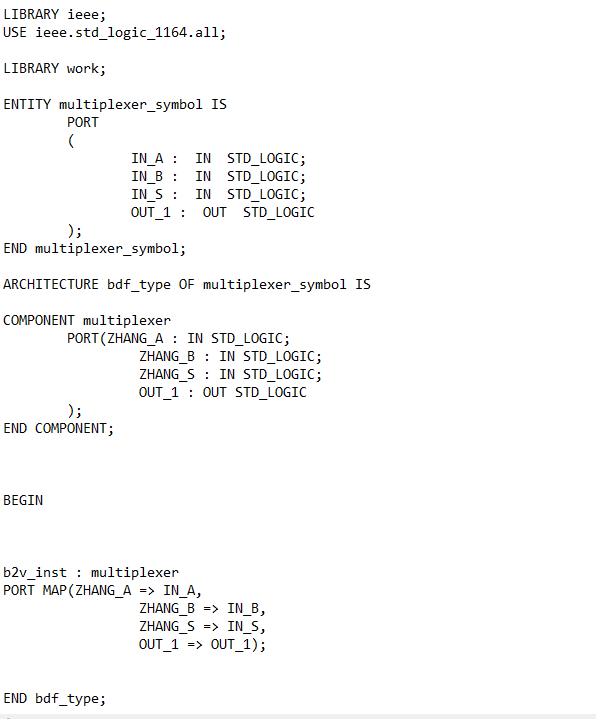
It is important to set the correct top level entity, in my case it is the multiplexer\_symbol.bdf which is basically the MUX as a symbol.



Next go to files -> create/Update -> create HDL Design File from Current File…



A popup should show looking like this. Make sure the file type is in VHDL and not Verilog HDL



A vhd file should be generated now in your directory. Go into your directory and open the vhd file. I chose to use notepad to observe my vhd code.