Software Documentation For Quartus Prime and ModelSim

Chethan T Bhat, Ritvik Tiwari, Karthik A Shet, Ajay Chaudari

May 2020

Table of Content

1 What are Quartus and ModelSim?

Intel Quartus Prime is programmable logic device design software by Intel; prior to Intel's acquisition of Altera the tool was called Altera Quartus Prime, earlier Altera Quartus II. Quartus Prime enables analysis and synthesis of HDL designs, which enables the developer to compile their designs, perform timing analysis, examine RTL diagrams, simulate a design's reaction to different stimuli, and configure the target device with the programmer. Quartus Prime includes an implementation of VHDL and Verilog for hardware description, visual editing of logic circuits, and vector waveform simulation.

ModelSim is a multi-language HDL simulation environment by Mentor Graphics, for simulation of hardware description languages such as VHDL, Verilog and SystemC, and includes a built-in C debugger. ModelSim can be used independently, or in conjunction with Intel Quartus Prime, Xilinx ISE or Xilinx Vivado. Simulation is performed using the graphical user interface (GUI), or automatically using scripts.

2 Installing Quartus and ModelSim

In this part of the guide we will walk through the process of installing the software Quartus Prime Lite Edition Version 19.1

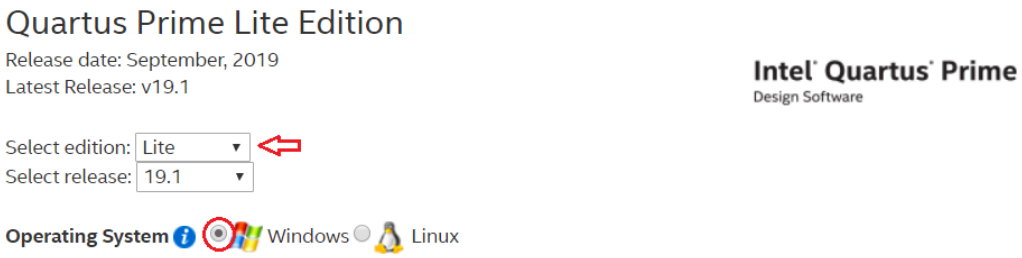
System Requirement:

1. A full installation of the Intel FPGA Complete Design Suite v19.1 requires approximately 14GB of available disk space on the drive or partition where you are installing the software.

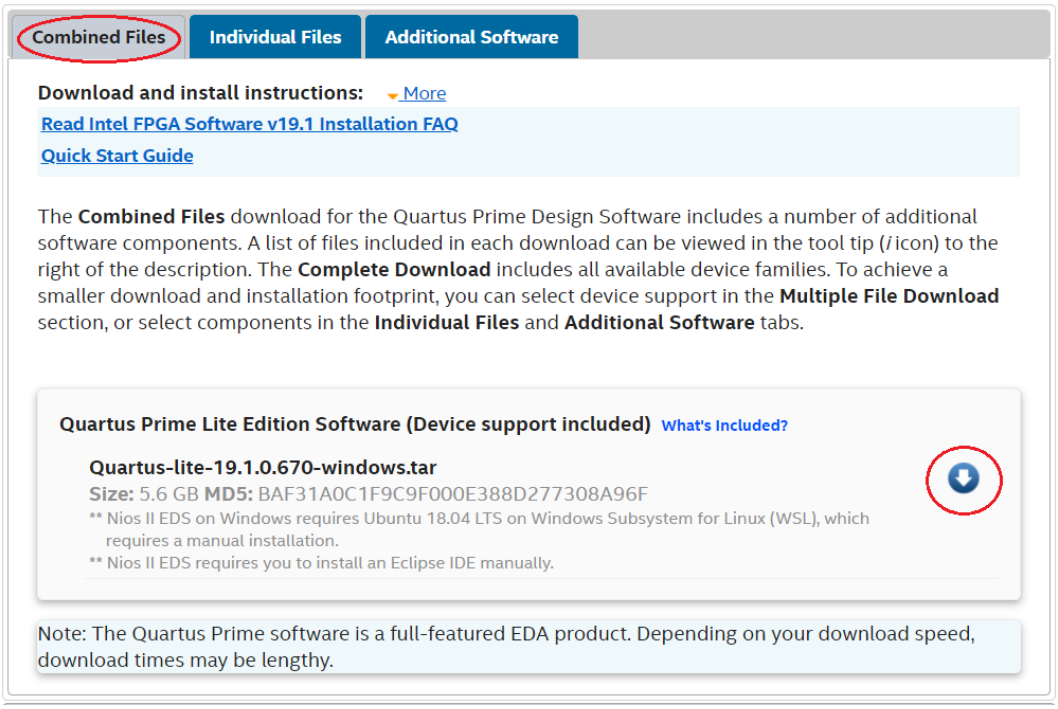
2. Recommended Physical RAM requirement is more than 2GB.

If you are running any type of antivirus software, you can temporarily disable the software during the Quartus Prime software download and installation process to avoid unnecessary issues.

On the Download Page select the edition as lite and select version as 19.1. Also select the desired operating system. We selected window for further installation.

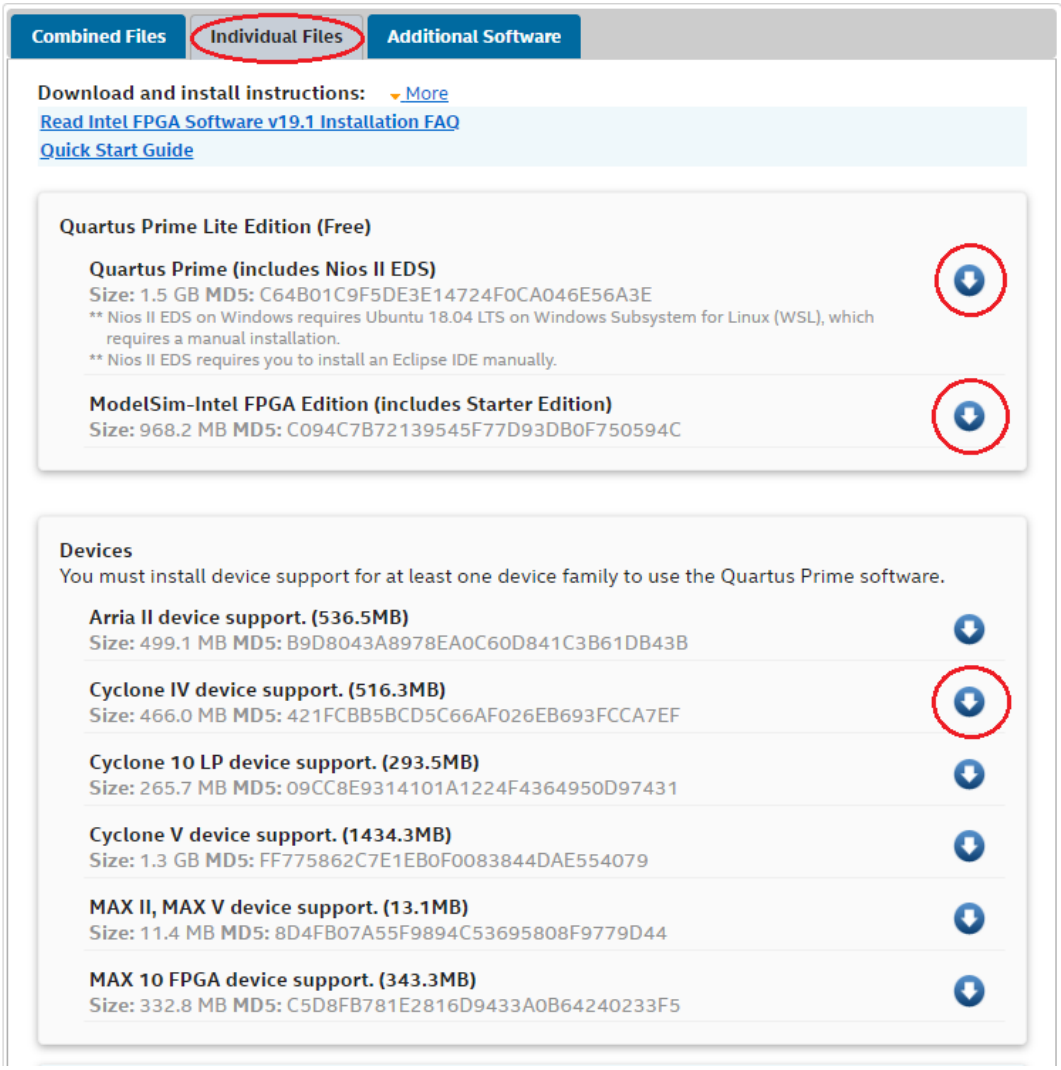


2.1 Download using Complete Software Package



1. The Combined Files download includes a number of additional software components. This file provides device support for various device families.
   (a) Arria II device support.
   (b) Cyclone IV device support.
   (c) Cyclone 10 LP device support.
   (d) Cyclone V device support.
   (e) MAX II, MAX V device support.
2. After download is done on your local machine, extract all of the files to the same directory using WinZip or any other software.

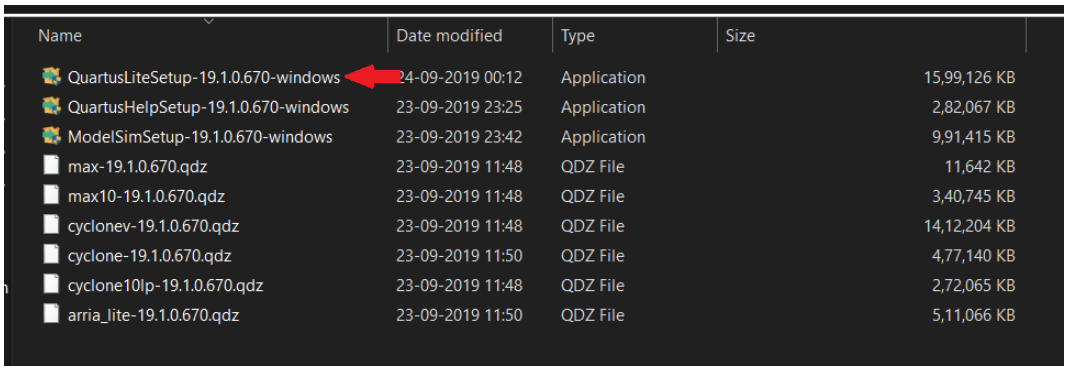
2.2 Download using Individual Software Package



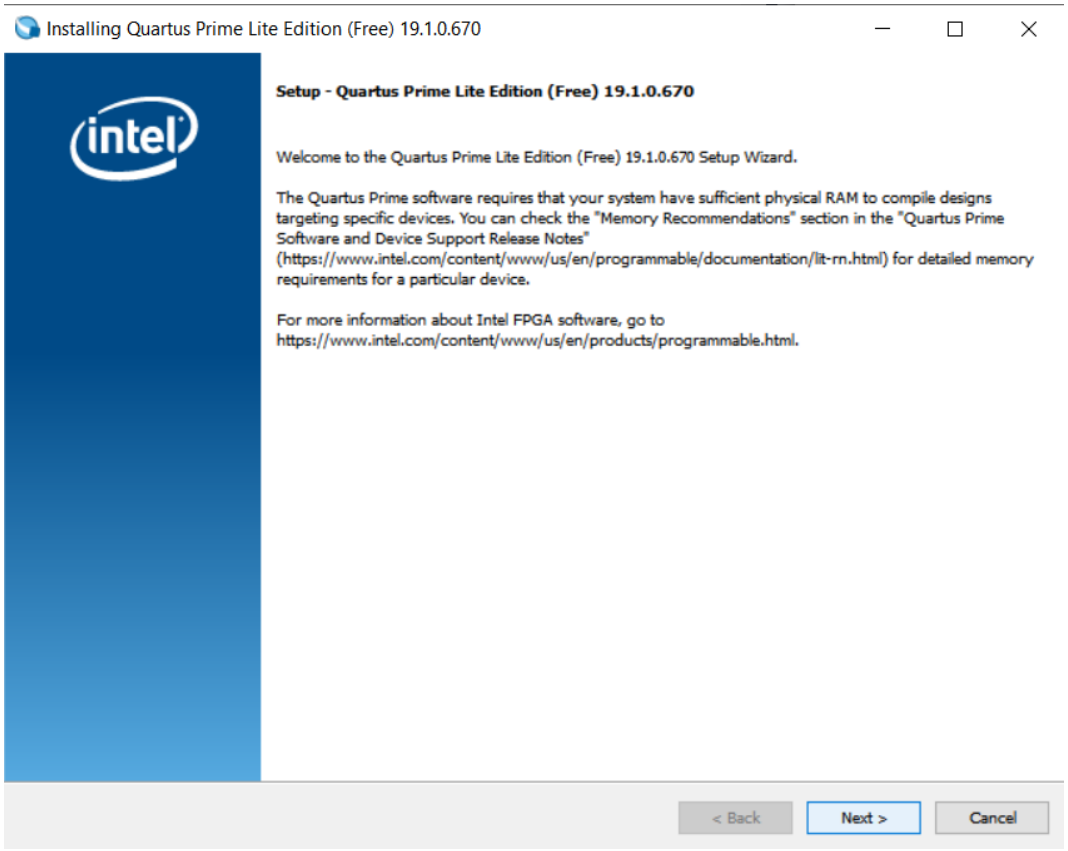
1. In Individual Files download, download both Quartus Prime lite edition and ModelSim, you also need to download files to get support for a particular device family.
2. Select from the below option which are necessary.
   (a) Arria II device support.
   (b) Cyclone IV device support.
   (c) Cyclone 10 LP device support.
   (d) Cyclone V device support.
   (e) MAX II, MAX V device support.
3. Download Cyclone IV device support as Intel DE0-Nano uses Altera Cyclone IV, it will also support other device from the same family.

