CSUS Tutorial for

Altera: Quartus II Web Edition

with Terasic DE0-Nano

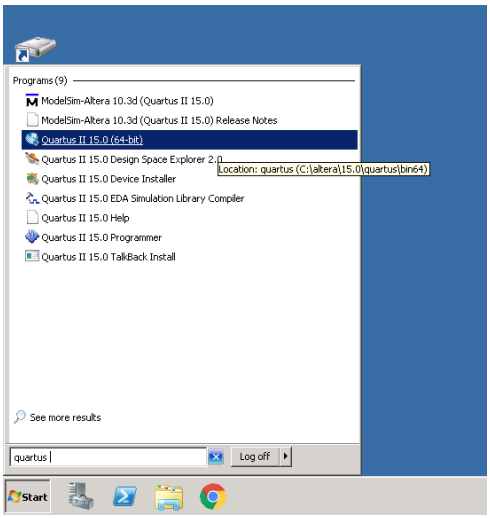
This tutorial assumes you are using an ECS Lab computer or your own computer that has had Altera Quartus II web edition installed properly and the IDE has recognized your JTAG Programmer.

Table of Contents

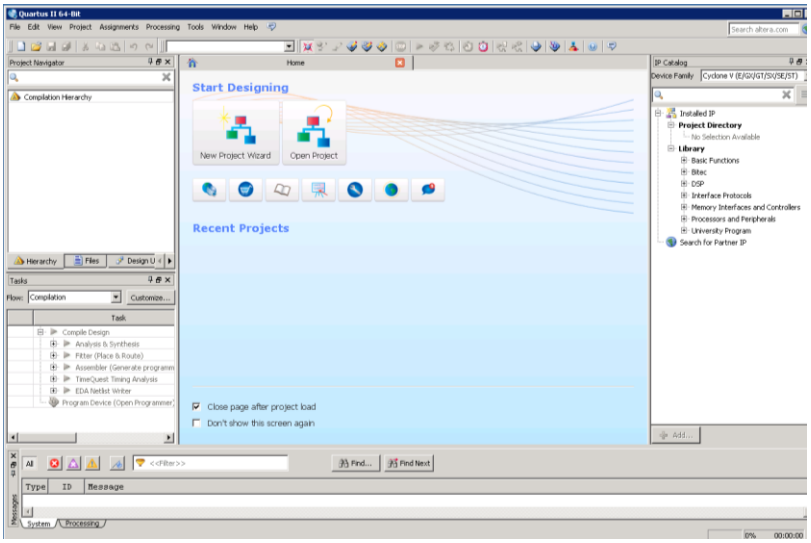
Getting Started with Quartus II

Creating a New Project

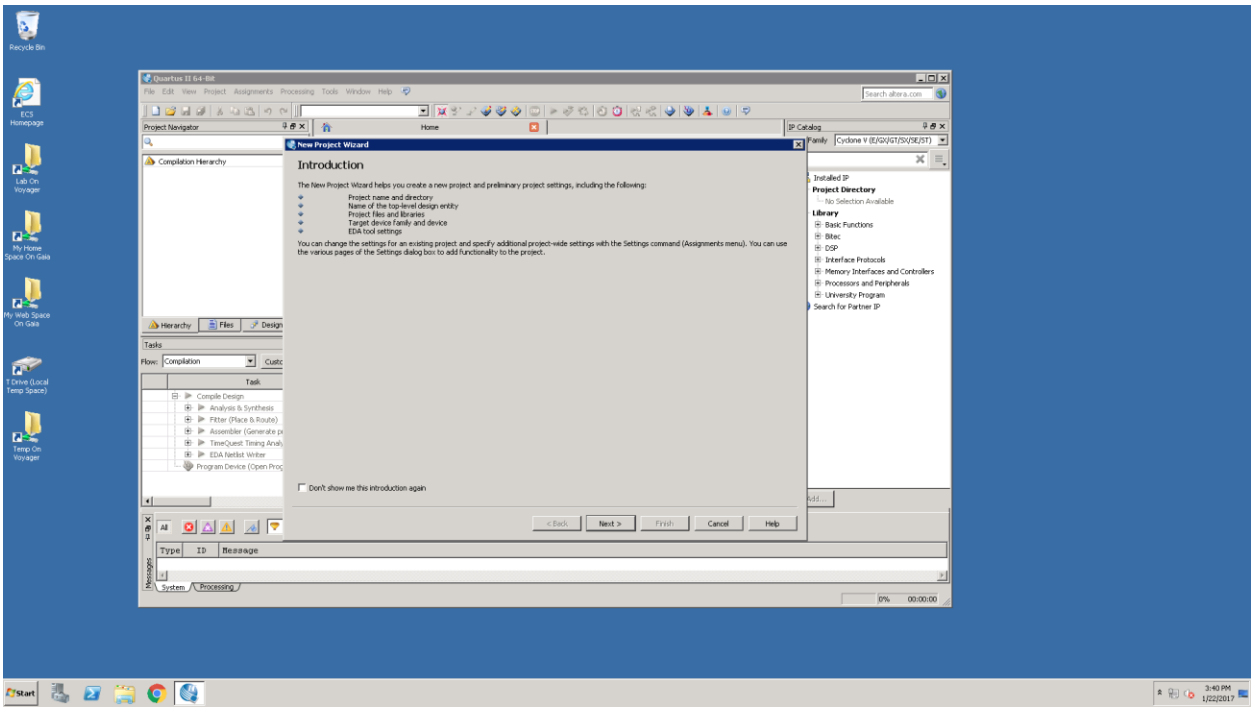
From the start menu type ‘quartus’ then select ‘Quartus II 15.0’



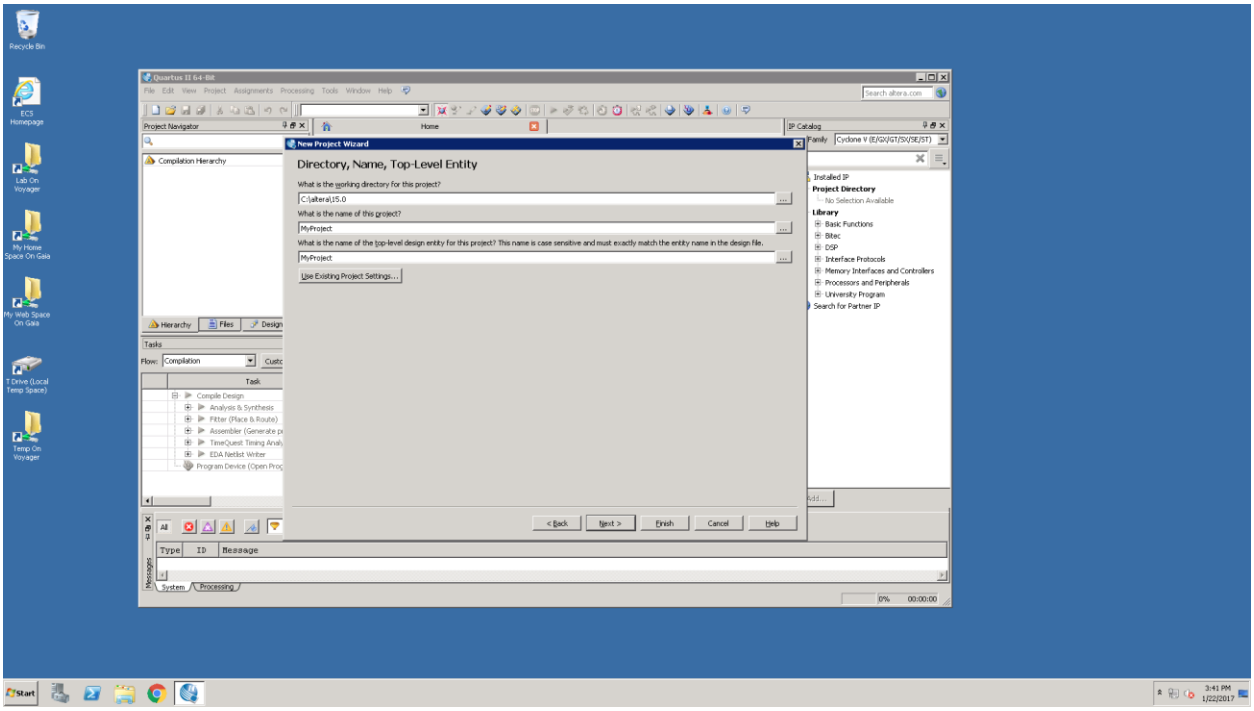
After the software opens click ‘’New Project Wizard”



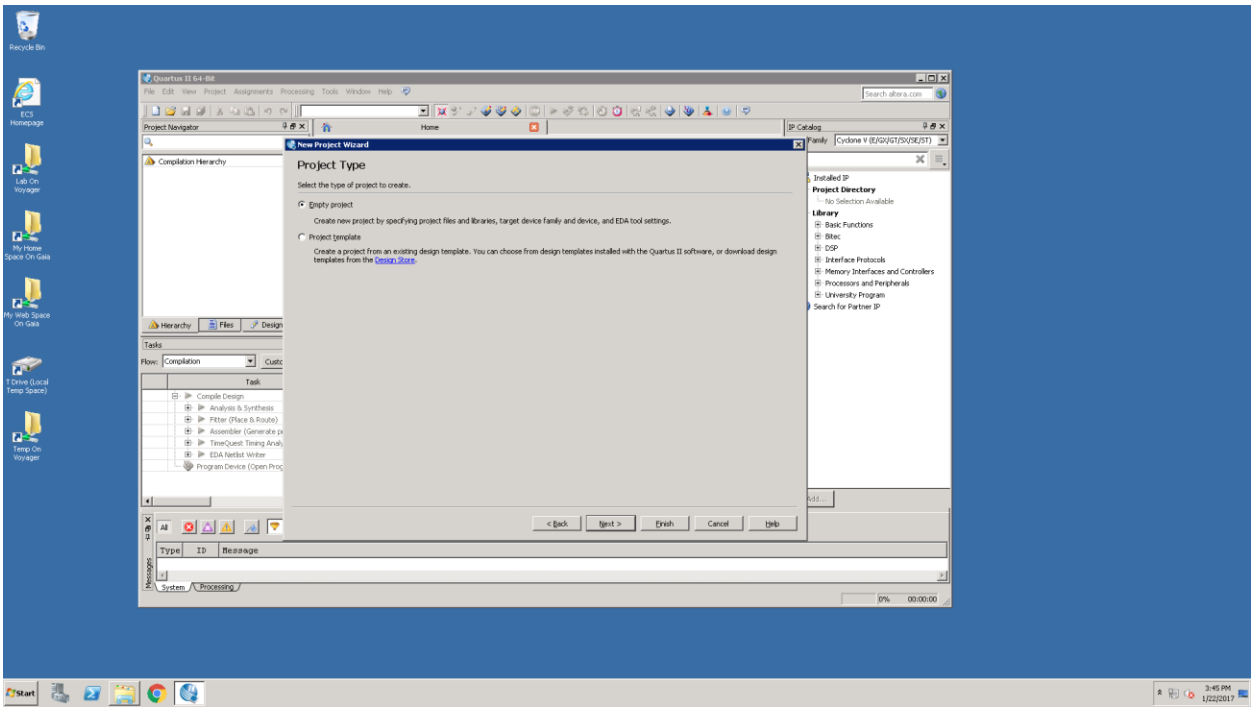
Then go through the following steps.