2.3 Installation after Download

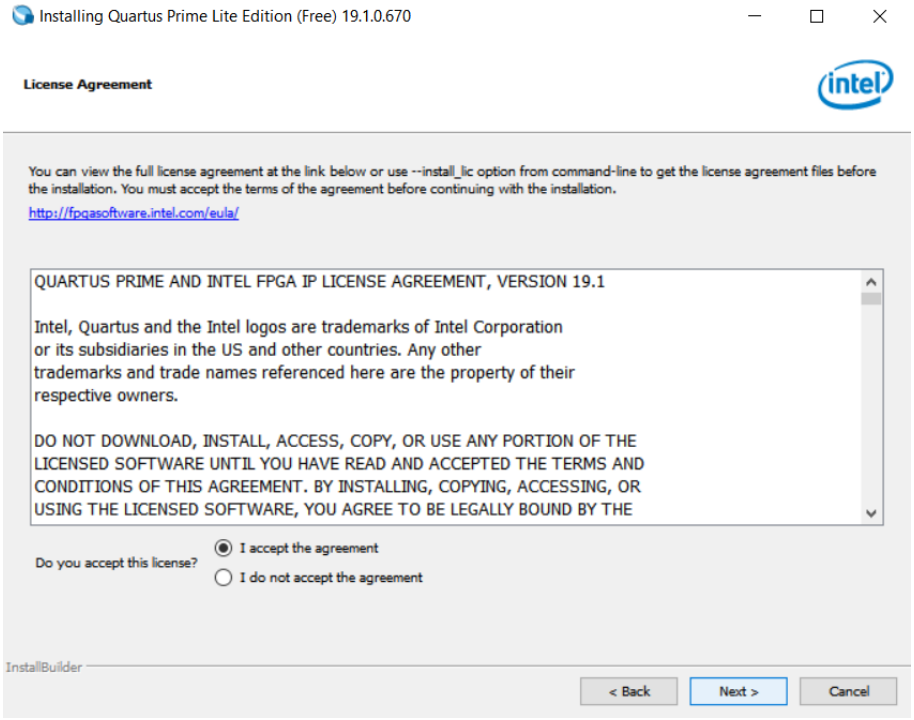
1. After Download necessary files extract them in a single folder and click on the Quartus-LiteSetup and allow



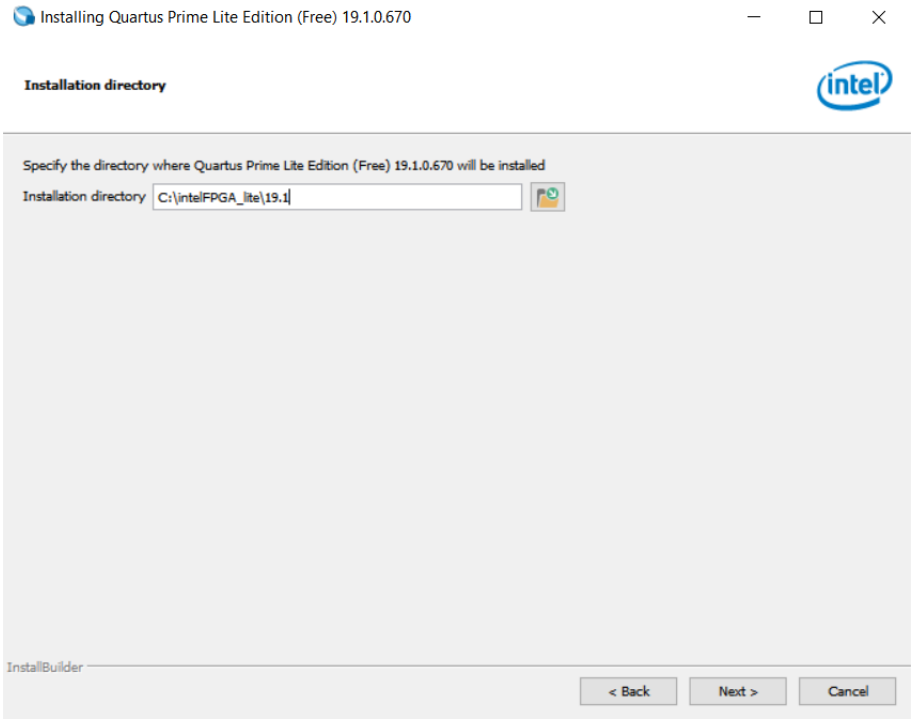
2. Click on next to start with the installation.



3. Click on agree with terms and condition and proceed

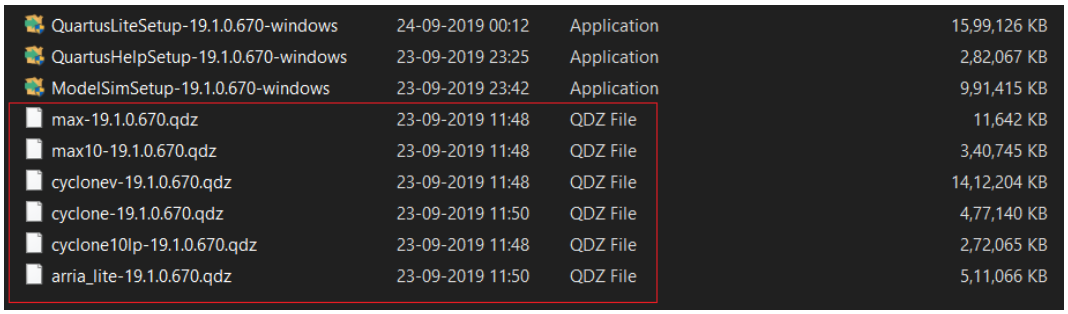


4. Enter the path where u need the software to be installed



5. After this the installation starts and may takes some time to complete during this time ModelSim and QuartusHelp will also get installed

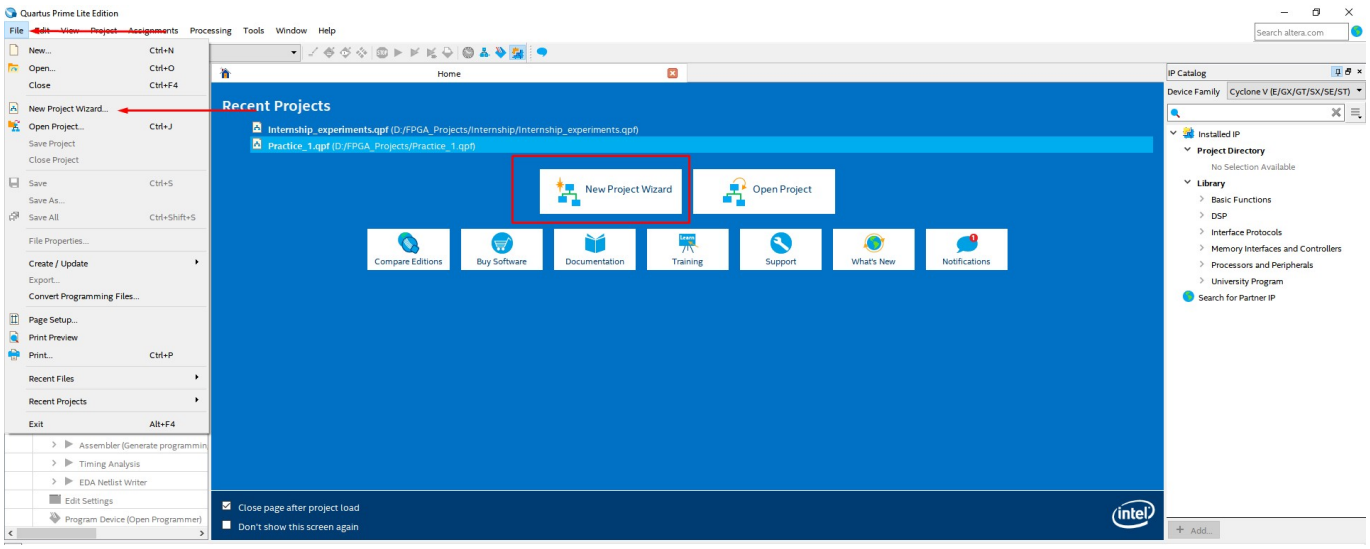
6. Copy the below file and paste it in the C:/intelFPGA_lite/19.1/modelsim_ase/win32aloem folder to get support of the device families while creating project