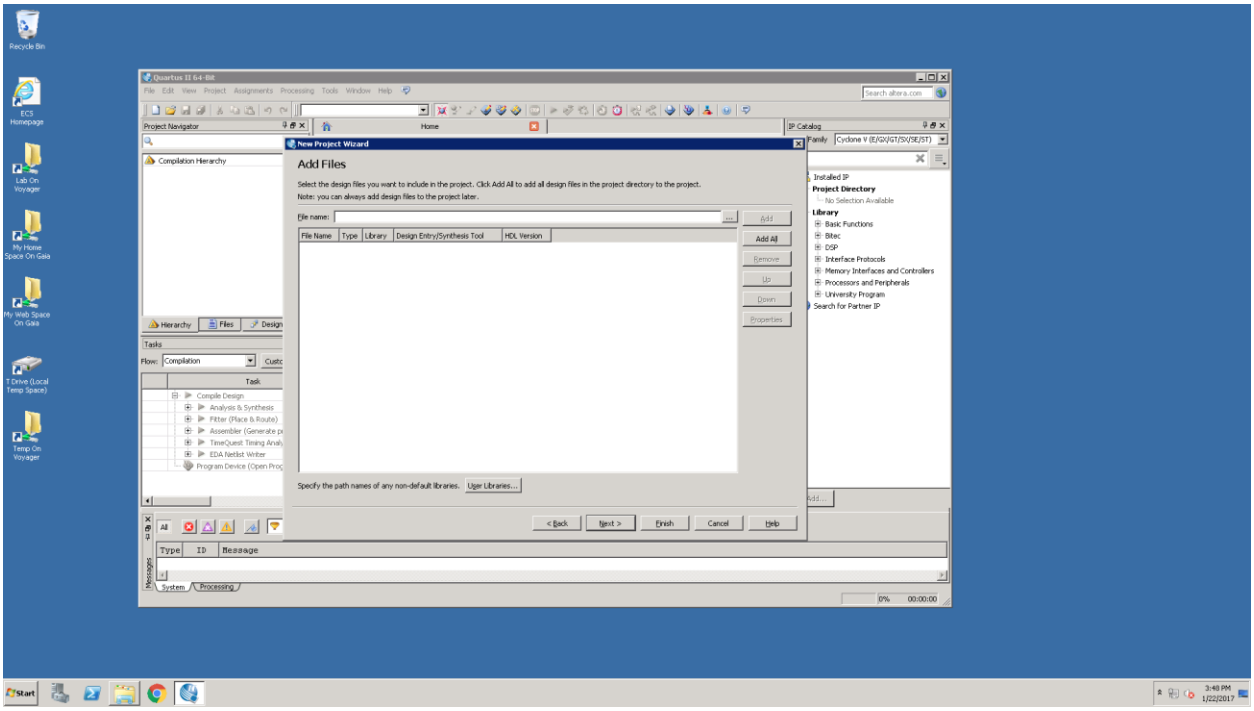


Create a name for the project. *Note: It is recommend that you save your project on a flash drive or on "My Home Space on Gaia "else it maybe erased when logging out of your ECS account. Also save the project in a new folder.*

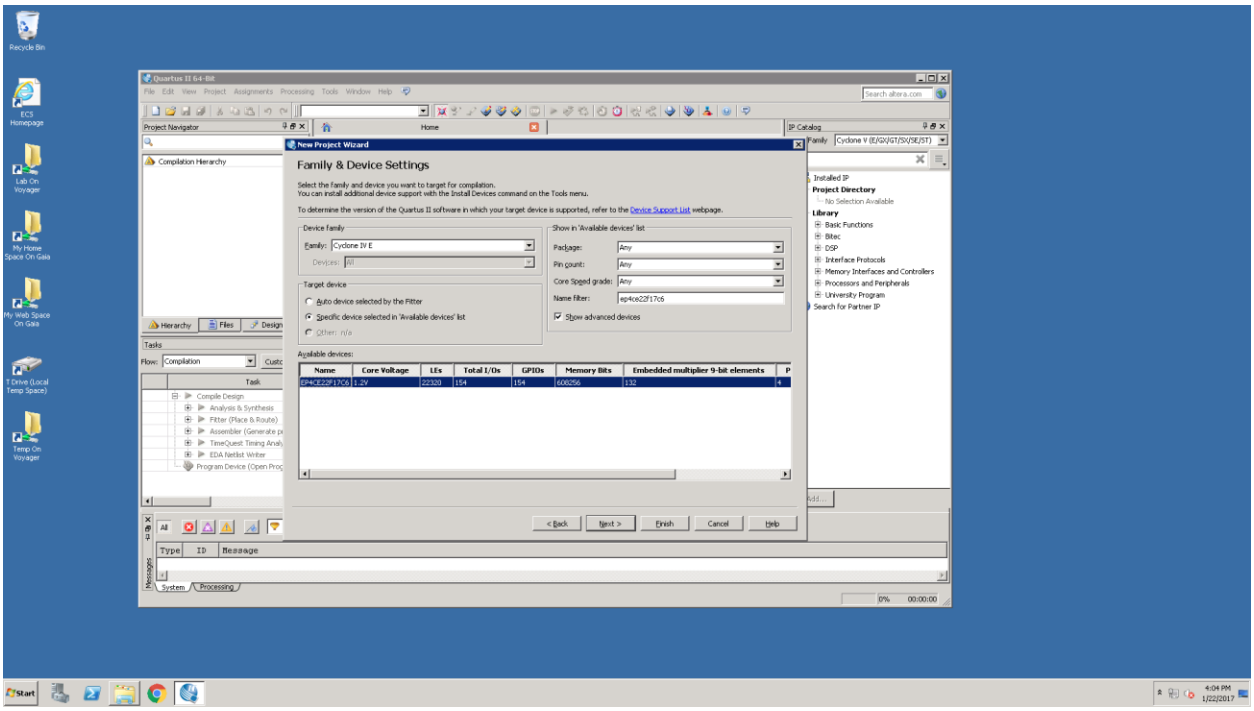


Select "Empty Project' and then hit next.

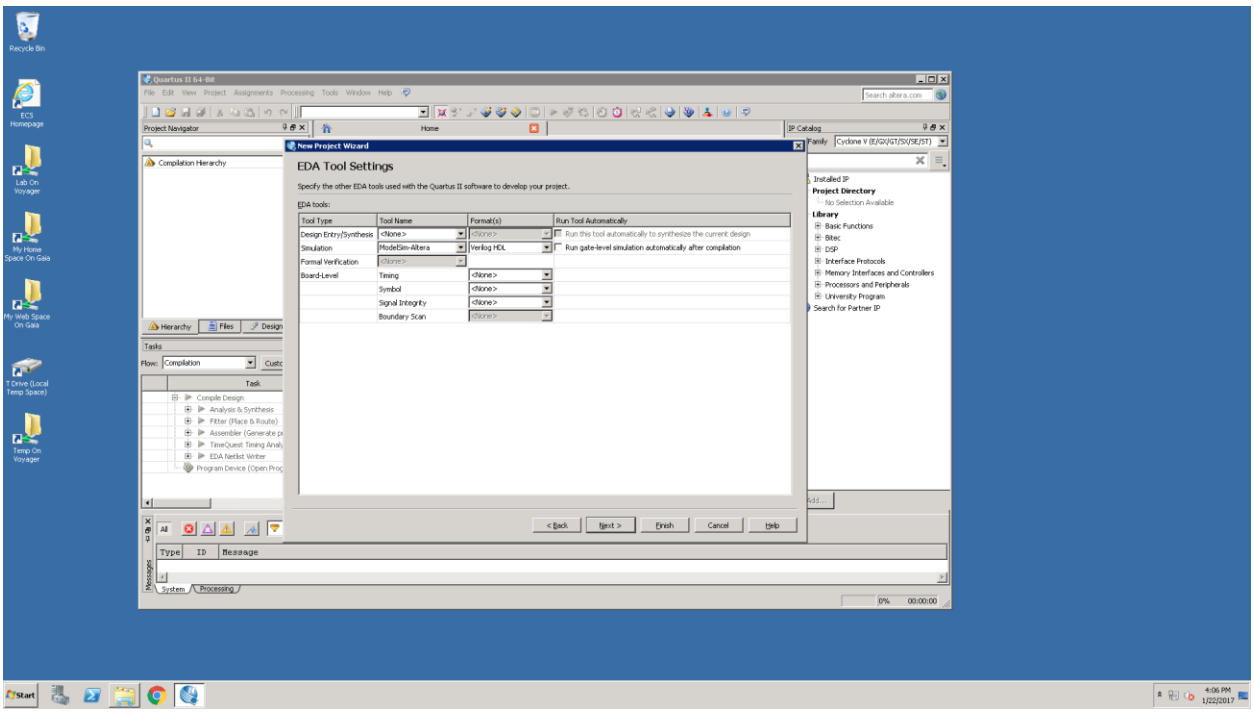




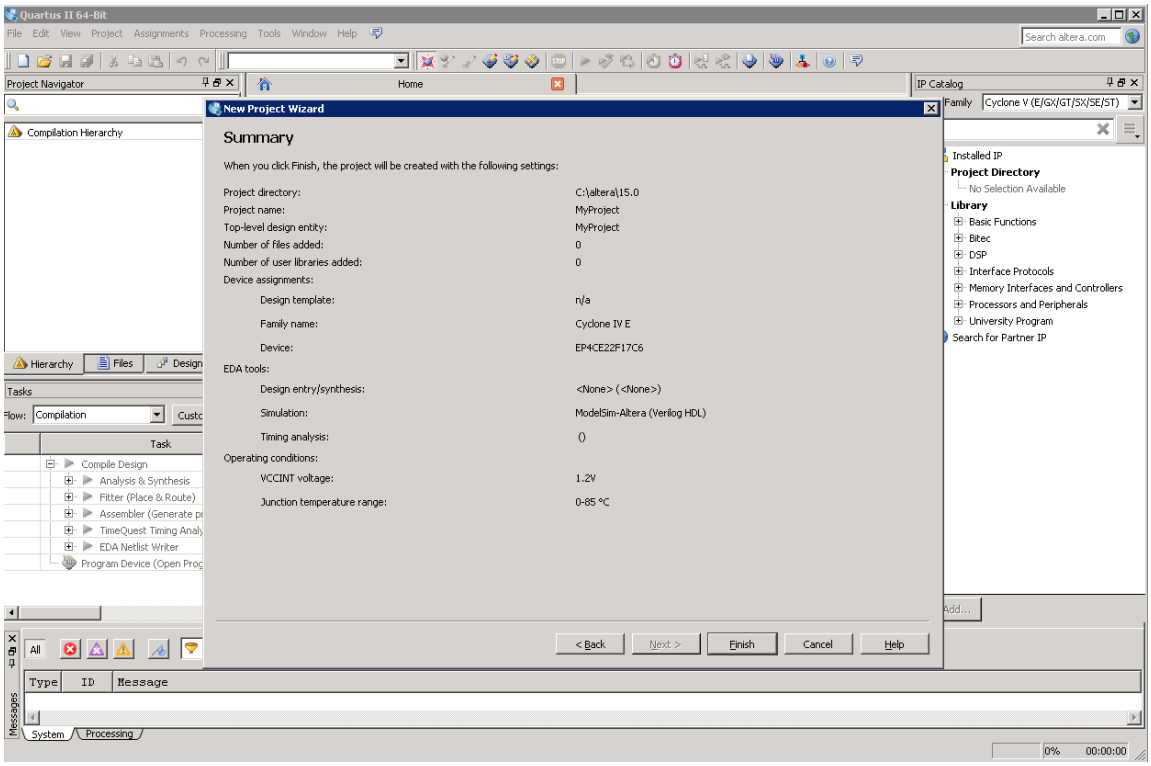
Now in the Family box select ‘Cyclone IV E’ and type ‘ep4ce22f17c6’ into the name filter. Then select from the list below, and hit next.



Make sure the fields match the same as the image below. Then select next

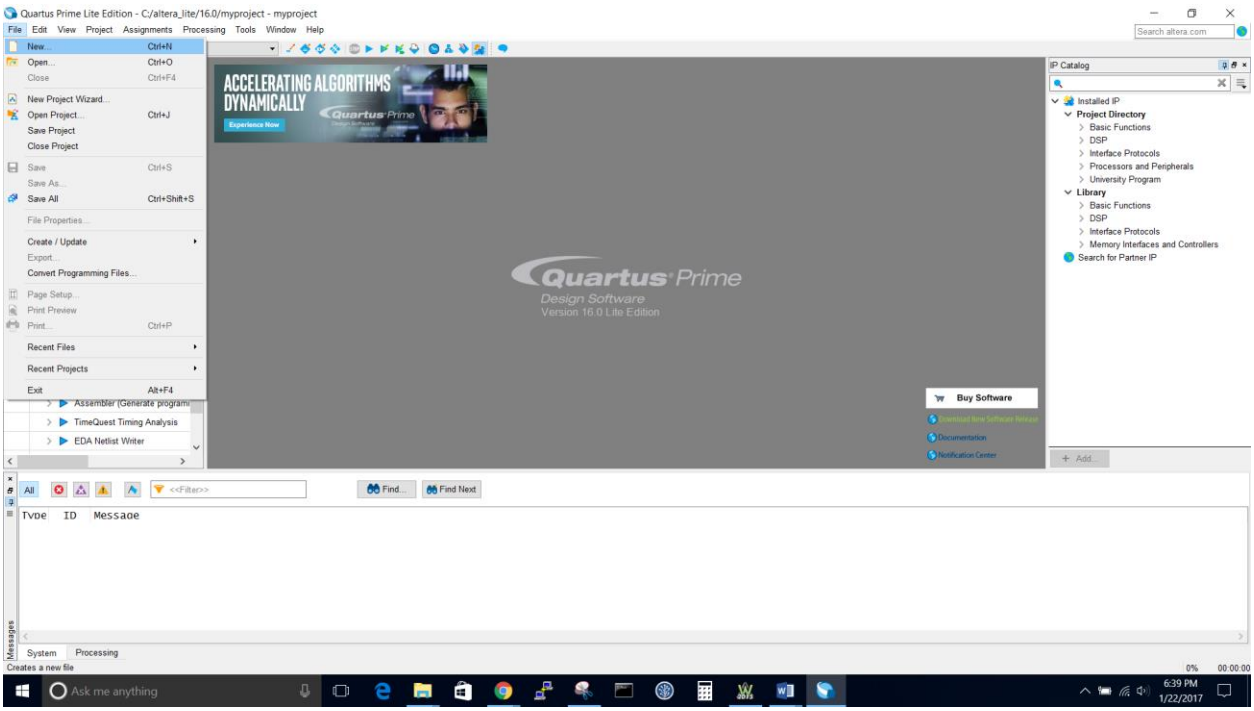


Confirm that all the information is correct then select Finish.

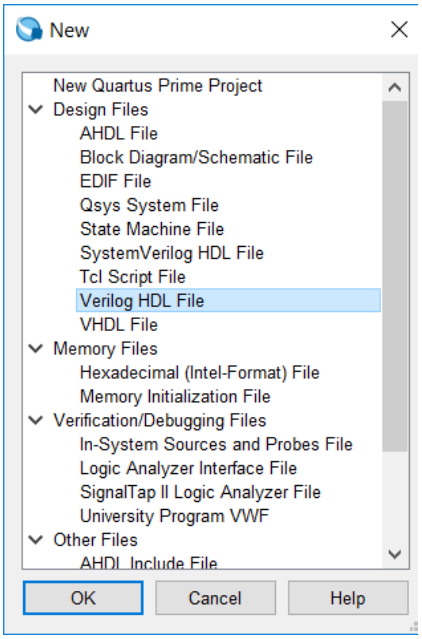


Create A New File

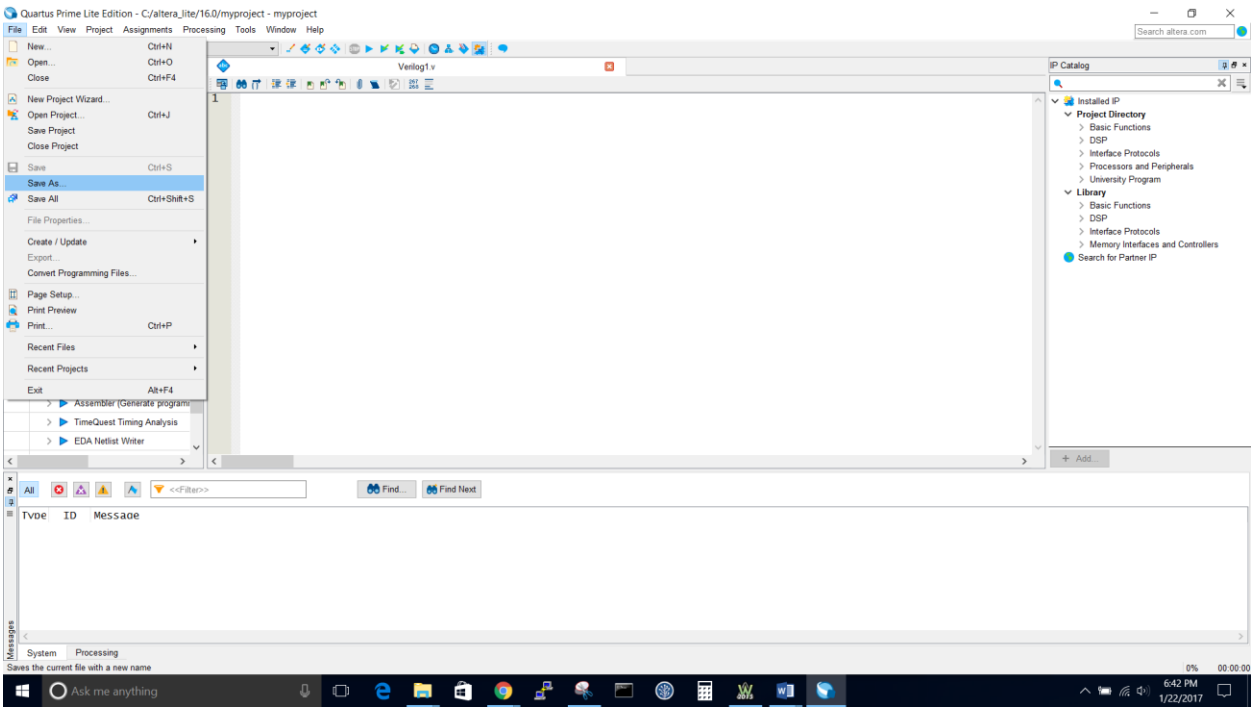
Now that the project is created, next is to create new Verilog file. Go to File < New



Next select ‘Verilog HDL File’

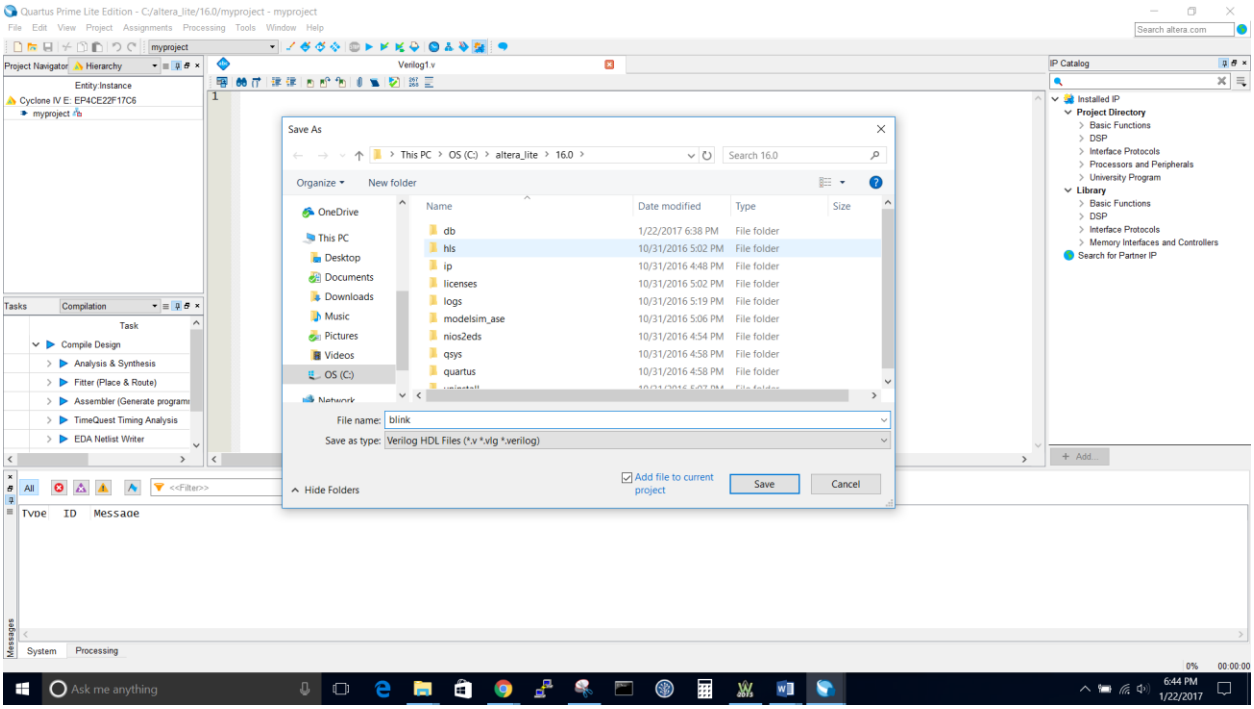


A blank document will open. Next we are going to save the file



Create Blink File

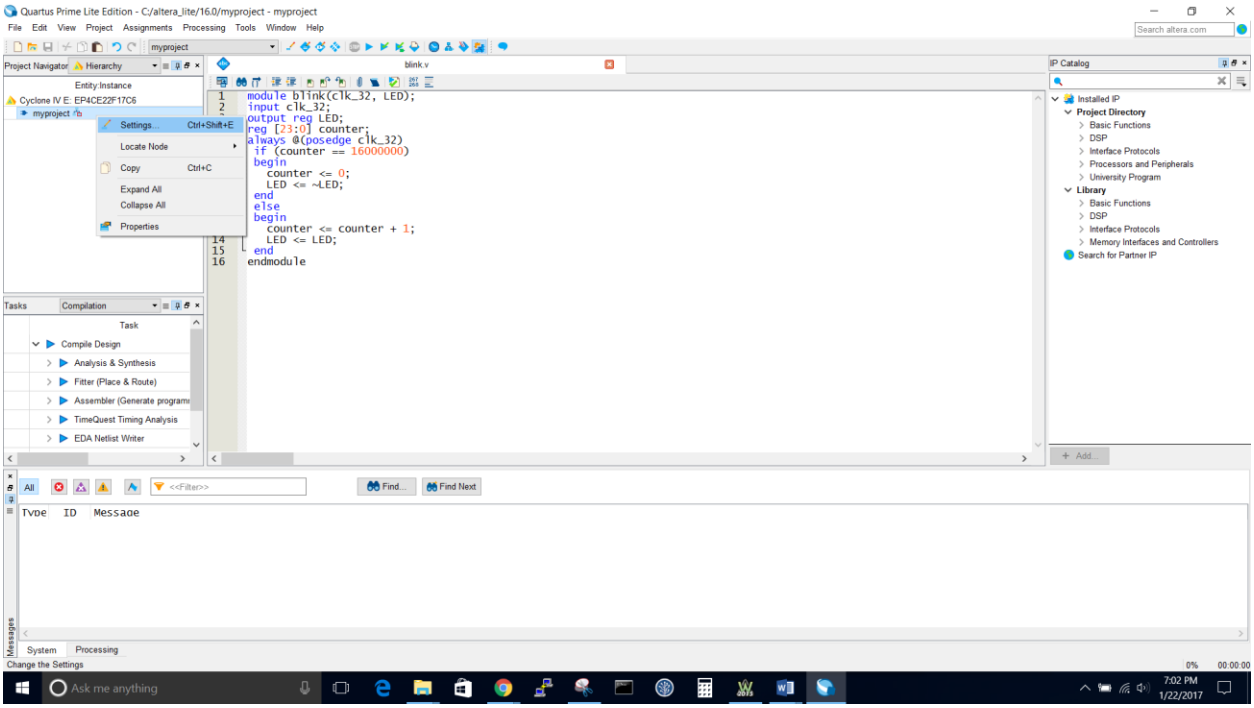
Then save the file name as “blink”



Then save. Next we copy and paste the following Verilog code into the document and save it.