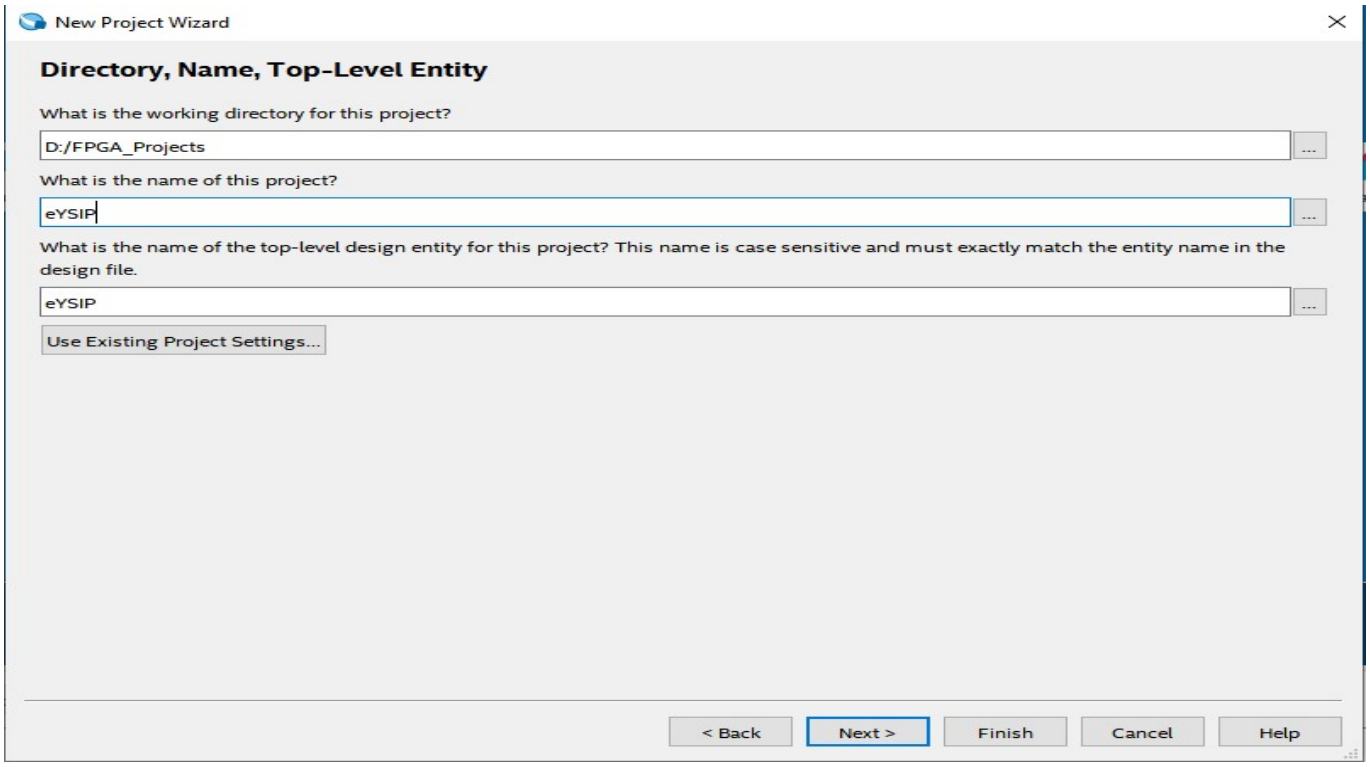
3 Getting started with Quartus prime

3.1 Creating a new project

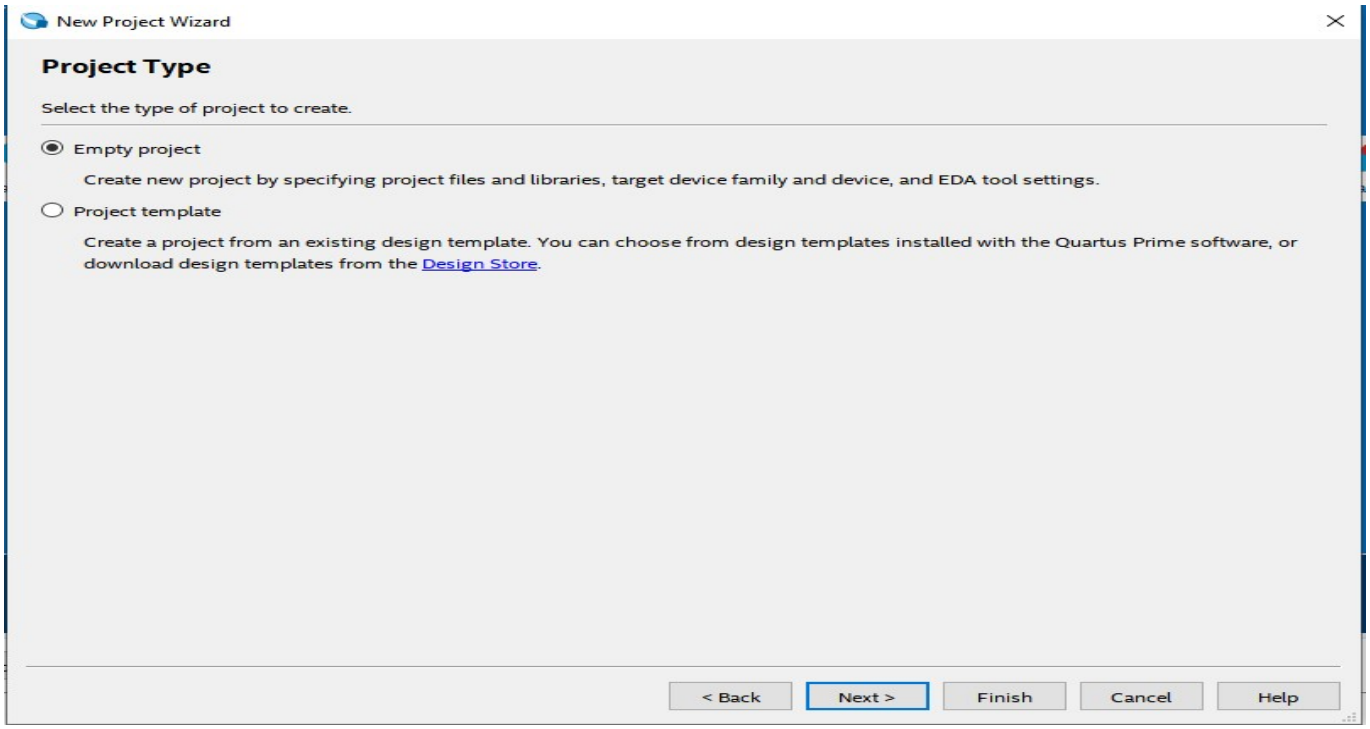
Step 1 : Open the Quartus Prime Software
Step 2 : Click on File and then click on New Project Wizard. Alternatively, New project wizard can be opened from the home tab that is seen when quartus is opened.



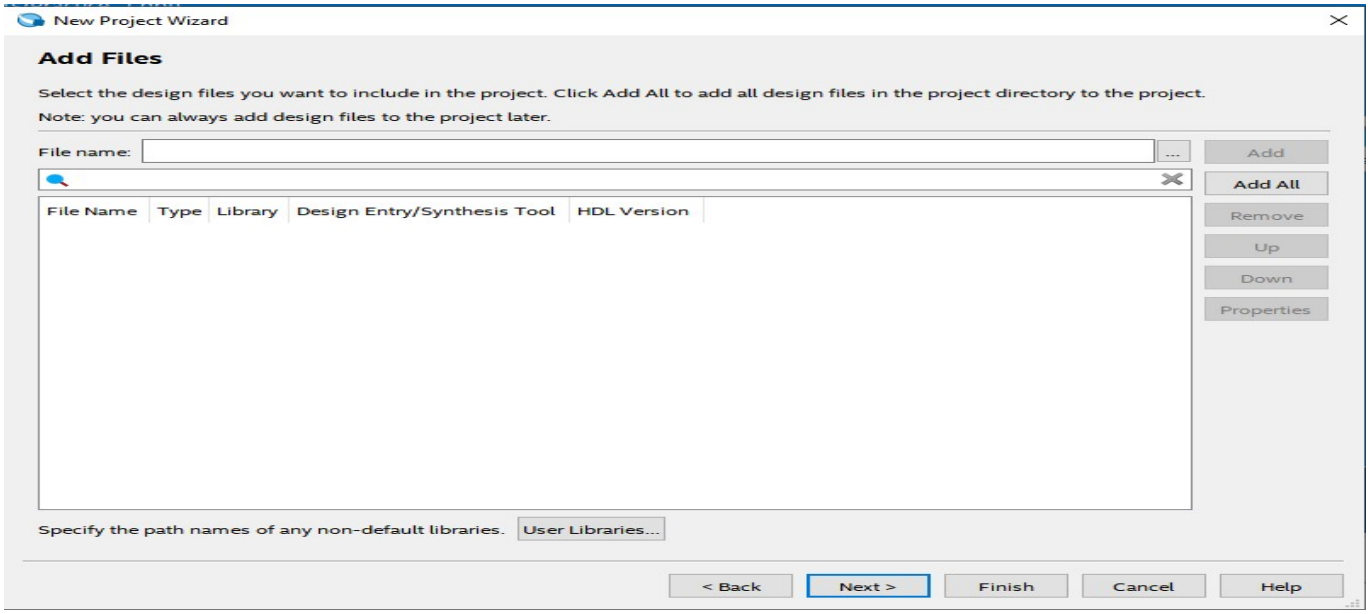
Step 3: Click Next on the Dialog Box and select the directory in which the project is to be saved. Give the project name and click on next.



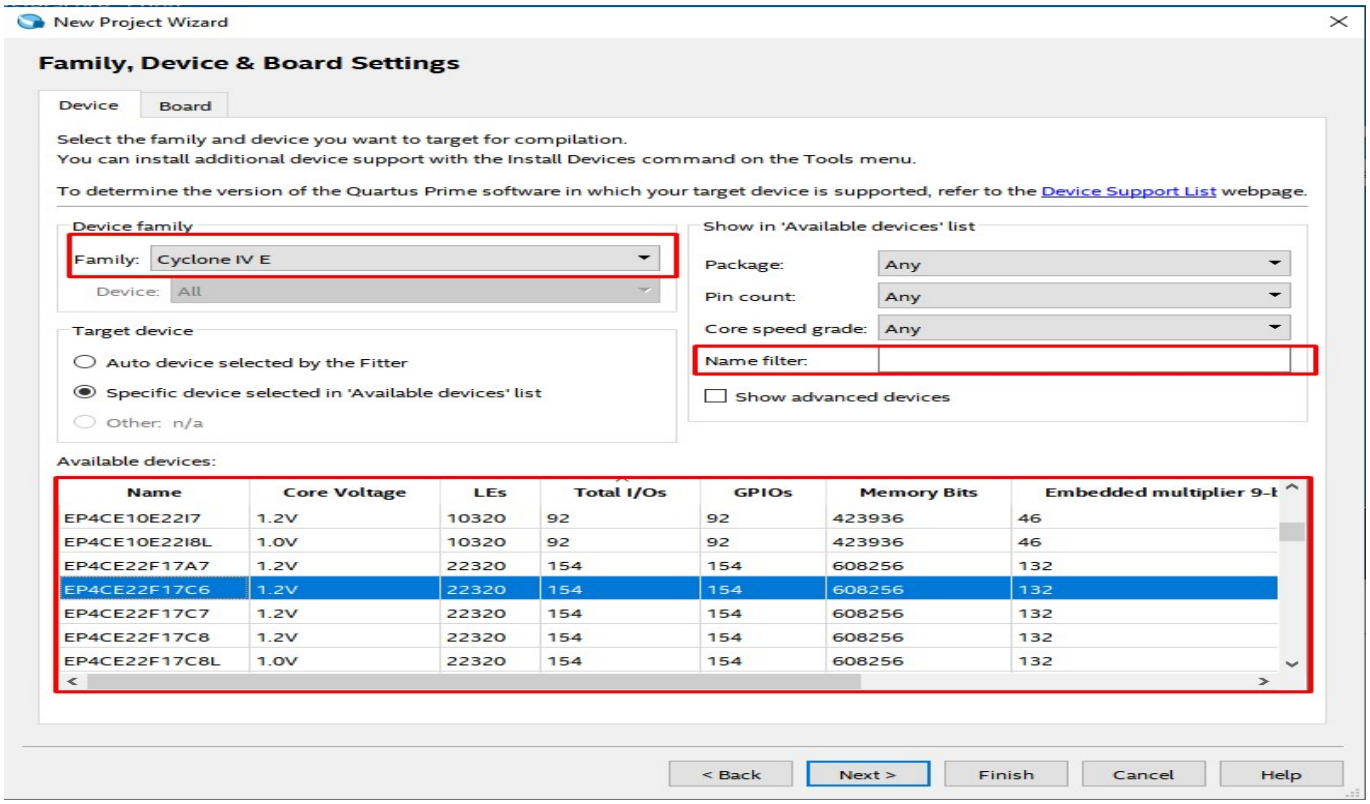
Step 4 : Choose a project Template if its present, else choose empty project.



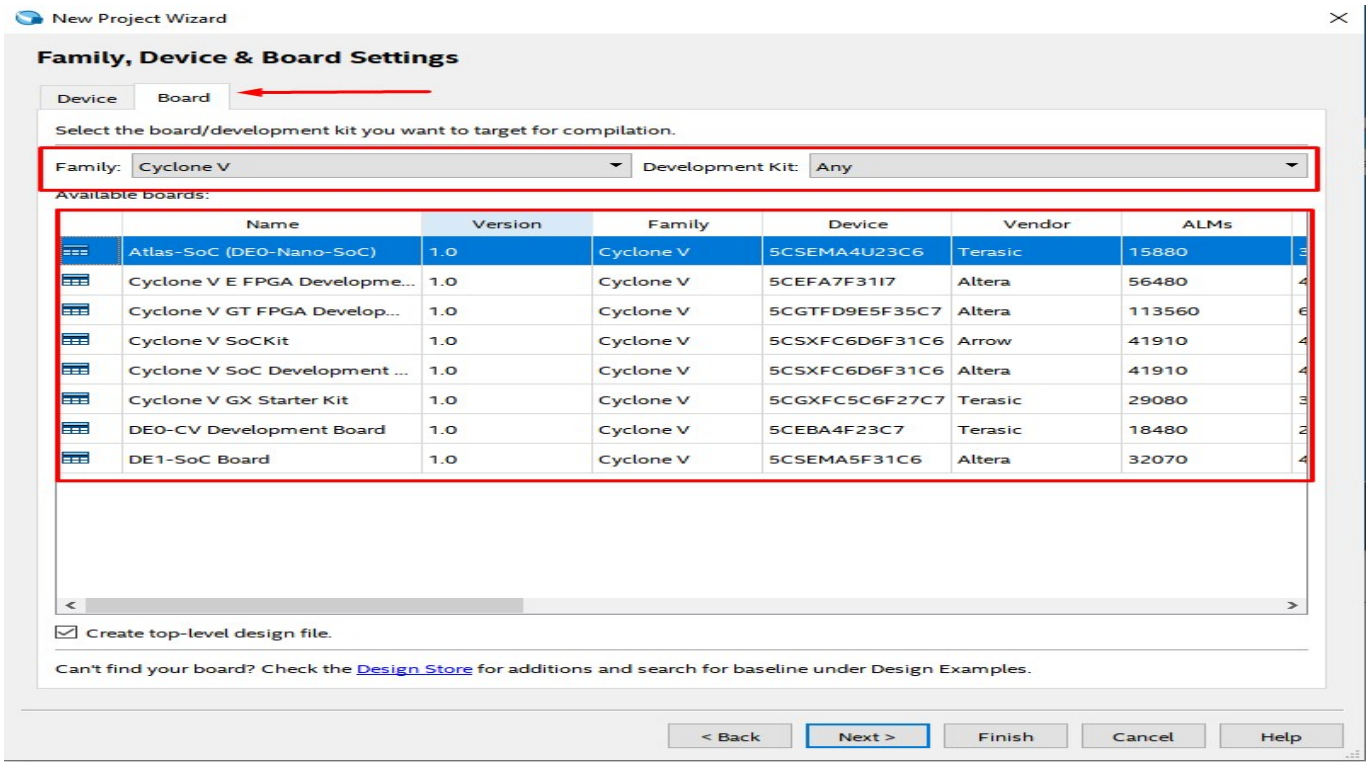
Step 5 :Add all the necessary files(if any) and click on next.