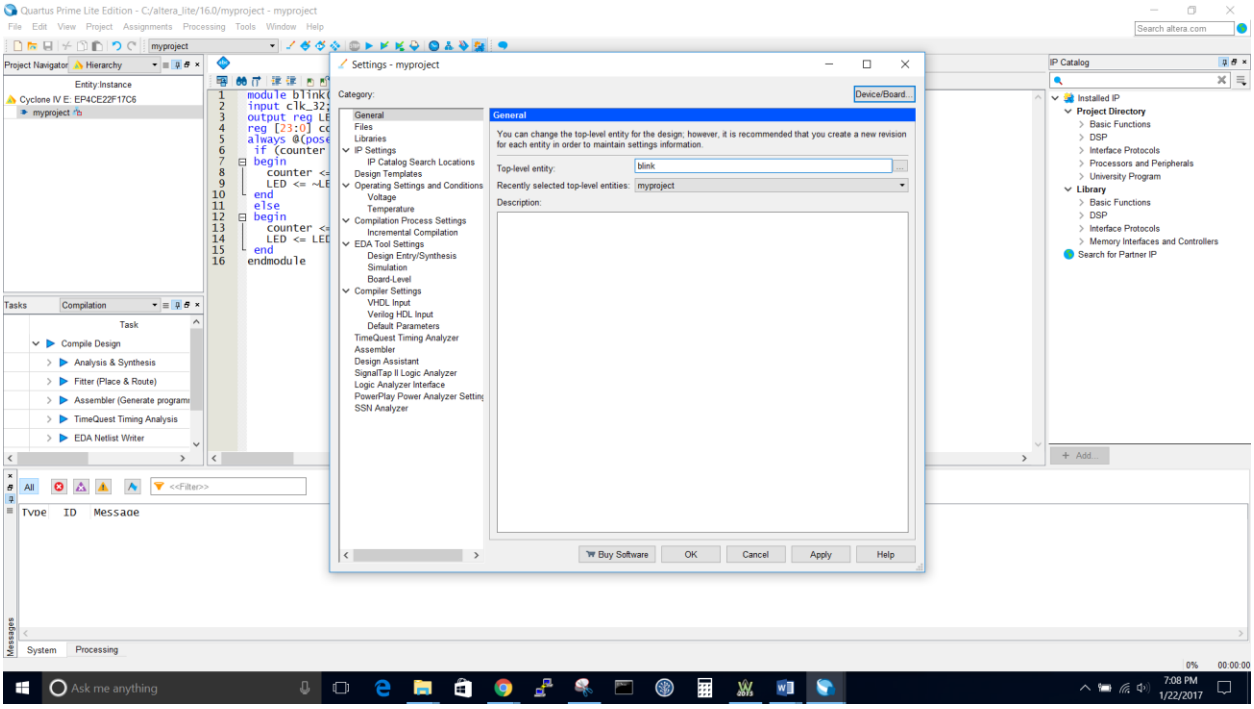
```
module blink(clk_50, LED);
input clk_50;
output reg LED;
reg [23:0] counter;
always @(posedge clk_50)
 if (counter == 16000000)
 begin
            counter <= 0;
            LED <= ~LED;
 end
 else
 begin
            counter <= counter + 1;
            LED <= LED;
 end
endmodule
```

Next we have to make sure that the blink module is the top level module. Go to the top left pane and right click the file name below cyclone IV. Then select settings as seen in the image below.

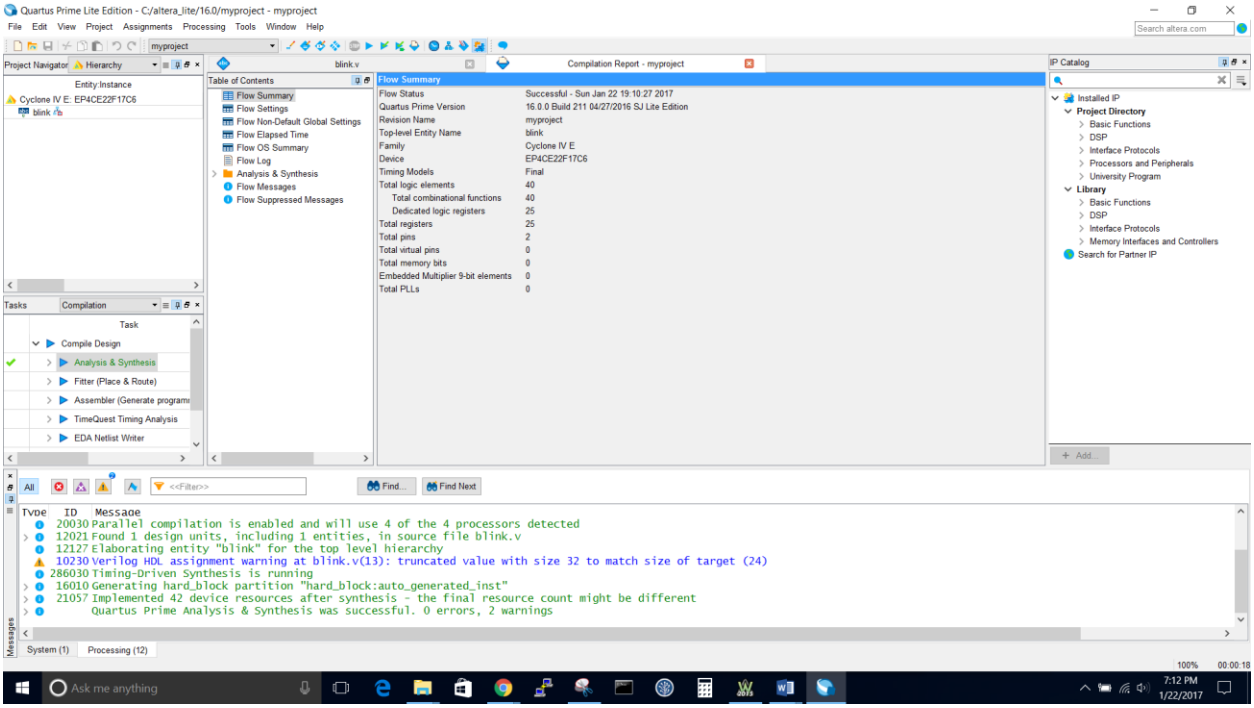


Change Top Level Entity

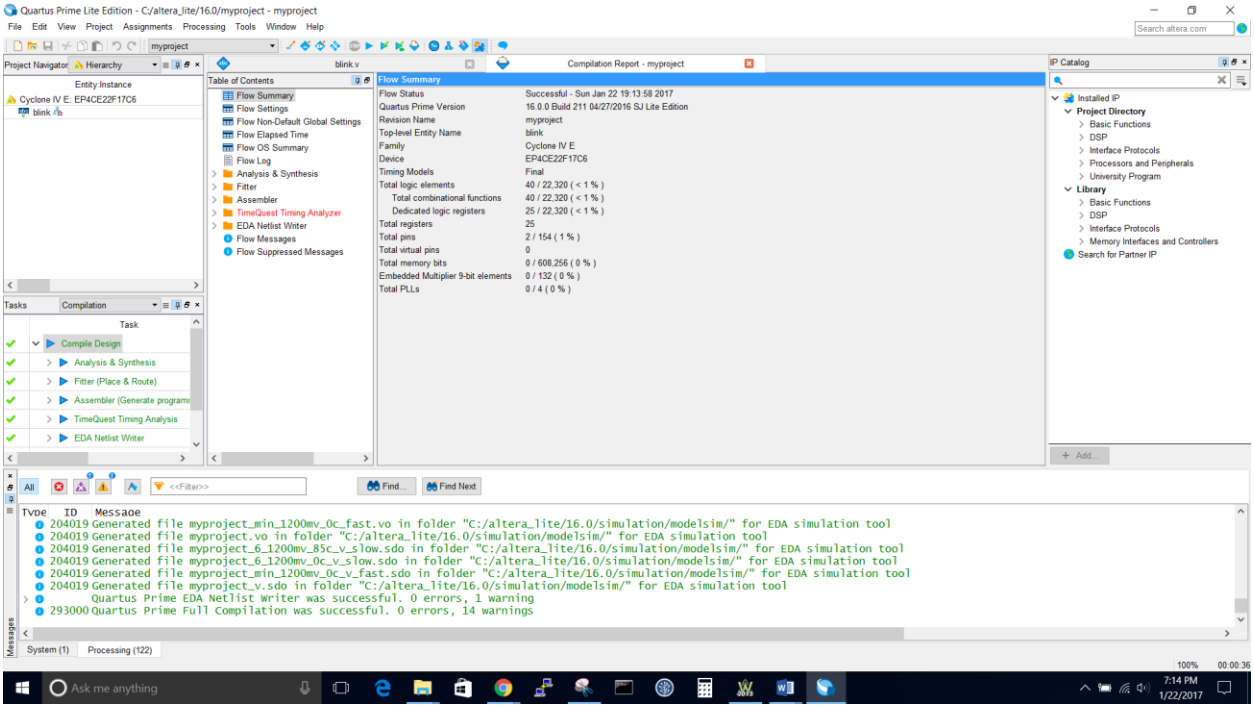
Then select the General tab on the left, and change top level entity to ‘blink’



Then double click ‘Analysis & Synthesis’. This will check for syntax. You should see the following results.

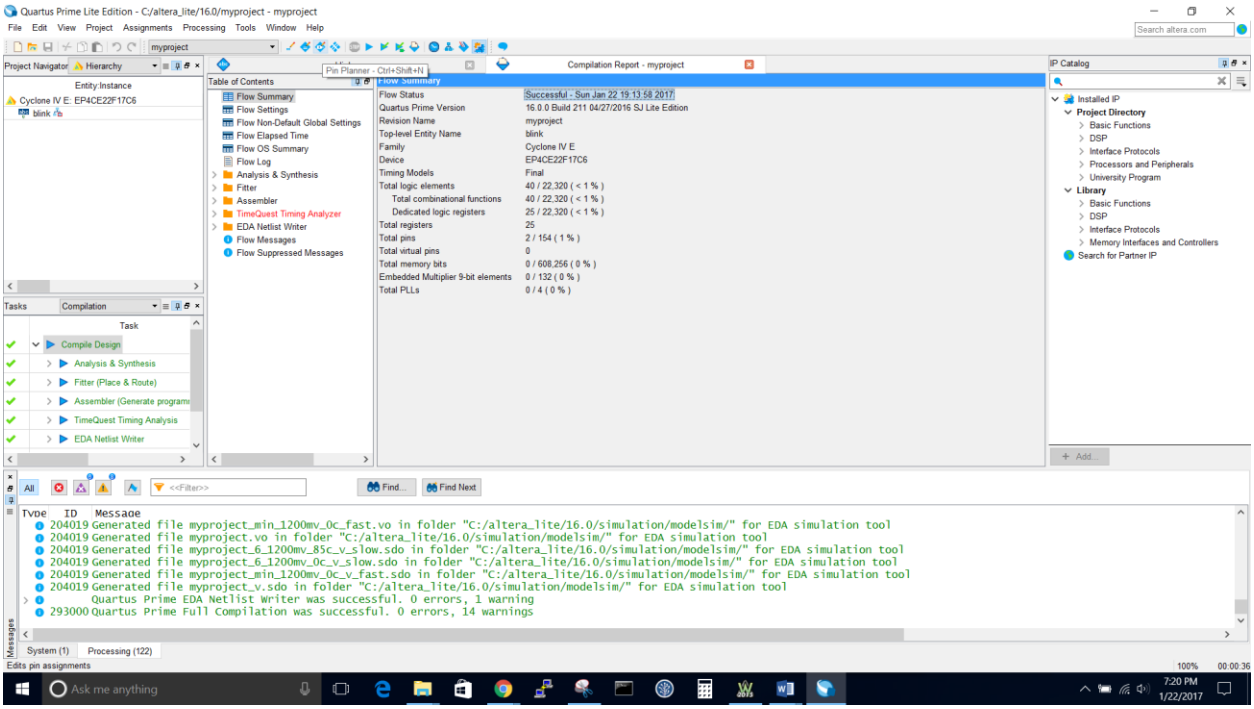


Then compile the whole design. Double click ‘Compile Design’. You should see the following results

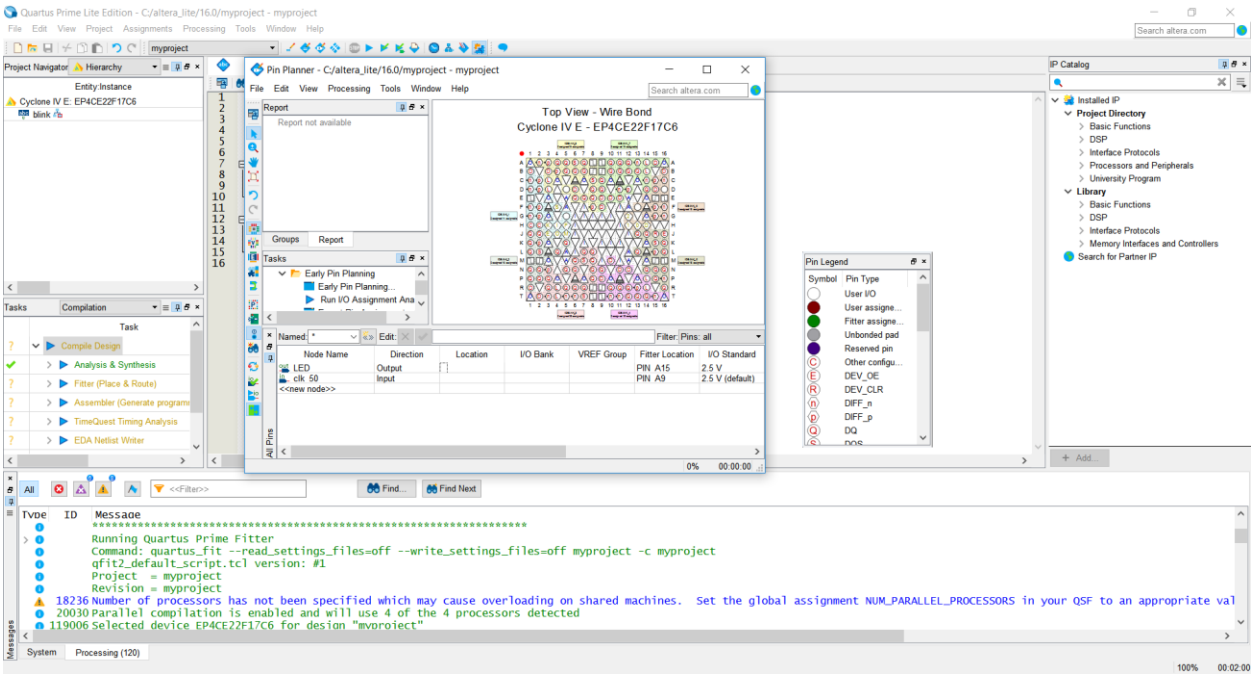


Assign Pins

Next to assign pins to the FPGA. Next open the pin planner.



The following window should appear



To find what pins to assign to the IO open the DE0-Nano user manual found here.

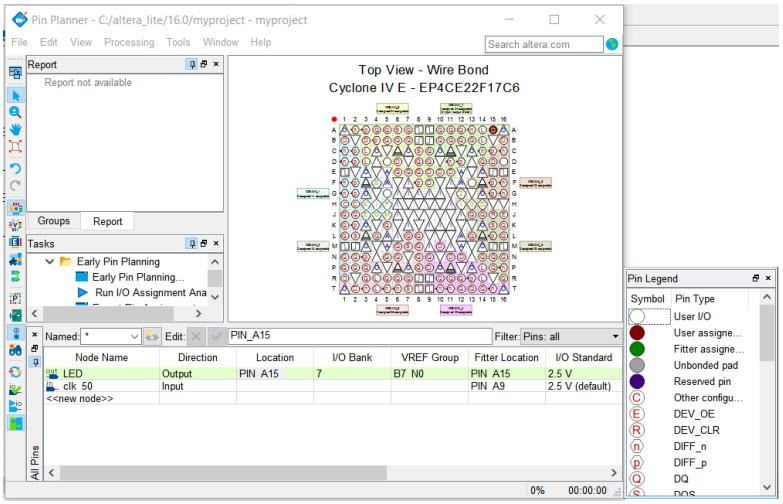
https://www.parallax.com/sites/default/files/downloads/60056-DE0-Nano-User-Manual-v1.07.pdf

In the general user input output section of the user manual. Looking at the table below choose a LED and put the pin value in to the pin planner in the LED output.

Table 3-2 Pin Assignments for LEDs

Signal Name	FPGA Pin No.	Description	I/O Standard
LED[0]	PIN_A15	LED Green[0]	3.3V
LED[1]	PIN_A13	LED Green[1]	3.3V
LED[2]	PIN_B13	LED Green[2]	3.3V
LED[3]	PIN_A11	LED Green[3]	3.3V
LED[4]	PIN_D1	LED Green[4]	3.3V
LED[5]	PIN_F3	LED Green[5]	3.3V
LED[6]	PIN_B1	LED Green[6]	3.3V
LED[7]	PIN_L3	LED Green[7]	3.3V

Selecting the LED[0] put PIN_A15 into the location column.



Next to assign the clock pin we look up the pin assignment for the 50M Hz on board clock in the DE0 Nano user manual. Found on page 23 you should find the following figure below.

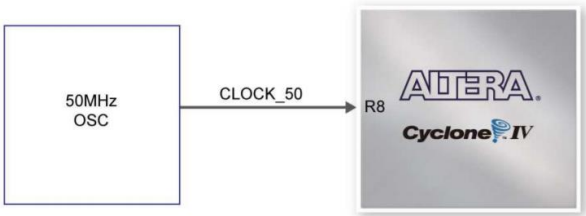
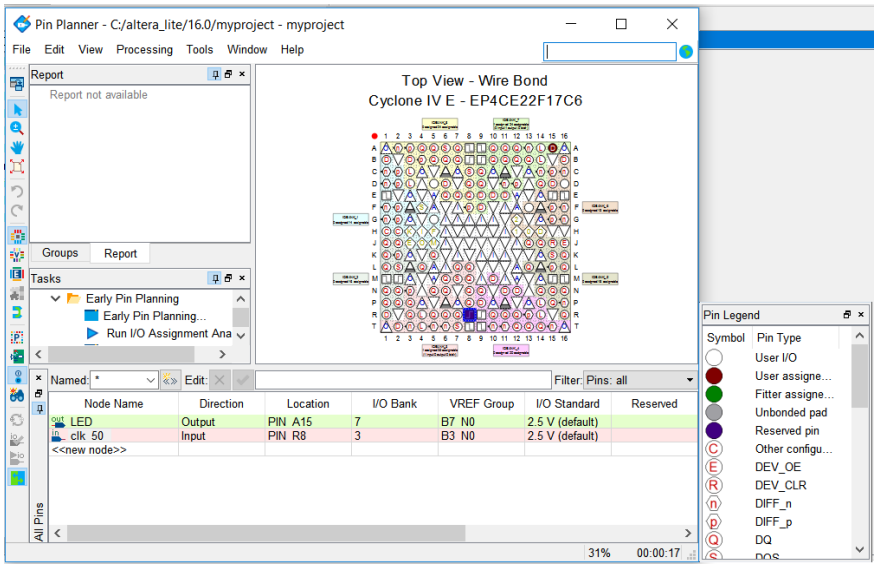
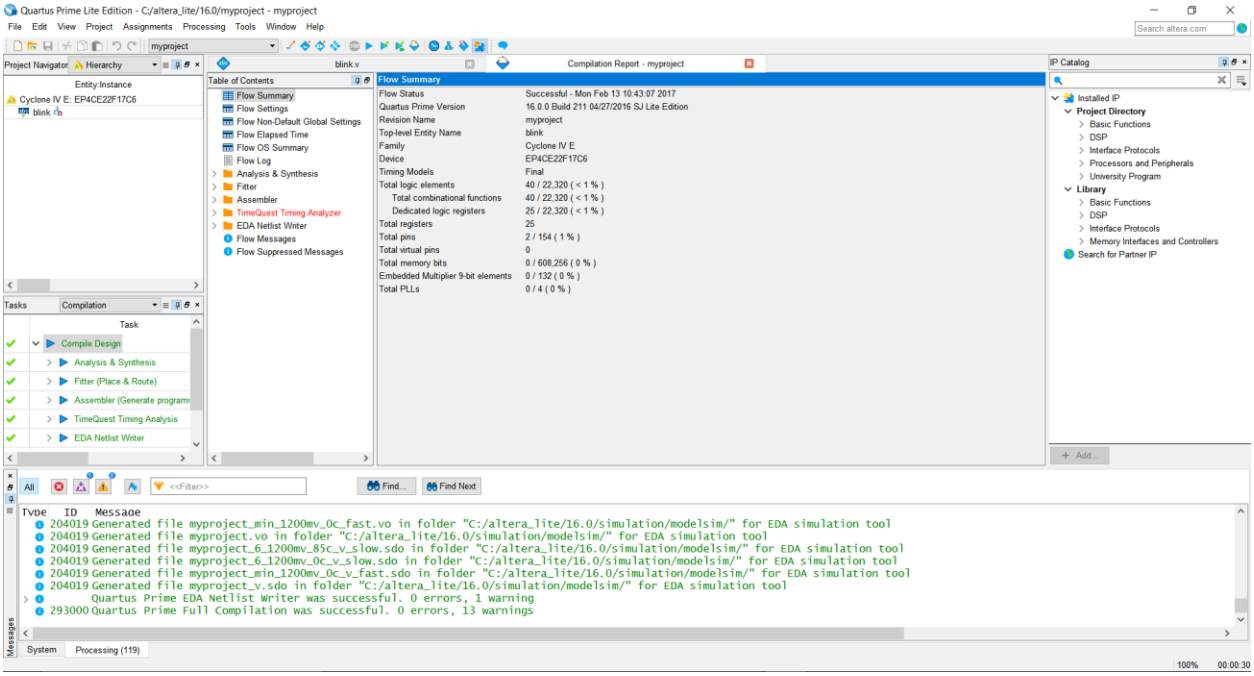


Figure 3-14 Block diagram of the clock distribution

Then assign PIN_R8 to the clk_50 pin.



Then recompile once you have assigned the pins.

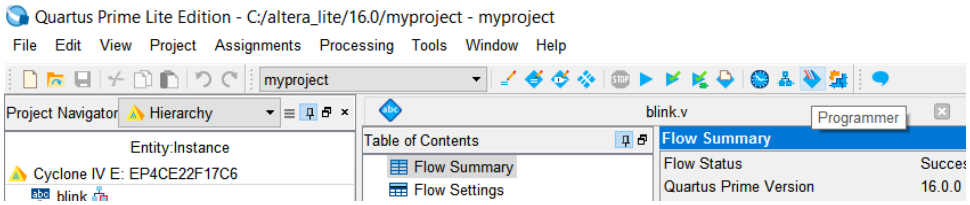


Programming the FPGA

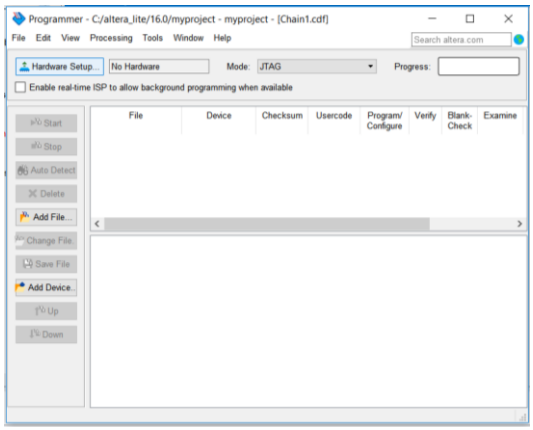
For this next part if you are using your own laptop you may need to install the device drivers, if you are using a lab computer you may skip this step. To install the driver, follow the steps in the video linked below:

[Install DE0-Nano Device Driver]

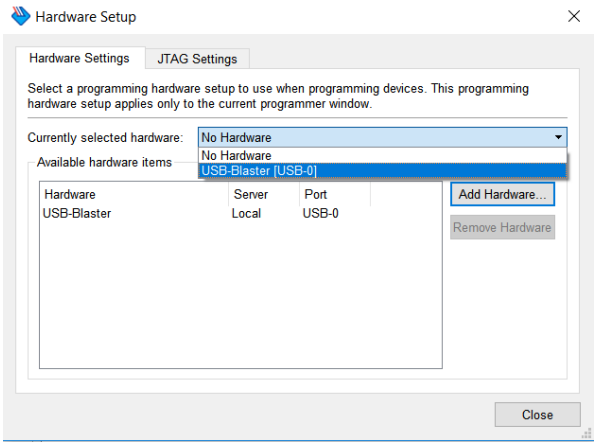
Next to program the FPGA we need to open the programmer.