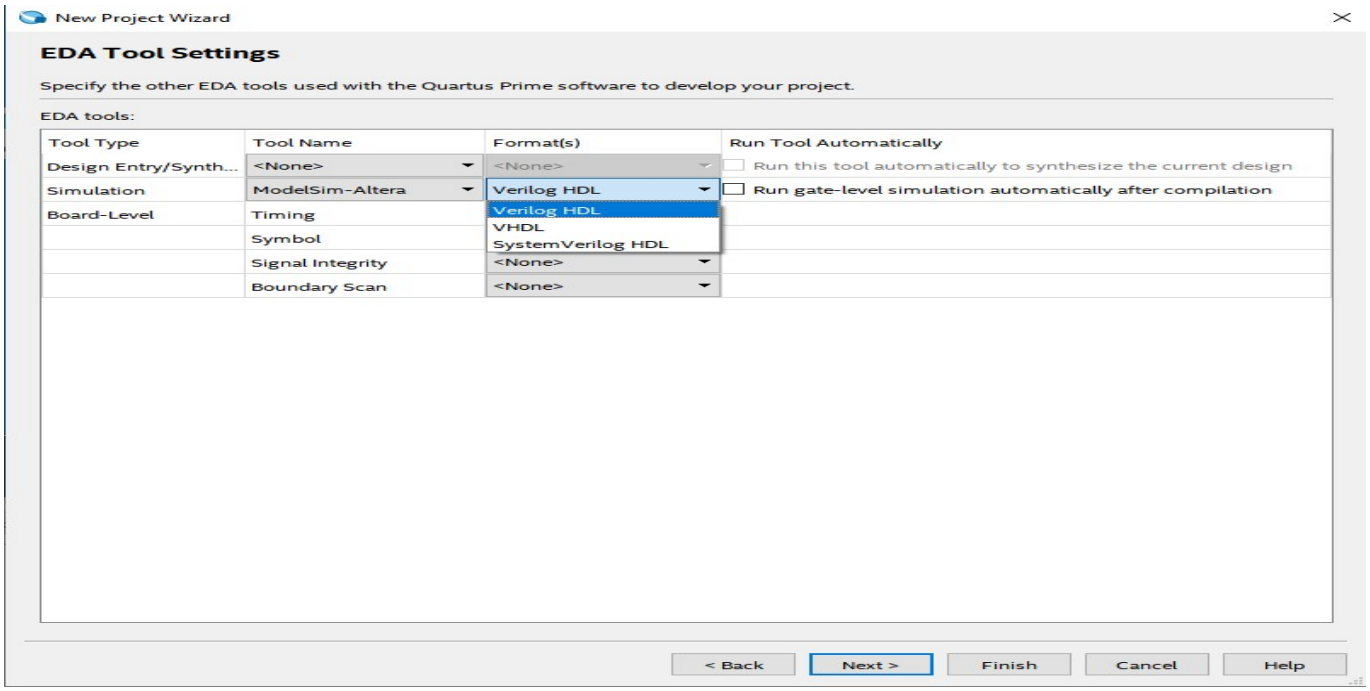
Step 6 : Choose the fpga device that is being used. Select the family and choose the chipset from the list or search for a specific name using the filter provided.If the device being worked on is a development kit, then click on the boards tab and choose the relevant kit



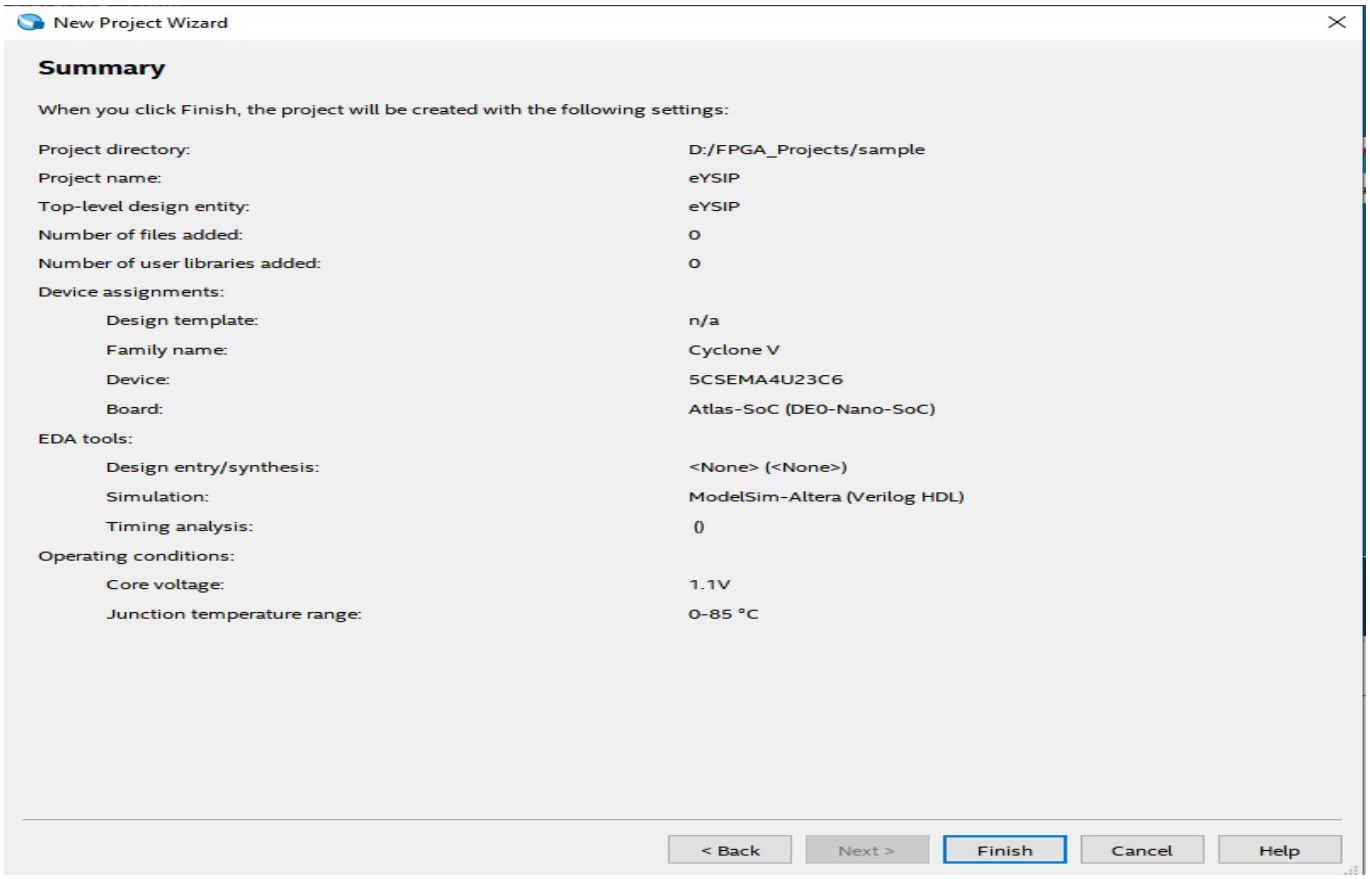
If the device being worked on is a development kit, then click on the boards tab and choose the relevant kit



Step 7: Select the tools used in the project. For simulation, choose ModelSim-alterra. Also choose a suitable format(verilog, VHDL or SystemVerilog)



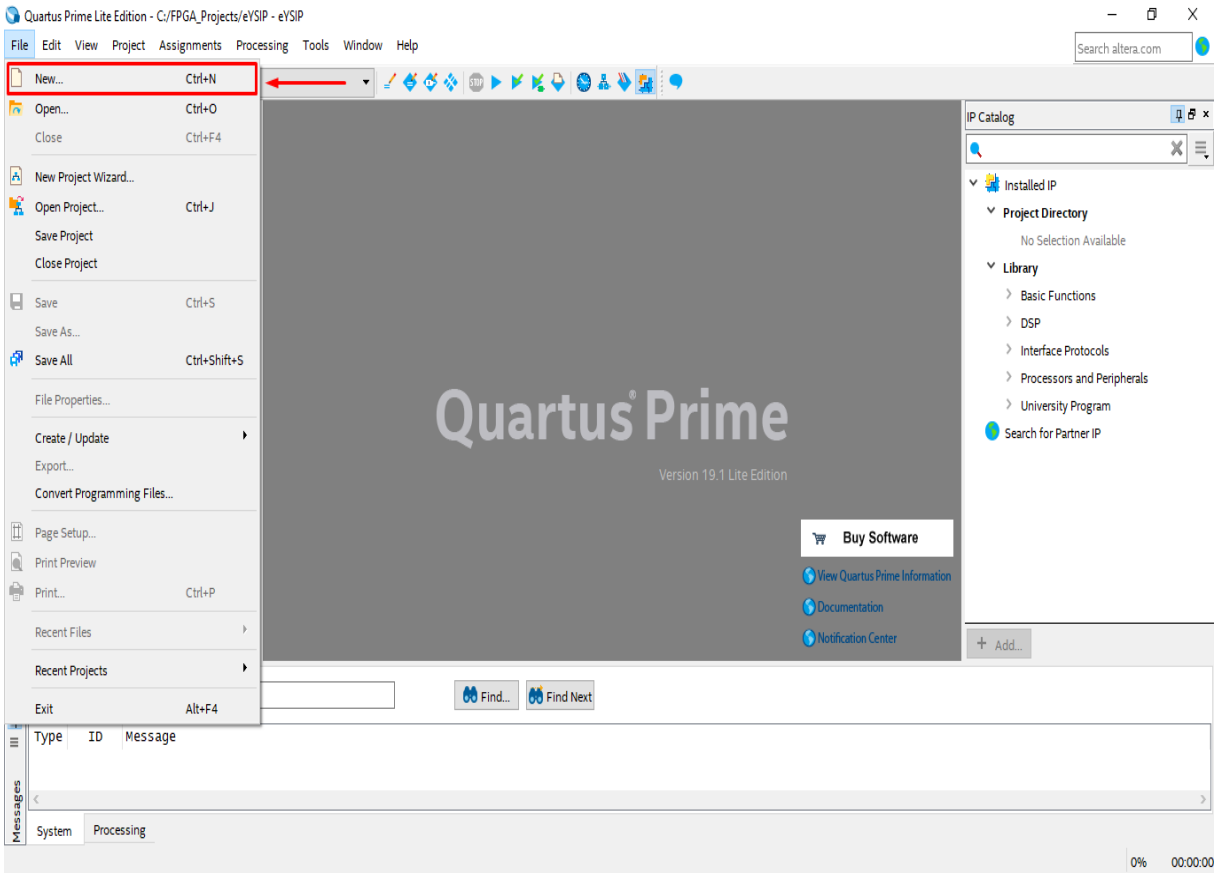
Step 8 :Review the information and click on finish to successfully create a Project



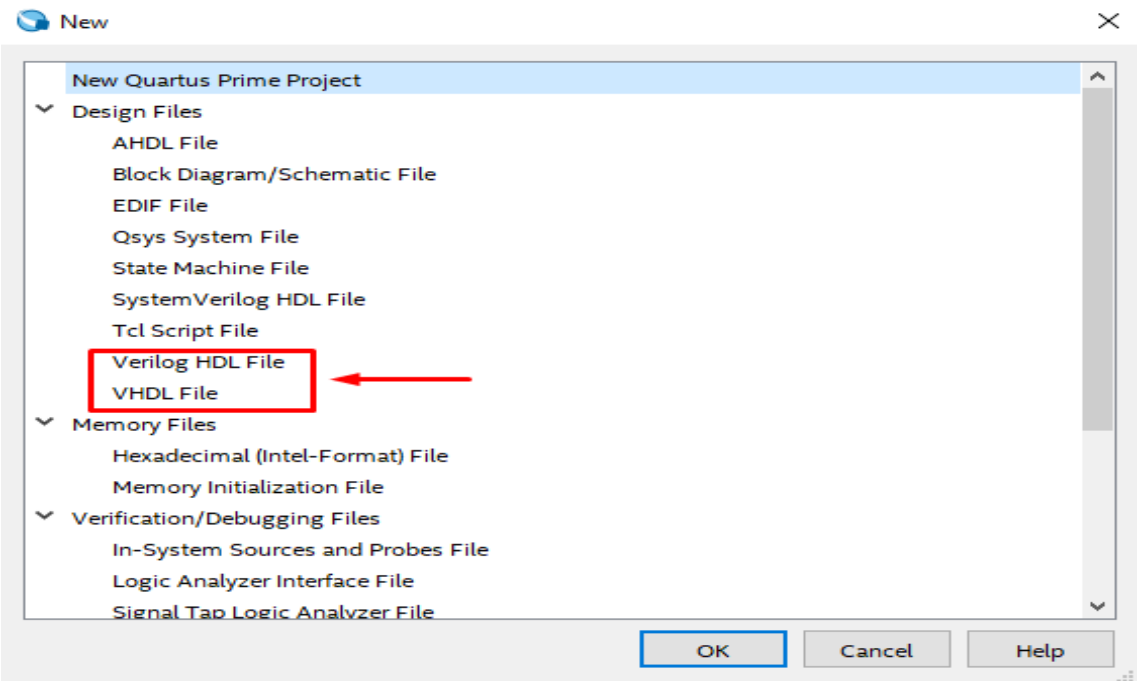
3.2 Creating new files in the project

1. Verilog/VHDL File

Step 1: Click on file and then click on new



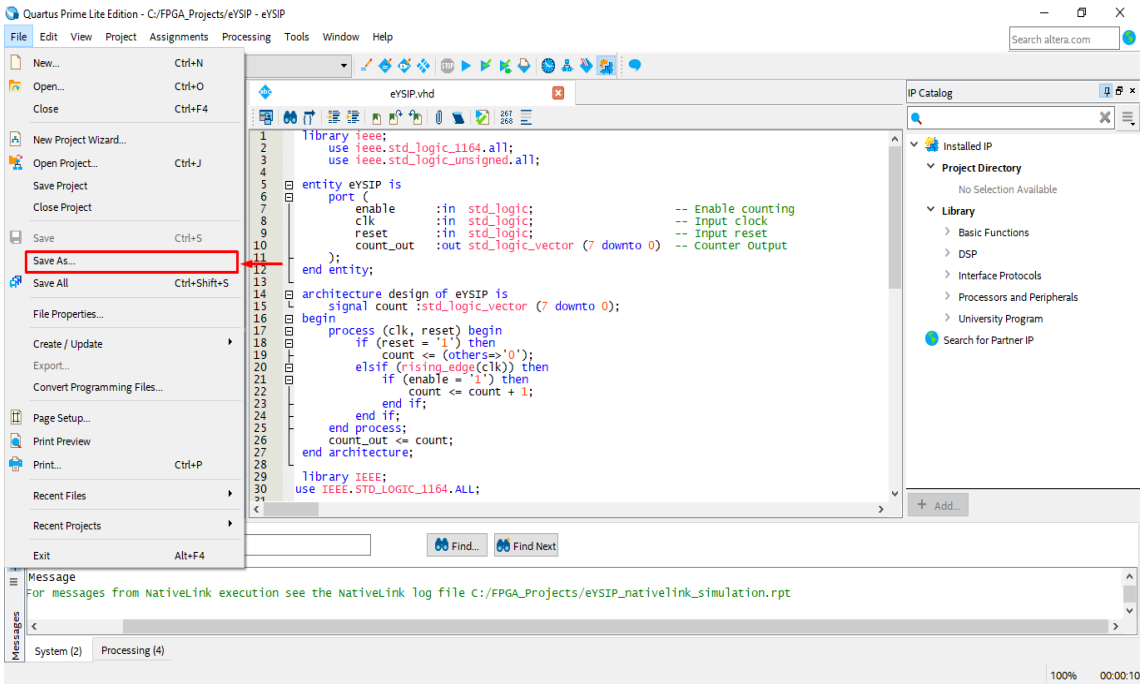
Step 2 :Select the file type from drop down menu i.e either verilog or VHDL