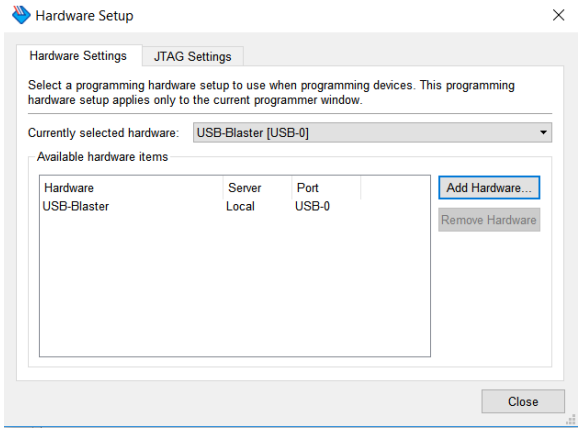
Once you select the programmer a new window should appear. Select “Hardware Setup”



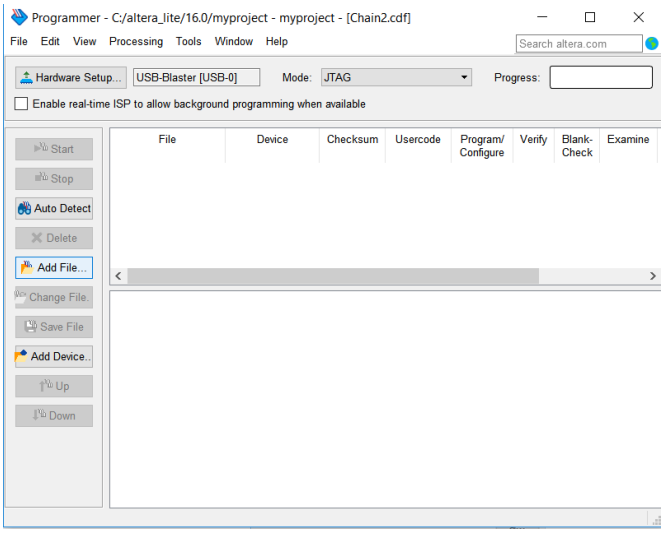
Once selecting a hardware setup a new window should appear.



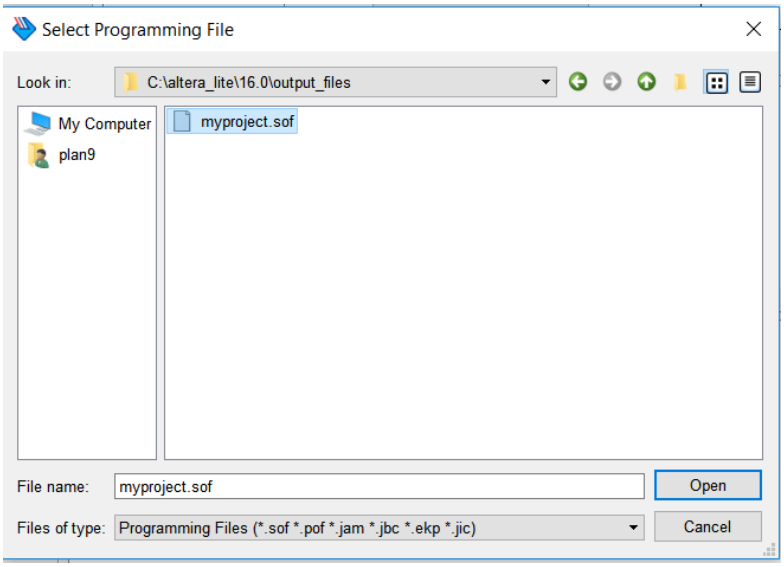
After the new window appears click the drop down box and select the USB Blaster [USB-0]. Then hit close.



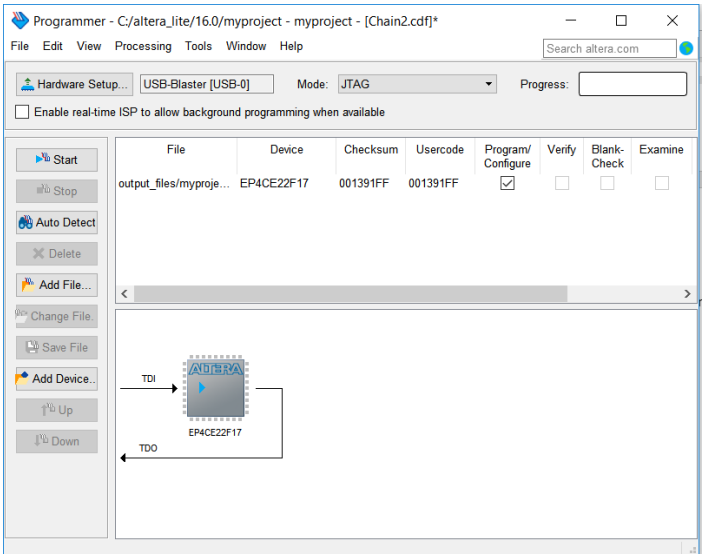
Then you should go back to the programmer window as seen earlier, but USB-Blaster in Hardware Setup. Next click “Add File”



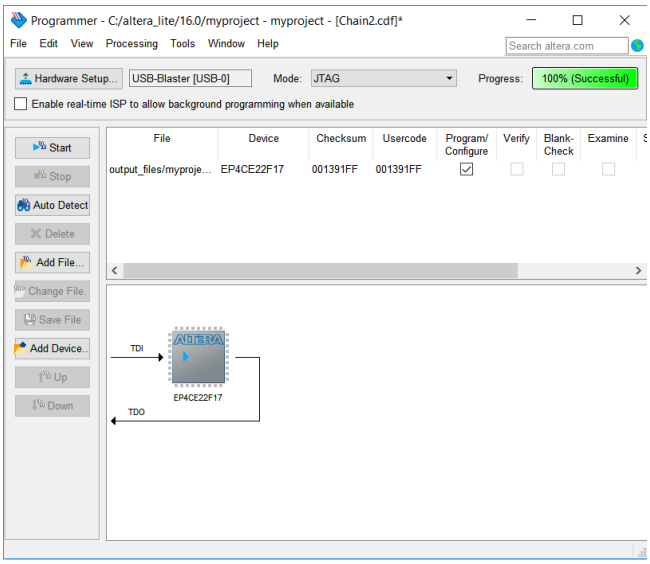
A new window should appear with the files in you project. Select output_files then the .sof file in that folder.



Click Open and you should see the following changes in the programmer window



Click start. If you your program successfully programmed the Progress bar should be green and say 100%.



You should also see one of the LEDs on the FPGA blink.

Creating a Test bench and Waveform

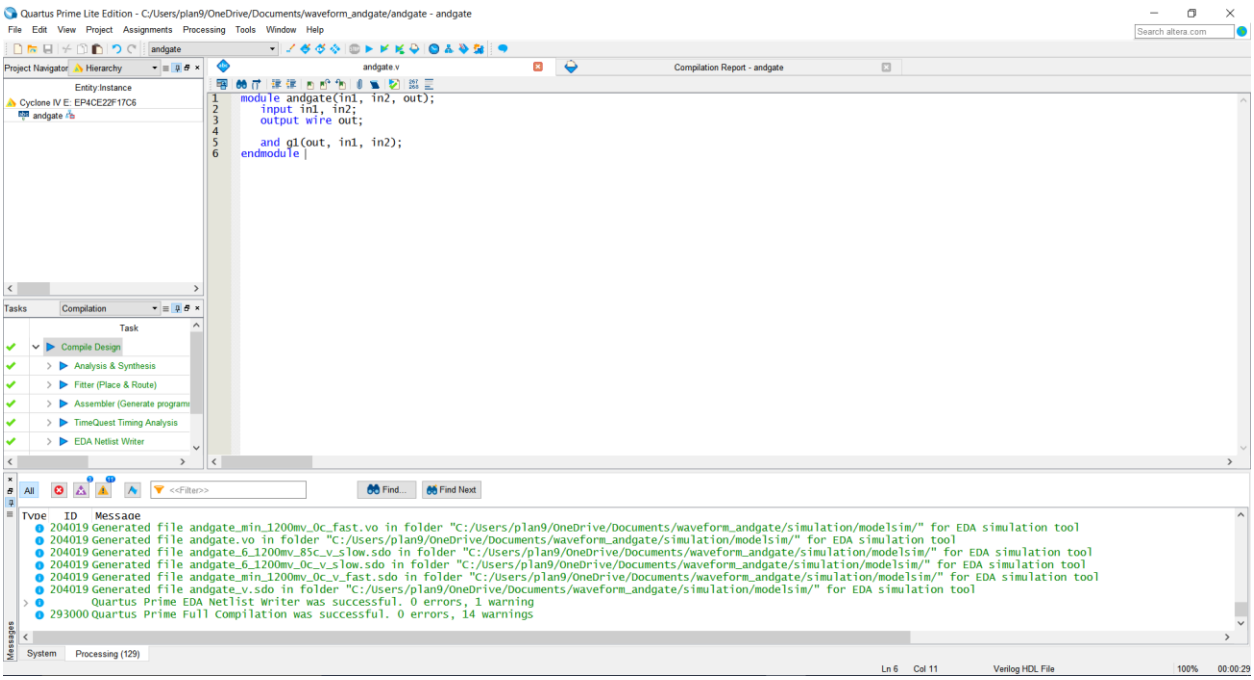
Next we are going to create a test bench that we can use to generate waveforms.

First create a new project and a new file with the following Verilog code.

```
module andgate(in1, in2, out);
	input in1, in2;
	output wire out;

	and g1(out, in1, in2);
endmodule
```

You should see the something similar to the figure below.



After compiling create a new Verilog file name it “andgate_tb”

Then copy and paste the following code and save.

```
`timescale 1ns        / 1ps
module andgate_tb();
            reg in1, in2; // Inputs
            wire out; // Output

            andgate uut (.in1(in1), .in2(in2), .out(out));

            initial begin
                        in1 = 1'b0;
                        in2 = 1'b0;

                        #50; // 50 time units (time unit = 1ns) later change values
                        in1 = 1'b0;
                        in2 = 1'b1;

                        #50;
                        in1 = 1'b1;
                        in2 = 1'b0;

                        #50;
                        in1 = 1'b1;
                        in2 = 1'b1;

                        #50;
                        $stop;

            end
endmodule
```

Then save once you copy and paste the following code.

Verilog Testbench Contents

While the testbench looks similar in structure to a regular module, it contains a few differences. Notably, there are no specified inputs or outputs. Instead, notice that input signals are **reg** datatypes and output signals are **wire** datatypes.