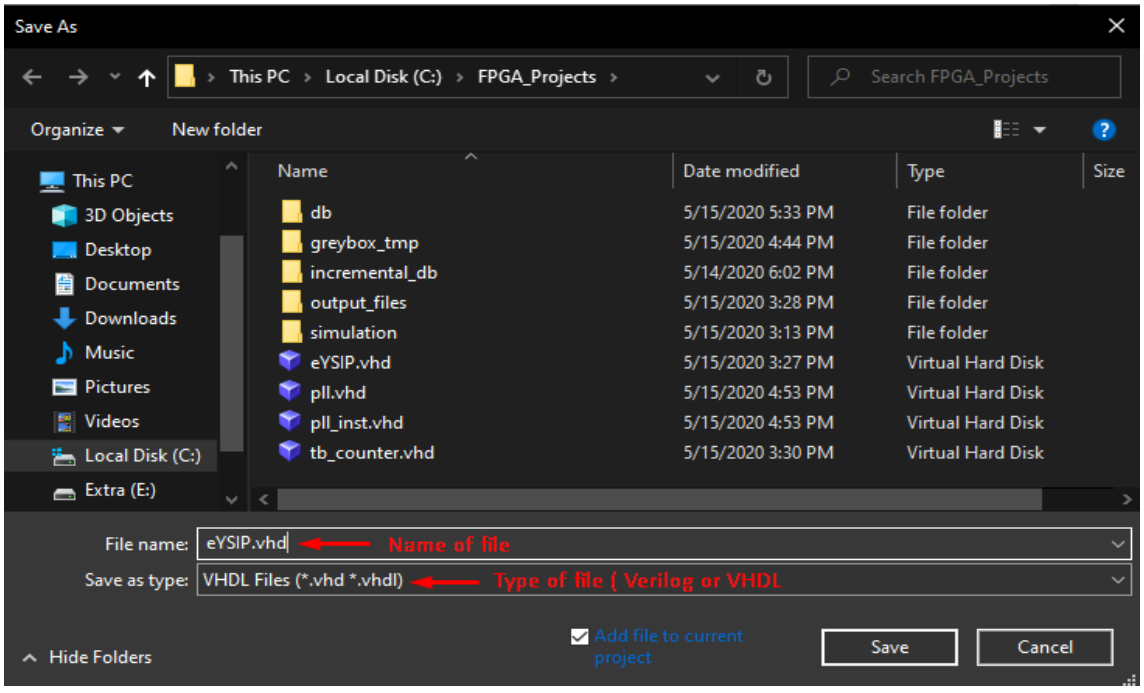
Step 3:Write down your verilog or VHDL code (For demonstration we have used a simple 8 bit up-counter code)

```
library ieee;
    use ieee.std_logic_1164.all;
    use ieee.std_logic_unsigned.all;

entity eYSIP is
    port (
        enable      :in  std_logic;                     -- Enable counting
        clk         :in  std_logic;                     -- Input clock
        reset       :in  std_logic;                     -- Input reset
        count_out   :out std_logic_vector (7 downto 0)  -- Counter Output
    );
end entity;

architecture design of eYSIP is
    signal count :std_logic_vector (7 downto 0);
begin
    process (clk, reset) begin
        if (reset = '1') then
            count <= (others=>'0');
        elsif (rising_edge(clk)) then
            if (enable = '1') then
                count <= count + 1;
            end if;
        end if;
    end process;
    count_out <= count;
end architecture;

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
```

Step 4 :After completing the code click on file and then select save as

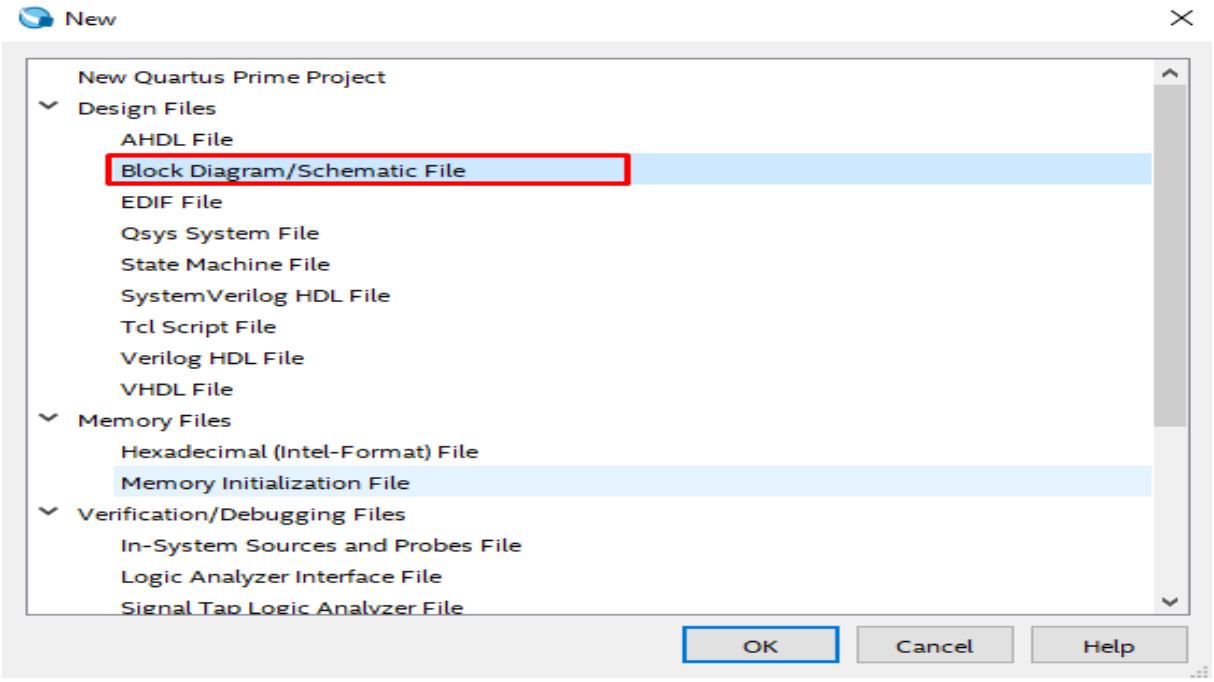


Step 5:Enter the name of file (it should be similar to module/entity name), select the file type and then click on save

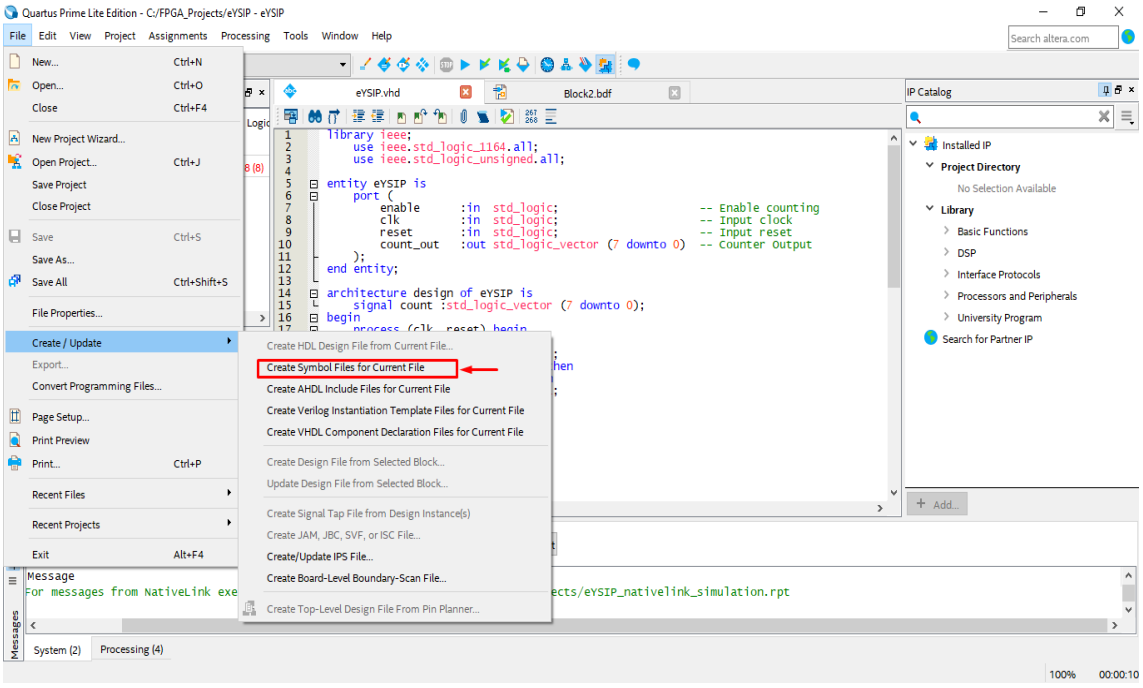


2. Block Diagram File

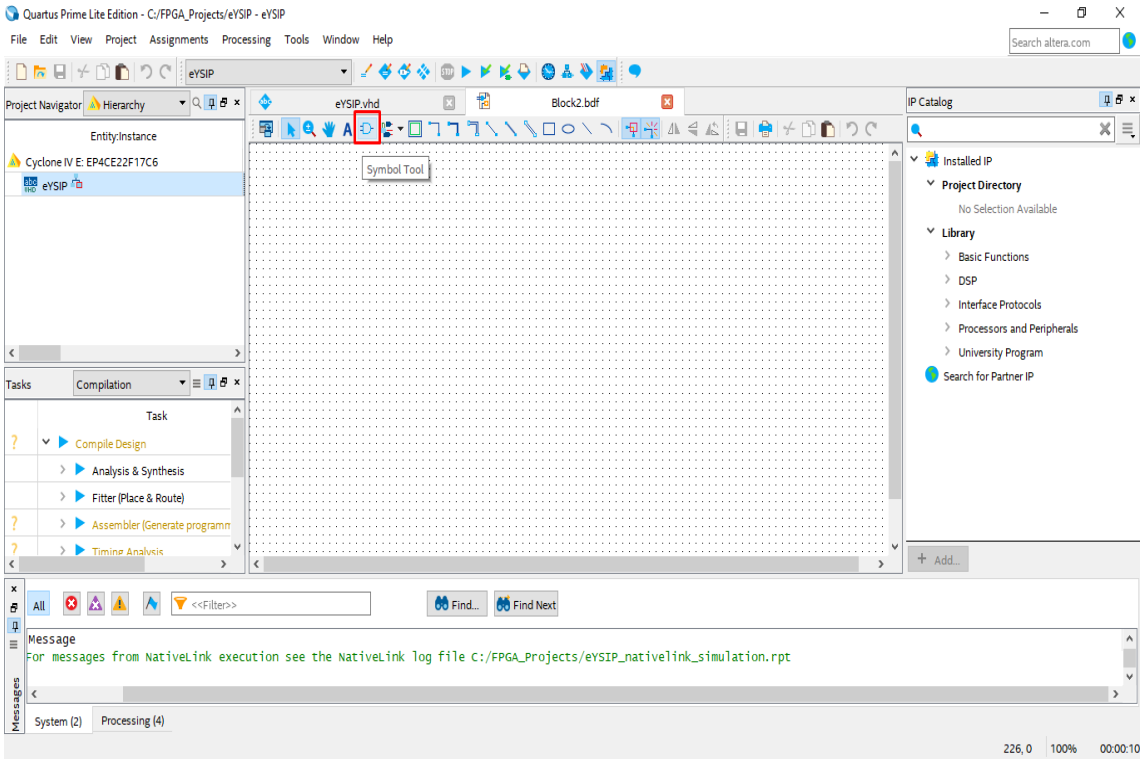
Step 1: Click on file→new and select Block Diagram/Schematic File from the drop down menu