In the above example, the unit under test is initialize in this line of code:

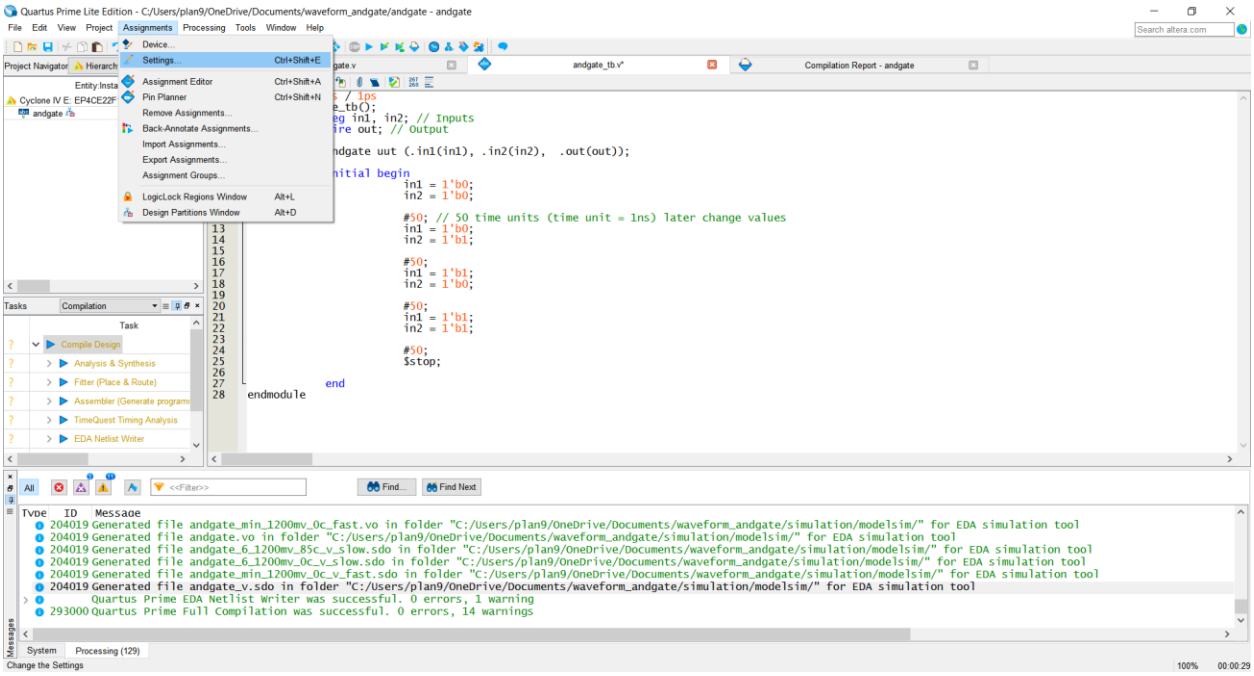
```
andgate uut (.in1(in1), .in2(in2), .out(out));
```

The testbench contains an “initial” block, which is run once when the simulation starts. For this guide, all Verilog code for the testbench is placed in this initial block.

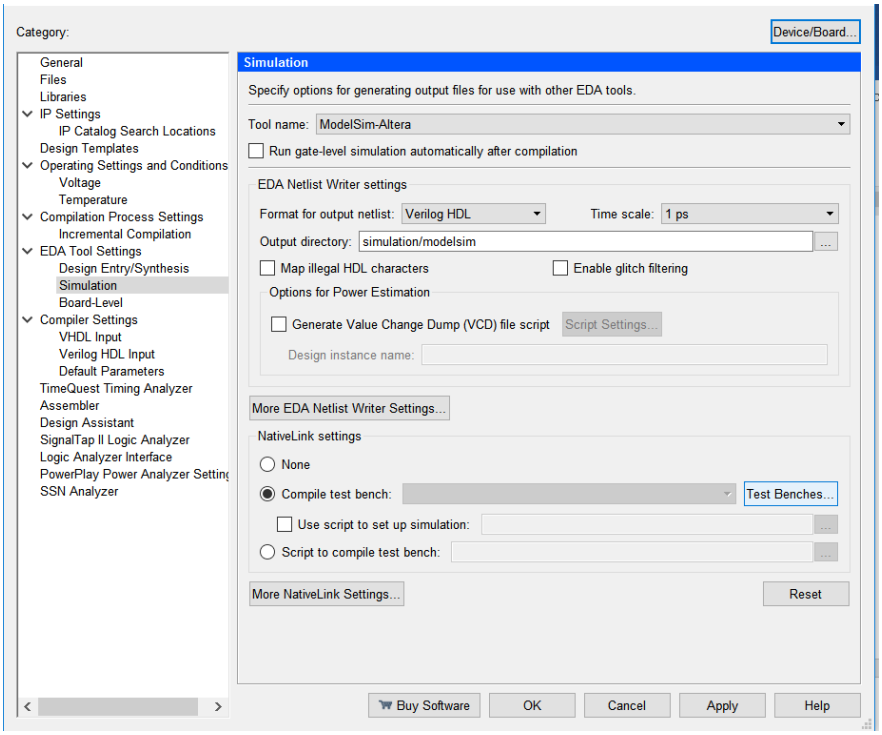
Running Testbench Simulations

Testbench simulations can be run using different methods. This guide outlines running simple testbench simulations for use in the CPE/EEE 64 lab.

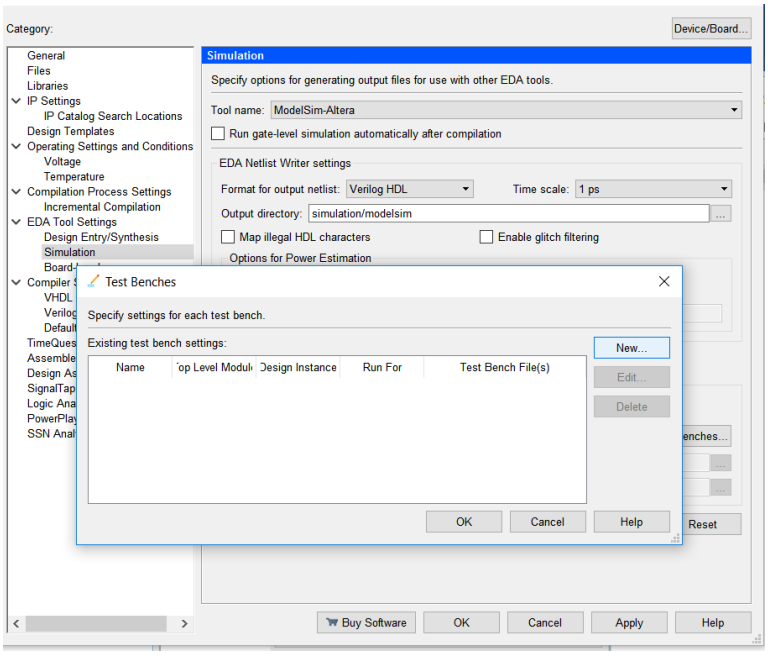
Next go to Assignments, then click settings, as seen in the image below.



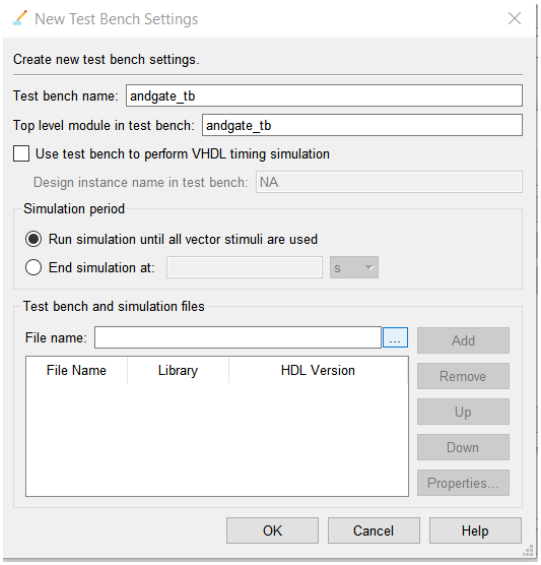
A new window should appear select “Simulation” under EDA Tool Settings



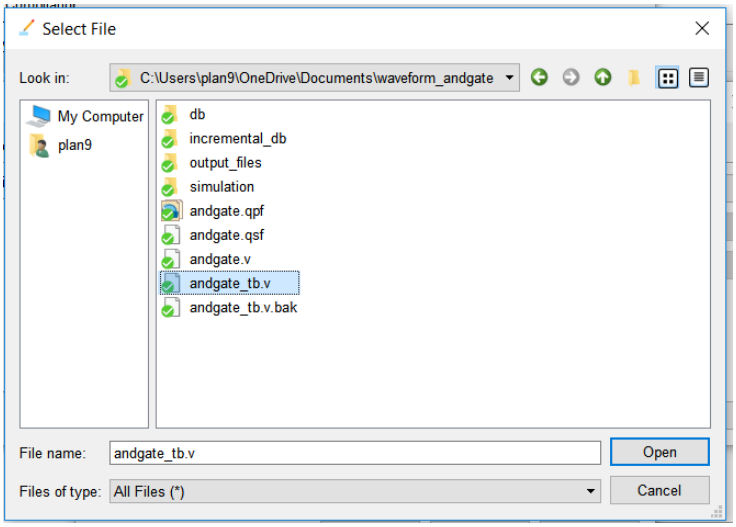
As seen in the image above select Compile test bench. Then click Test Benches.



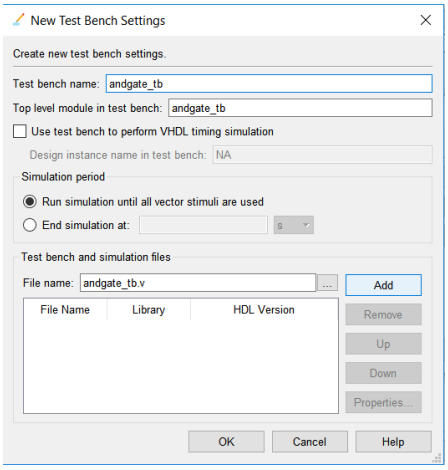
A new window will appear, as seen in the image above. Select new.



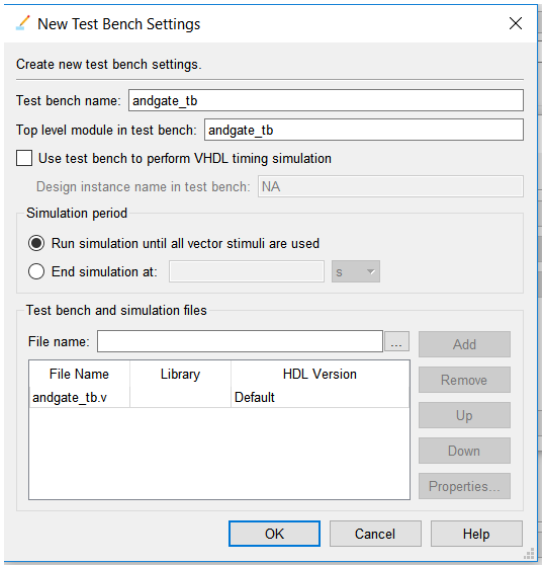
In the Test bench name put the current name of our testbench “andgate_tb”. Then in File name select the “…”. It should open a new window and select your testbench file.