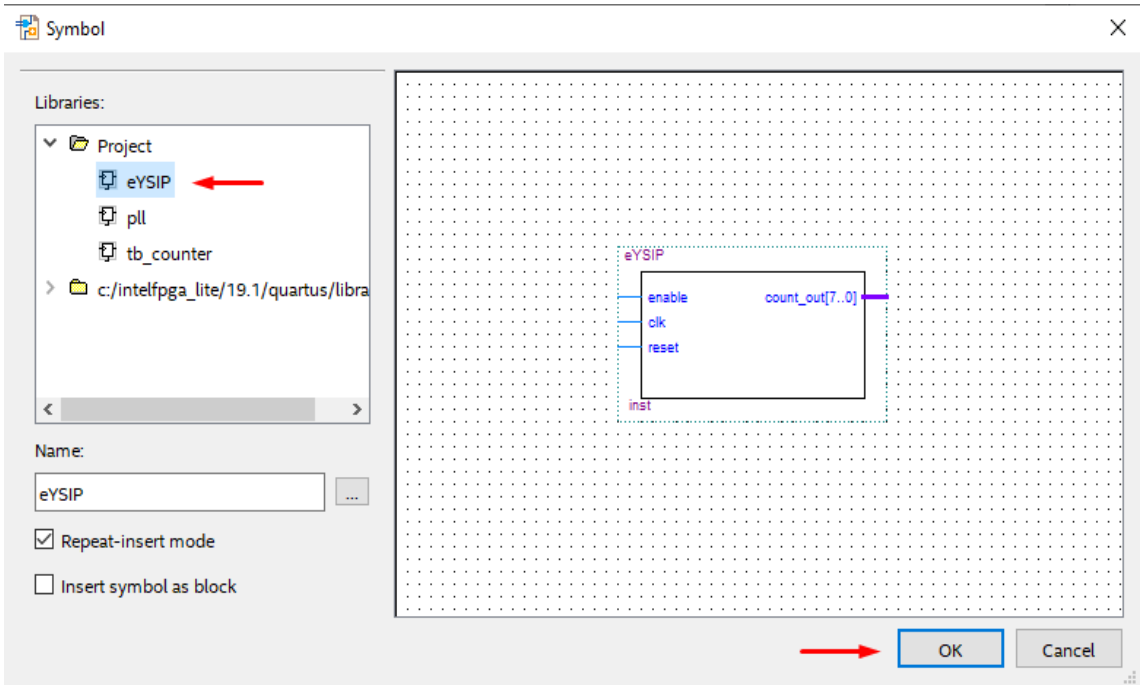
Step 2: Create symbol files for your block diagram. First of all go to your vhdl/verilog code. Then, Go to file→Create/Update→Create symbol files for current file



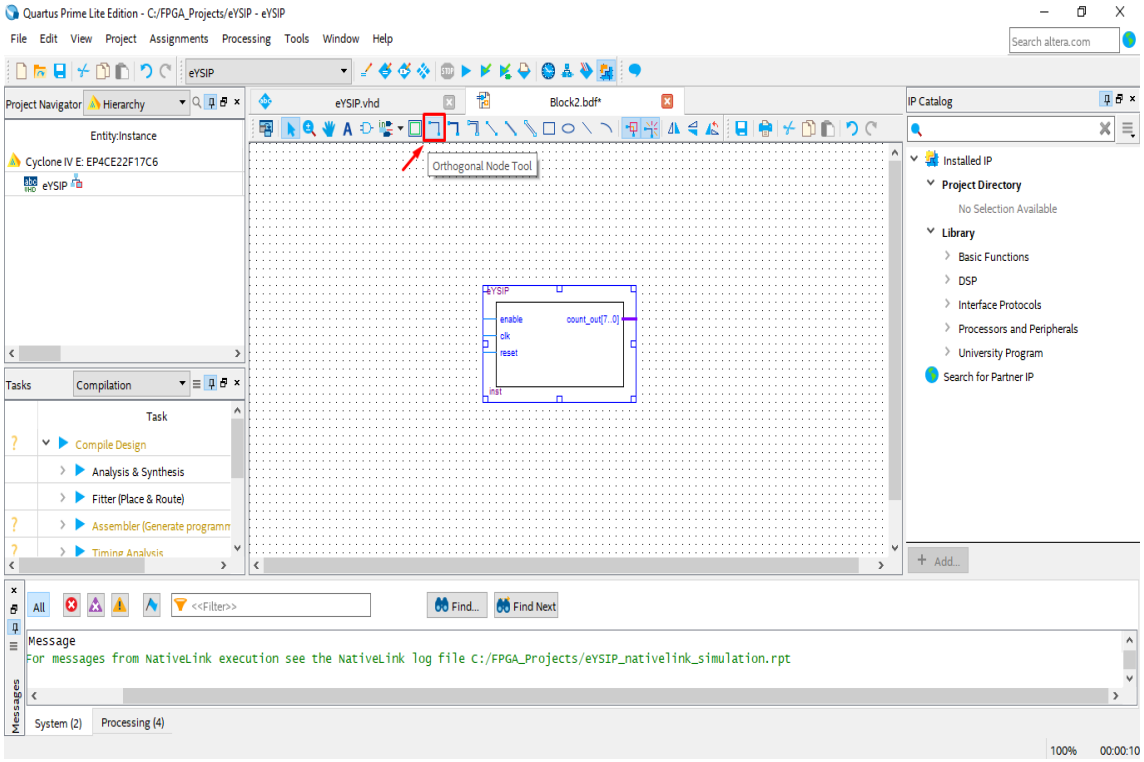
Step 3:Import your symbol files in your block diagram. Click on the symbol tool



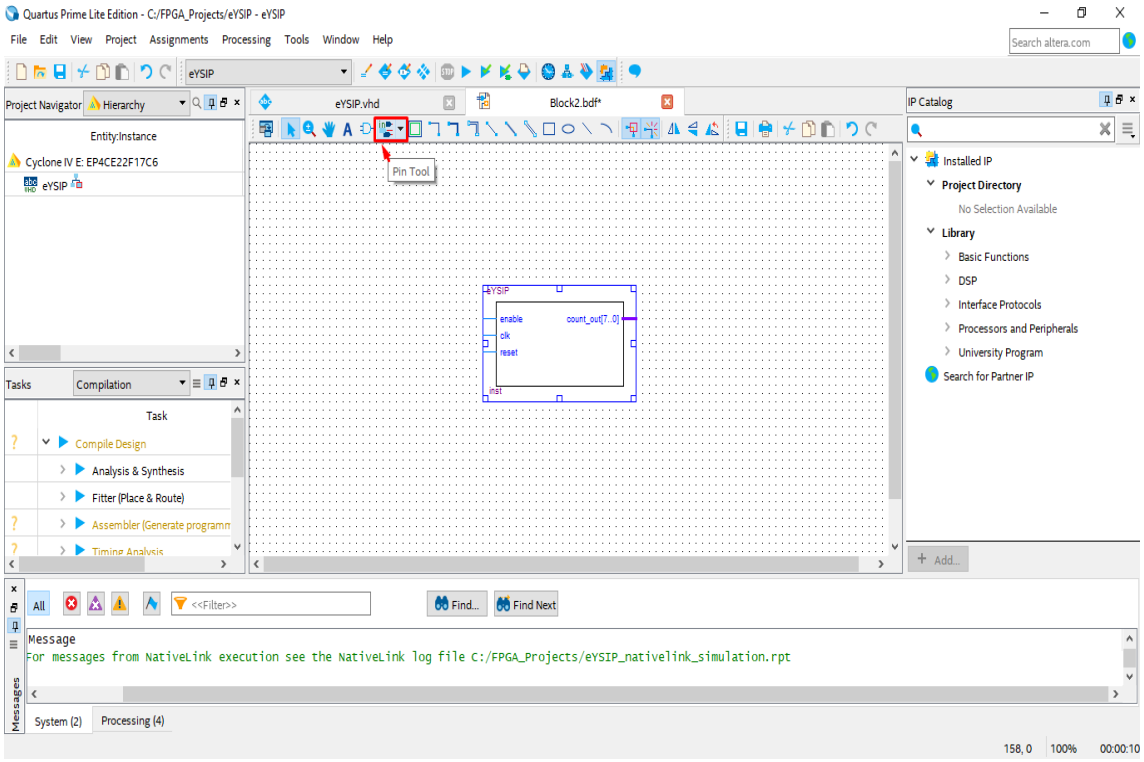
Step 4:Now, select your project and click on OK. You can place the symbol anywhere in your layout. Note: You can place many duplicates of the same symbol by clicking repeatedly, until you press esc.



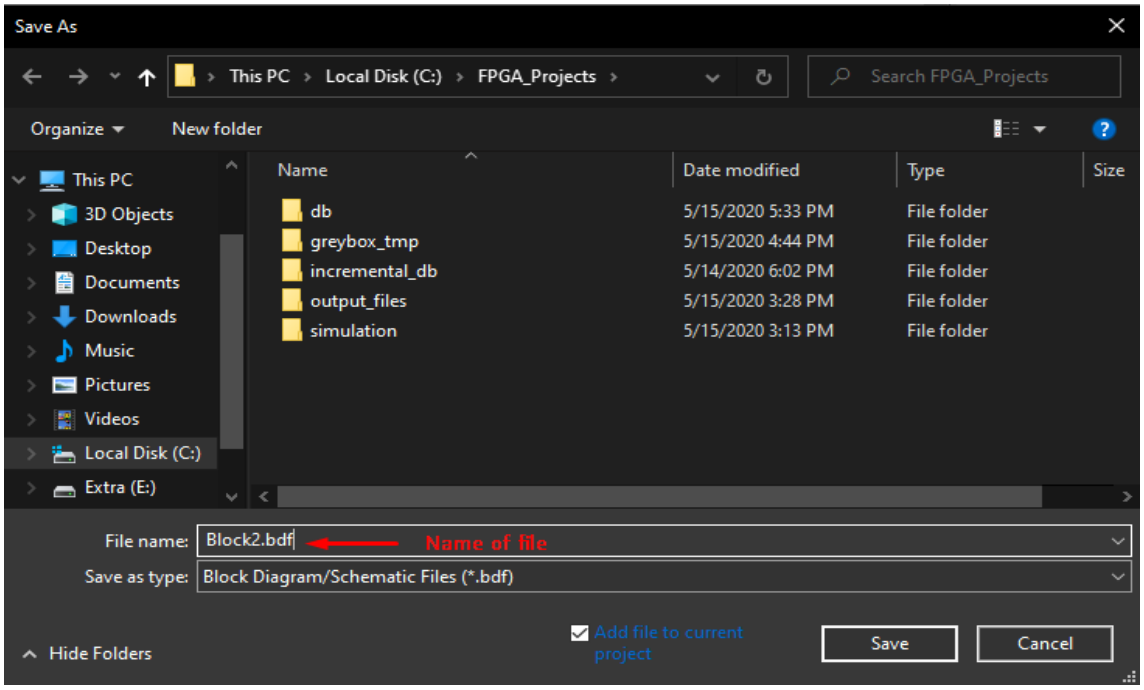
Step 5:You can use the Orthogonal Node Tool for making connections



Step 6:You can use the Pin Tool for assigning I/O ports

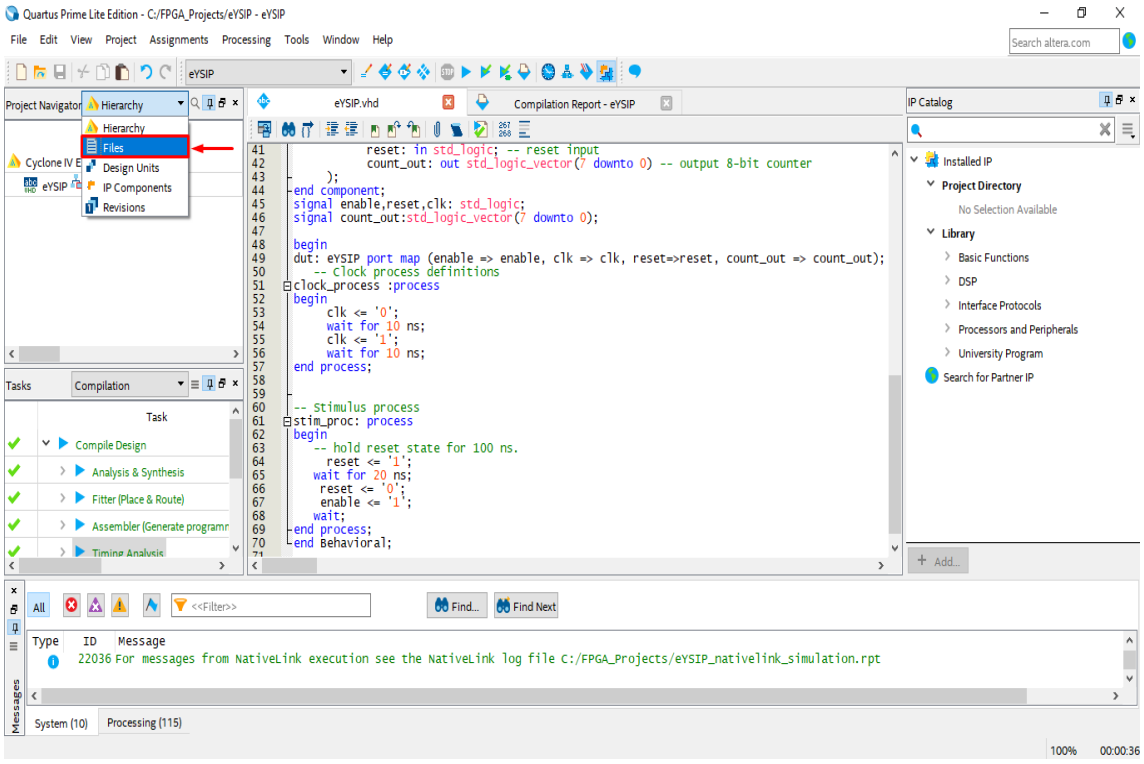


Step 7:To save the file, click on File→Save as... Enter the name of your file and click on save.

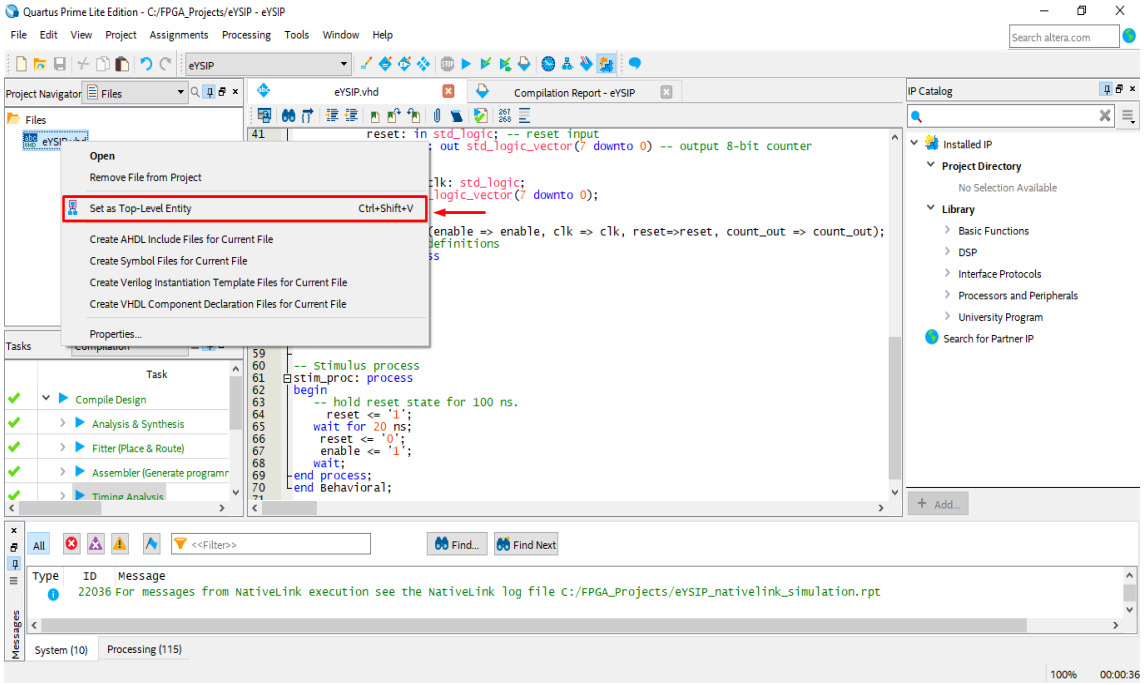


3.3 Compiling the project files

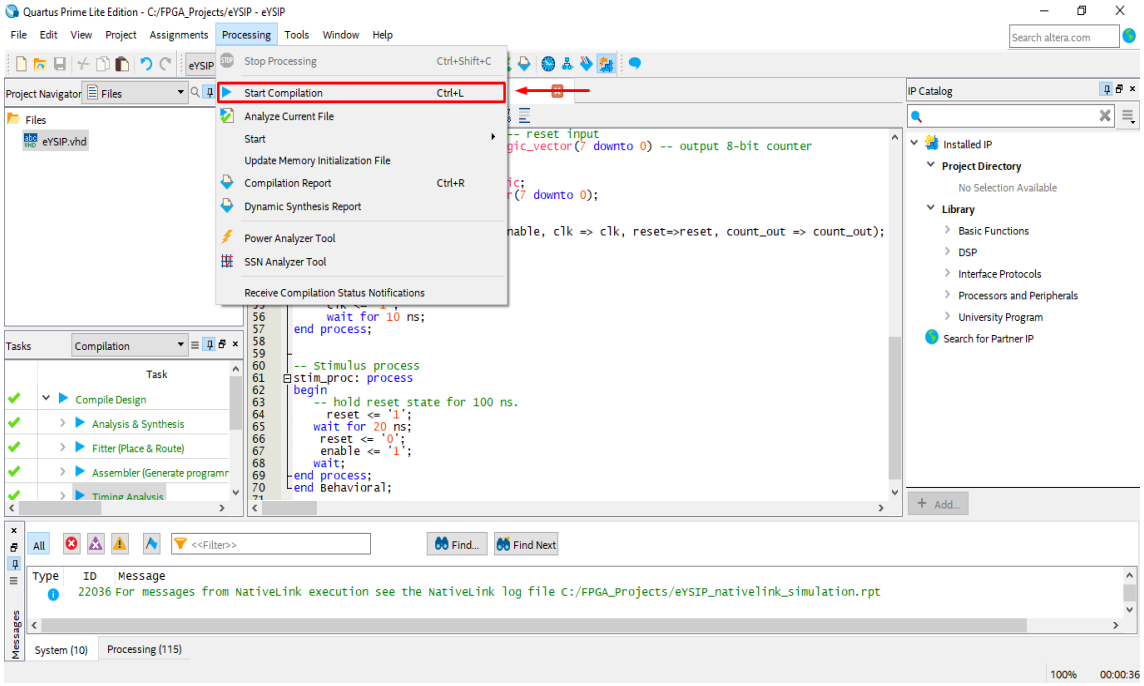
Step 1:From the project navigator select the files option



Step 2:Right click on your verilog/VHDL file and select 'set as top level entity' option



Step 3: Go to processing and select 'start compilation' option



4 Pin Assignment and Loading the design on FPGA board

4.1 Pin Assignment

Before starting with the pin assignment let's first look at how many I/O pins are provided with DE0-Nano packages. Generally FPGAs tend to have lots of pins and we can divide them into two bins: "General User Pins" and "Dedicated Pins".

DE0-Nano package contains the following General user I/O pins:

- 8 green LEDs
- 2 debounced pushbuttons
- 4-position DIP switch

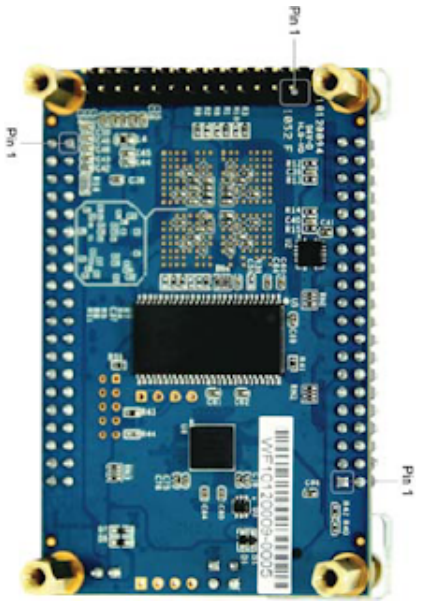
Dedicated pins includes

- Two 40-pin Headers (GPIOs) provide 72 I/O pins, 5V power pins, two 3.3V power pins and four ground pins.
- 2x13 header, A/D converter

GPIO_0_IN0	A8	1	2	D3	GPIO_00
GPIO_0_IN1	B8	3	4	C3	GPIO_01
GPIO_02	A2	5	6	A3	GPIO_03
GPIO_04	B3	7	8	B4	GPIO_05
GPIO_06	A4	9	10	B5	GPIO_07
VCC_5V		11	12		GND
GPIO_08	A5	13	14	D5	GPIO_09
GPIO_010	B6	15	16	A6	GPIO_011
GPIO_012	B7	17	18	D6	GPIO_013
GPIO_014	A7	19	20	C6	GPIO_015
GPIO_016	C8	21	22	E6	GPIO_017
GPIO_018	E7	23	24	D8	GPIO_019
GPIO_020	E8	25	26	F8	GPIO_021
GPIO_022	F9	27	28	E9	GPIO_023
VCC_3V3		29	30		GND
GPIO_024	C9	31	32	D9	GPIO_025
GPIO_026	E11	33	34	E10	GPIO_027
GPIO_028	C11	35	36	B11	GPIO_029
GPIO_030	A12	37	38	D11	GPIO_031
GPIO_032	D12	39	40	B12	GPIO_033

GPIO_1_IN0	T9	1	2	F13	GPIO_10
GPIO_1_IN1	R9	3	4	T15	GPIO_11
GPIO_12	T14	5	6	T13	GPIO_13
GPIO_14	R13	7	8	T12	GPIO_15
GPIO_16	R12	9	10	T11	GPIO_17
VCC_5V		11	12		GND
GPIO_18	T10	13	14	R11	GPIO_19
GPIO_110	P11	15	16	R10	GPIO_111
GPIO_112	N12	17	18	P9	GPIO_113
GPIO_114	N9	19	20	N11	GPIO_115
GPIO_116	L16	21	22	K16	GPIO_117
GPIO_118	R16	23	24	L15	GPIO_119
GPIO_120	P15	25	26	P16	GPIO_121
GPIO_122	R14	27	28	N16	GPIO_123
VCC_3V3		29	30		GND
GPIO_124	N15	31	32	P14	GPIO_125
GPIO_126	L14	33	34	N14	GPIO_127
GPIO_128	M10	35	36	L13	GPIO_129
GPIO_130	J16	37	38	K15	GPIO_131
GPIO_132	J13	39	40	J14	GPIO_133

GPIO_2_IN0	E15	2	1		VCC_3V3
GPIO_2_IN2	M16	4	3	E16	GPIO_2_IN1
GPIO_21	B16	6	5	A14	GPIO_20
GPIO_23	C16	8	7	C14	GPIO_22
GPIO_25	D16	10	9	C15	GPIO_24
GPIO_27	D14	12	11	D15	GPIO_26
GPIO_29	F16	14	13	F15	GPIO_28
GPIO_211	G16	16	15	F14	GPIO_210
Analog_In5		18	17	G15	GPIO_212
Analog_In7		20	19		Analog_In6
Analog_In2		22	21		Analog_In3
Analog_In0		24	23		Analog_In4
GND		26	25		Analog_In1