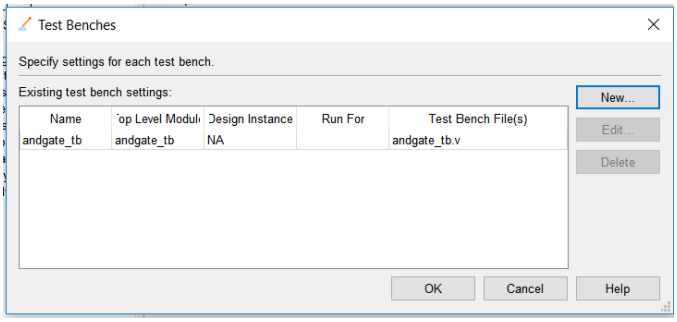
Click Open. Then click add



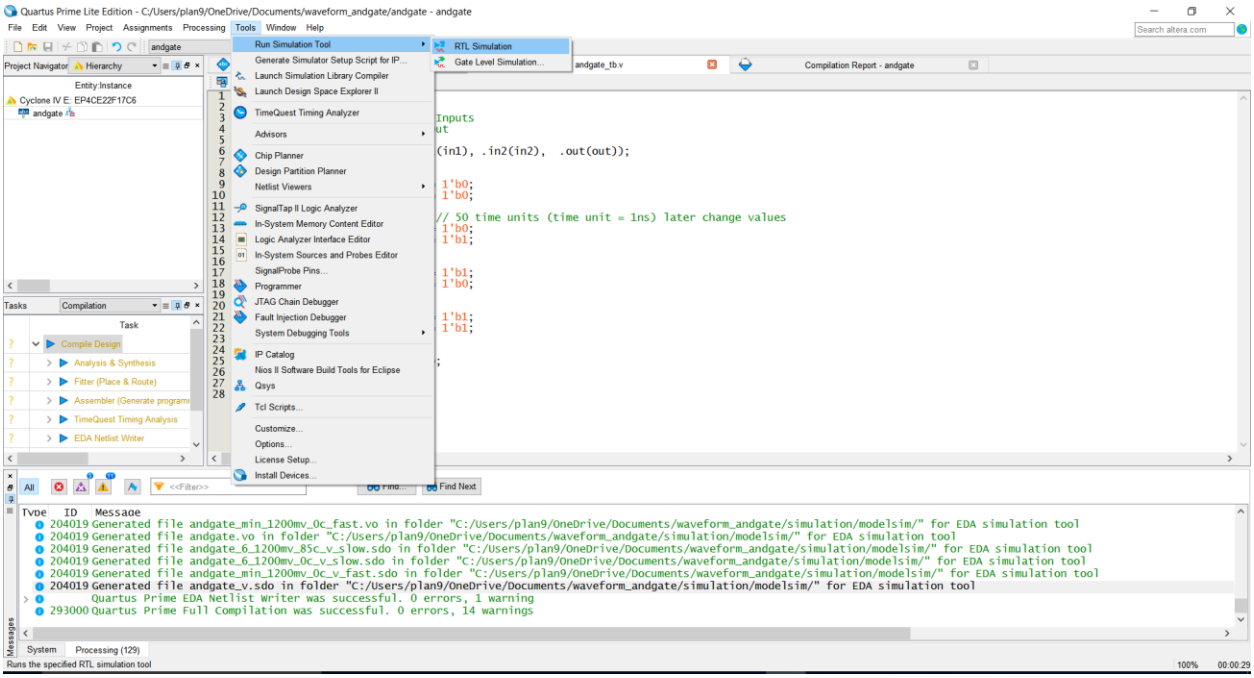
After you click add. And the following change should appear on the window.



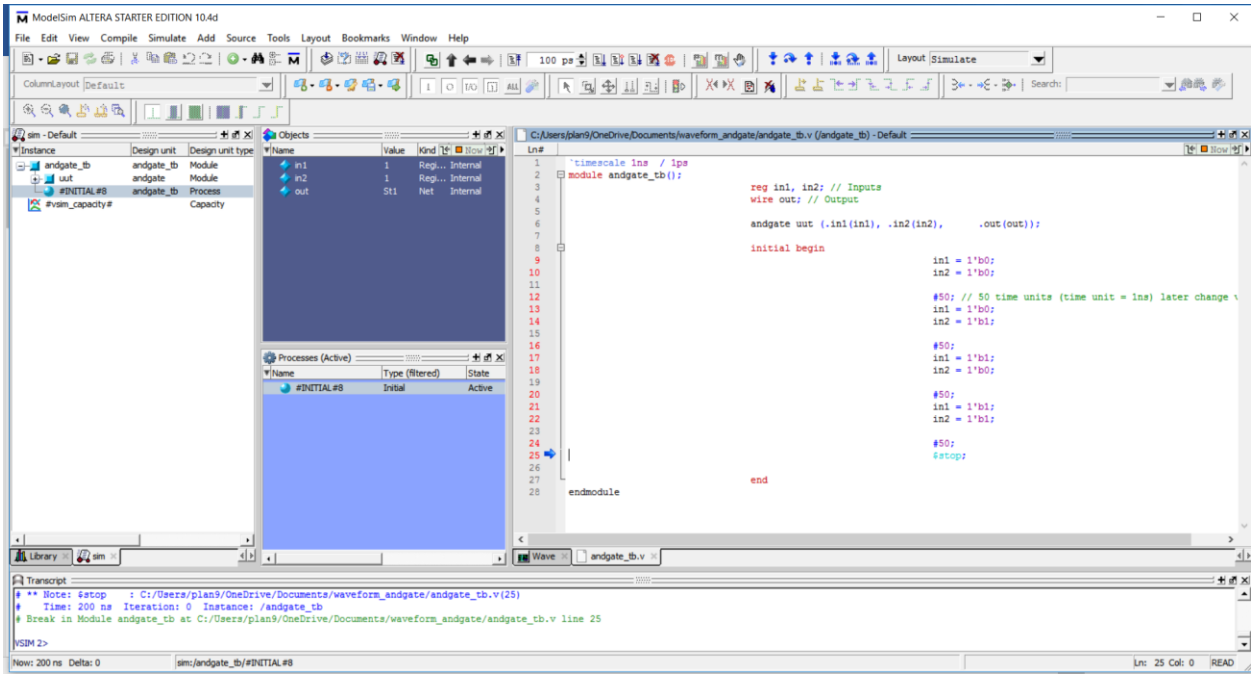
Next click OK and it should take you back to the previous window and select ok



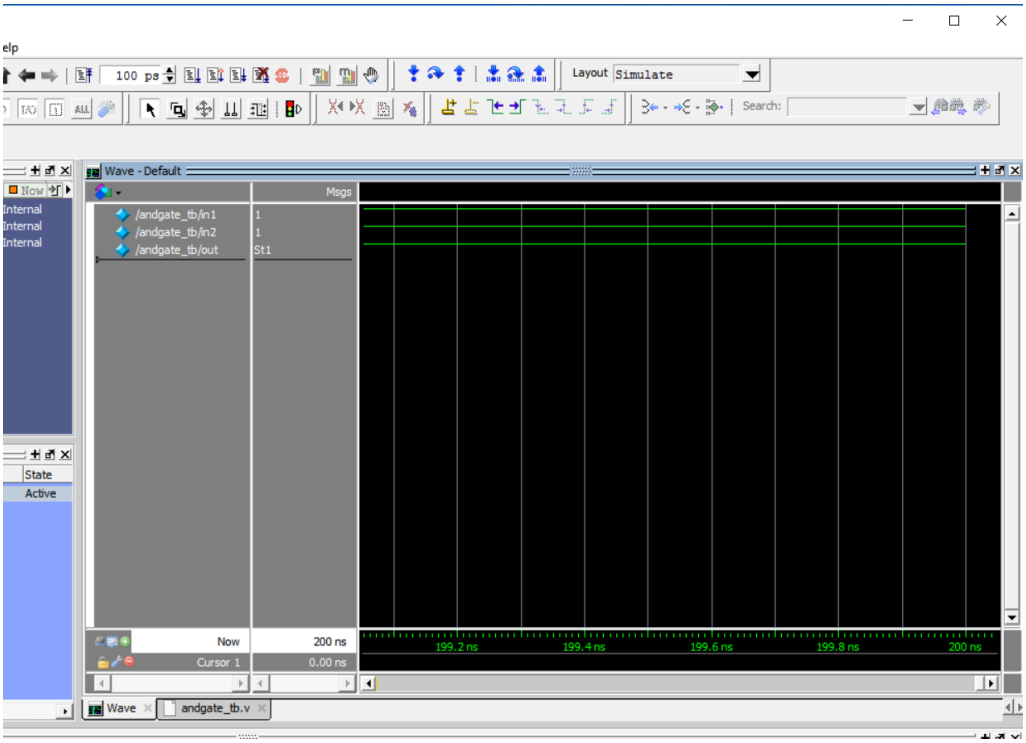
You should see a similar image as above. Then select ok. Then select ok for the settings window as well. Next select tools > Run Simulation Tool > RTL Simulation.



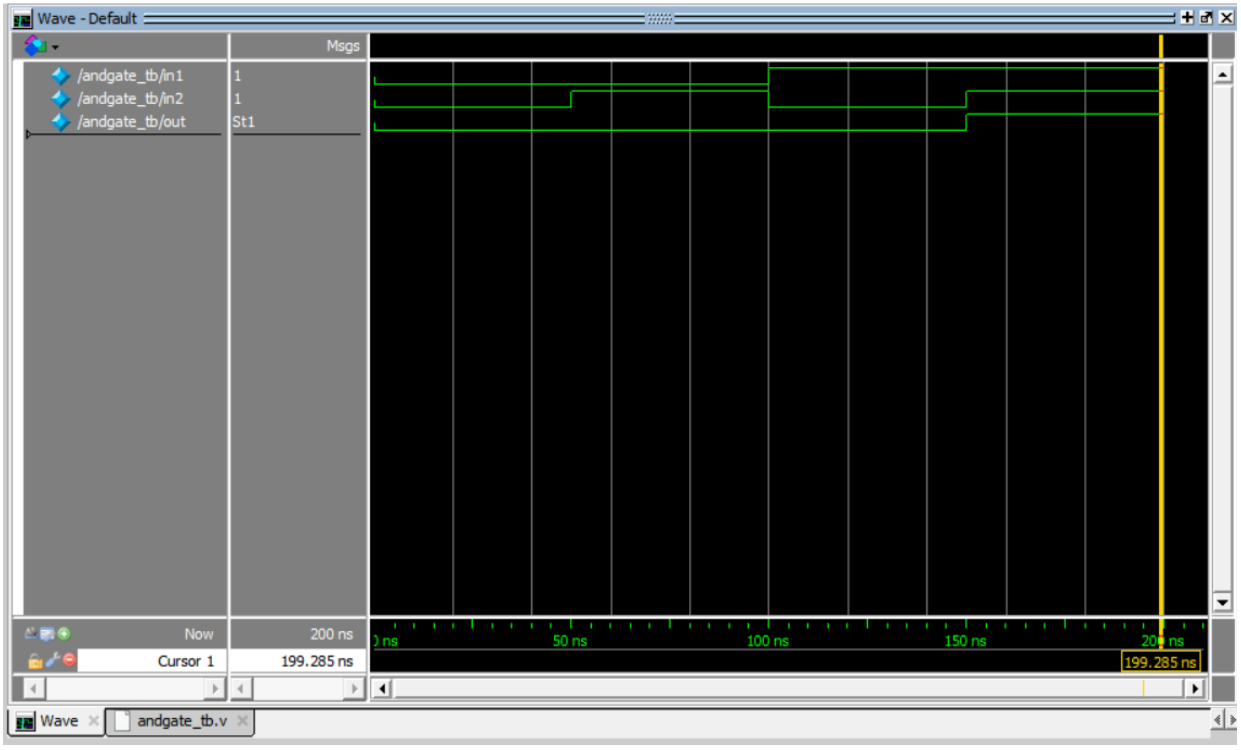
Once you select RTL Simulation Modelsim will open, and you will see the following on your screen. *Note: Be patient it may take a few seconds to open.*



Next select the Wave tab on the center of the screen.



Click the Wave window then press the ‘F’ key on your keyboard. The window should display the following Waveform.



As we noticed in the Waveform above the out signal only goes high when both input signals are high. This provides us with a useful way to debug our logic and see the digital logic behavior before we run the code on our hardware.

What's next?

While this guide offers a basic introduction to simulations and testbenches, there are many more features available that will help test, debug, and validate designs. Learning how to use the various tools early with simple designs will facilitate testing and design of more complex designs later on.

Resources

These resources provide some more information on testbenches and simulation.

• Altera - Getting Started with Quartus II Simulation Using the ModelSim-Altera Software
http://www.altera.com/literature/ug/ug_gs_msa_qii.pdf

• ASIC World - Art of Writing TestBenches

http://www.asic-world.com/verilog/art_testbench_writing.html