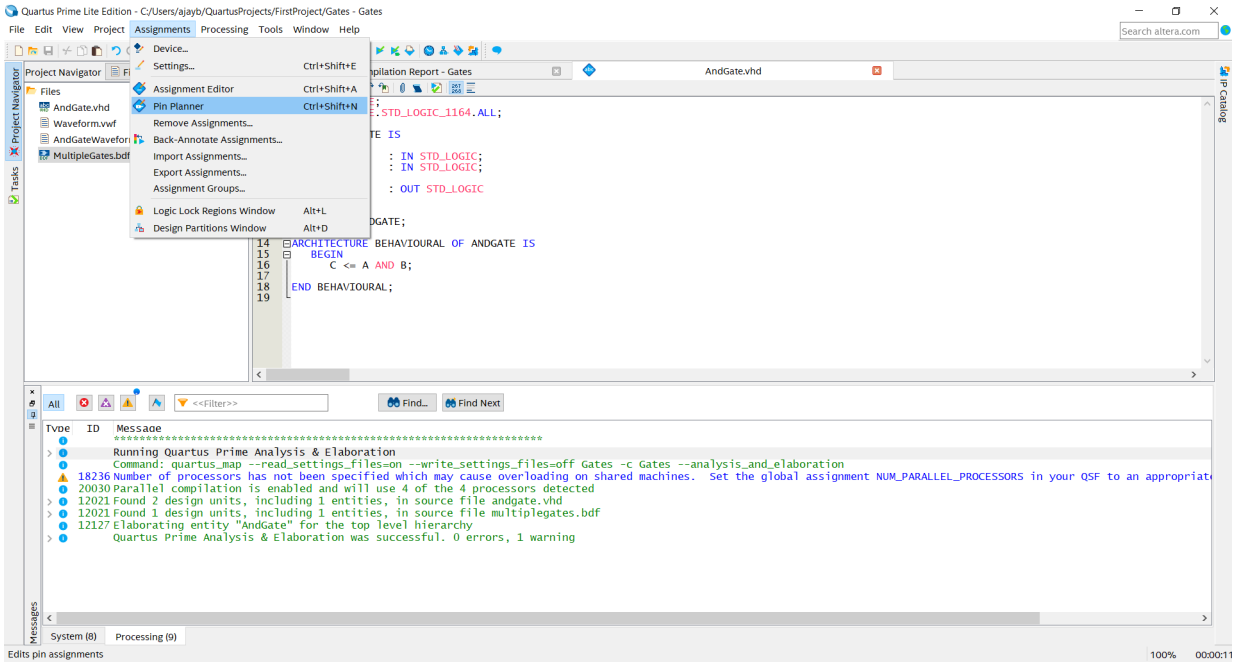


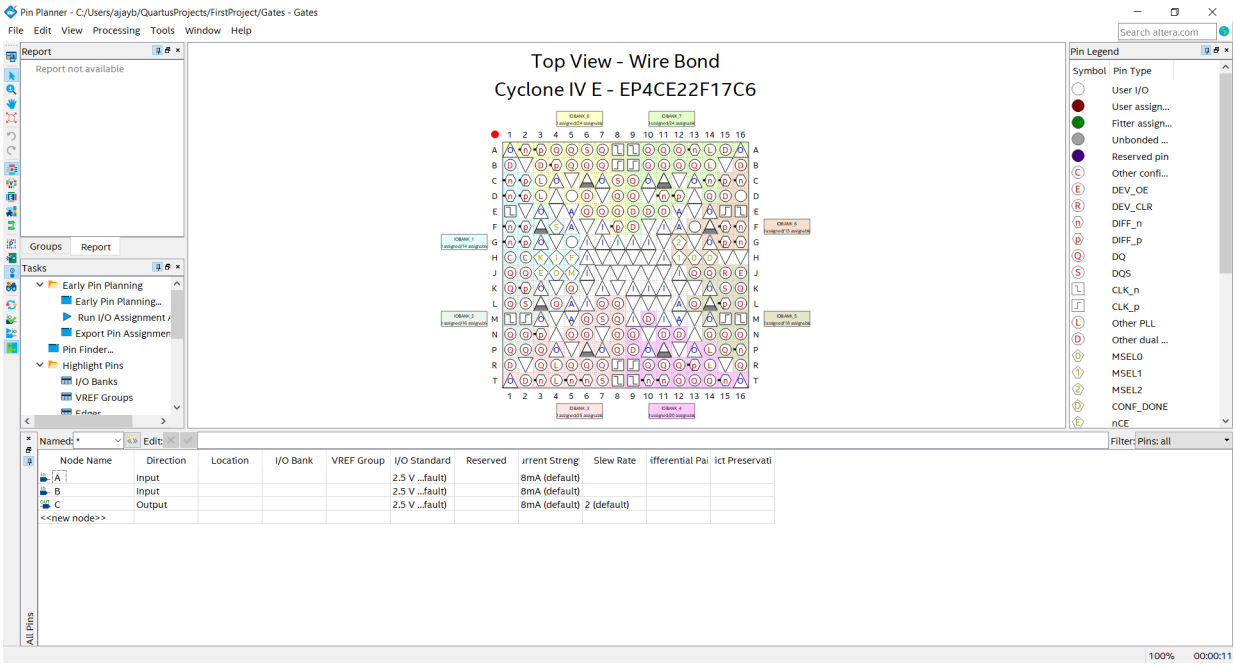
Pin Name for sliding Switches		
Signal Name	FPGA PinNo.	Description
DW[0]	PIN_M1	DIP Switch[0]
DW[1]	PIN_T8	DIP Switch[1]
DW[2]	PIN_B9	DIP Switch[2]
DW[3]	PIN_M15	DIP Switch[3] V
Pin Name for Push Button		
Signal Name	FPGA PinNo.	Description
KEY[0]	PIN_J15	Push-button[0]
KEY[0]	PIN_E1	Push-button[1]
Pin Name for LEDs		
Signal Name	FPGA PinNo.	Description
LED[0]	PIN_A15	LED Green[0]
LED[1]	PIN_A13	LED Green[1]
LED[2]	PIN_B13	LED Green[2]
LED[3]	PIN_A11	LED Green[3]
LED[4]	PIN_D1	LED Green[4]
LED[5]	PIN_F3	LED Green[5]
LED[6]	PIN_B1	LED Green[6]
LED[7]	PIN_L3	LED Green[7]

You can assign input and output (I/O) signals to package pins in your design with the help of above table. Using the Pin Planner, you can assign I/O locations, prohibit I/O locations. This process operates on the top entity in your design. This allows you to assign input and output signals to package pins before the underlying logic in the design has been developed.

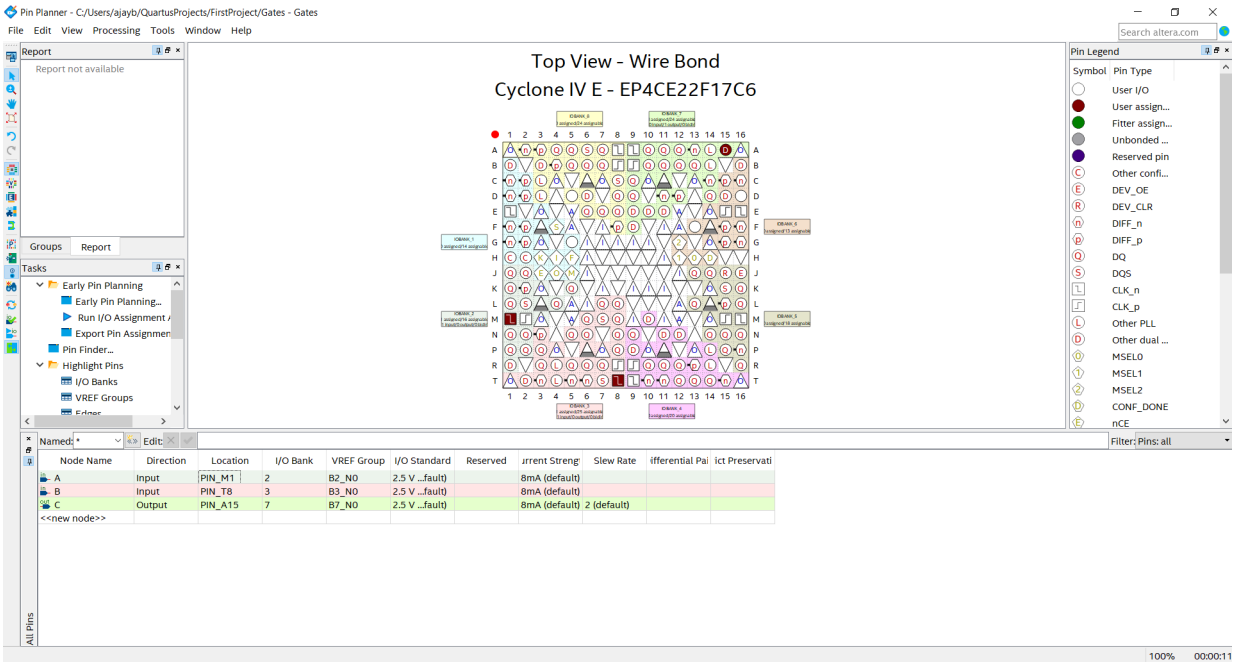
Further in this section we will map the I/O pins created in our design with the actual pins of the DE0-Nano. Before starting pin assignment make sure to do Analysis Elaboration check.

1. Click on Assignments and select Pin Planner, which opens the Pin Planner, the Pin Planner shows the I/O that we have created in our design.

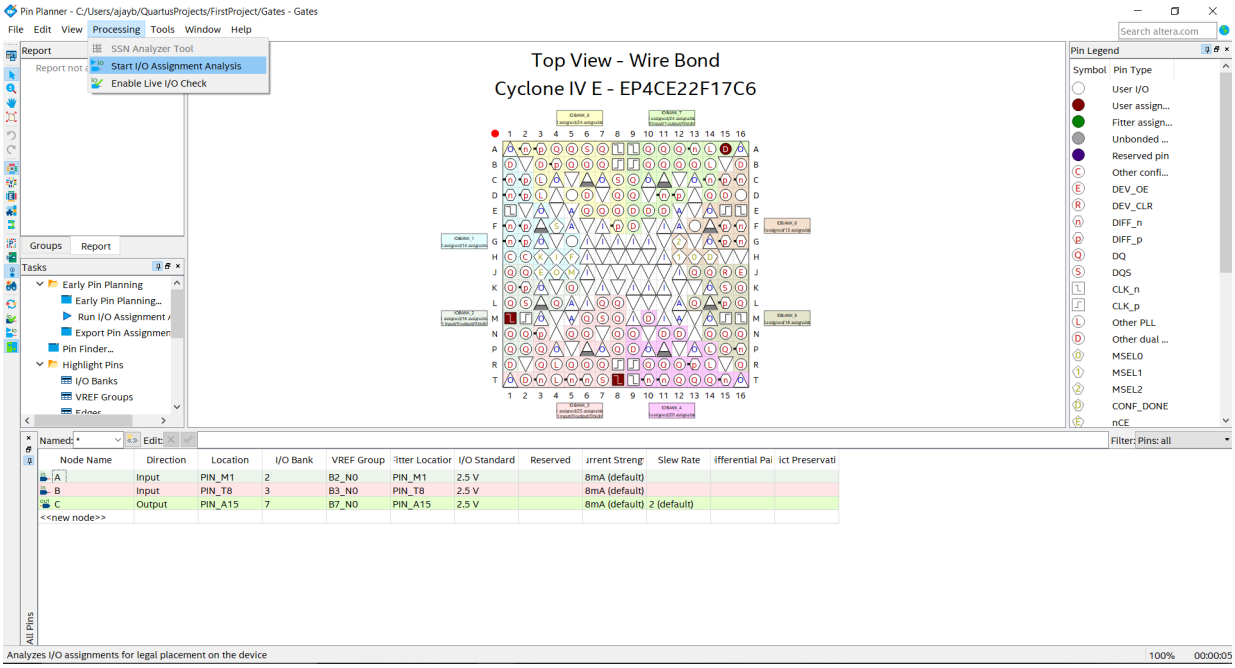




2. We will select LED[0] as the output of the AND gate and Slide switch 0 and 1 for input signal A and B.



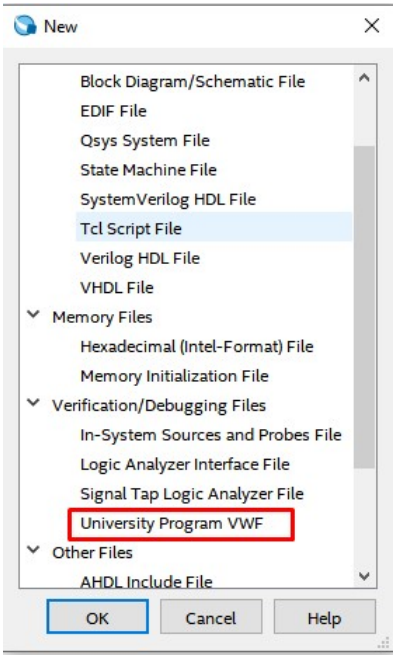
3. Then click on Start I/O analysis



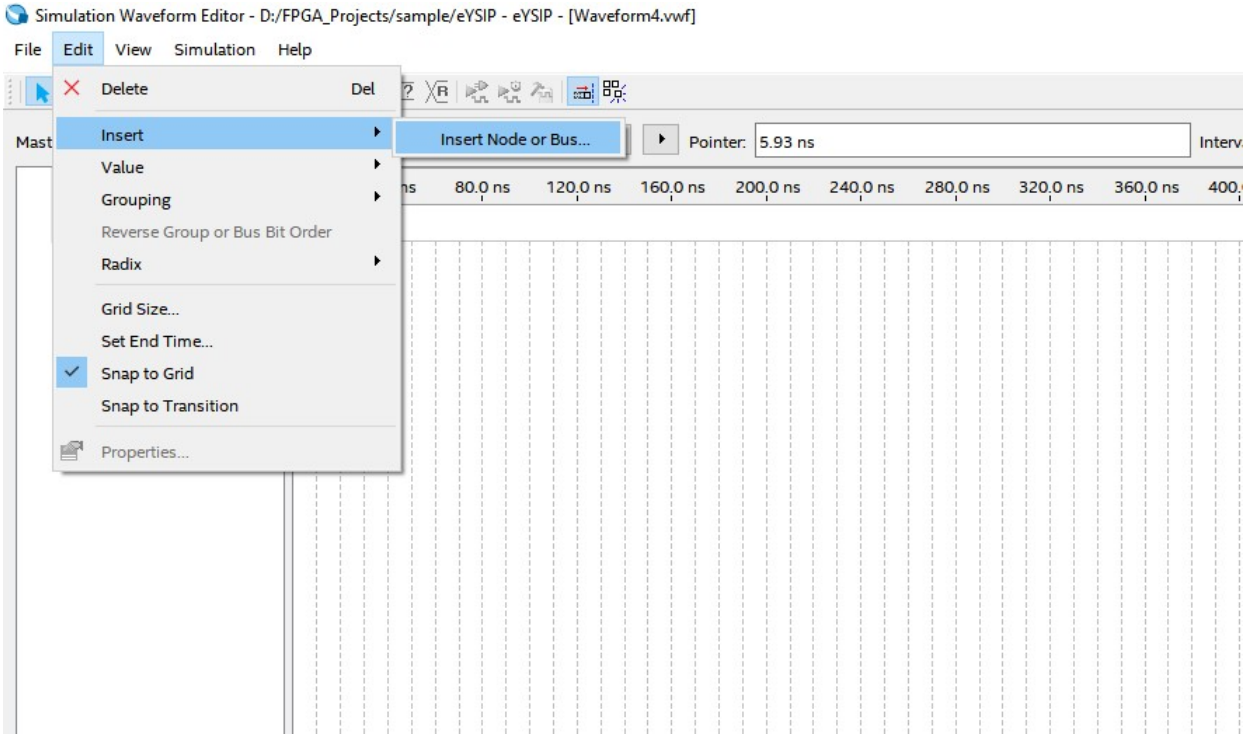
5 Verification by simulation(using ModelSim)

5.1 Without TestBench

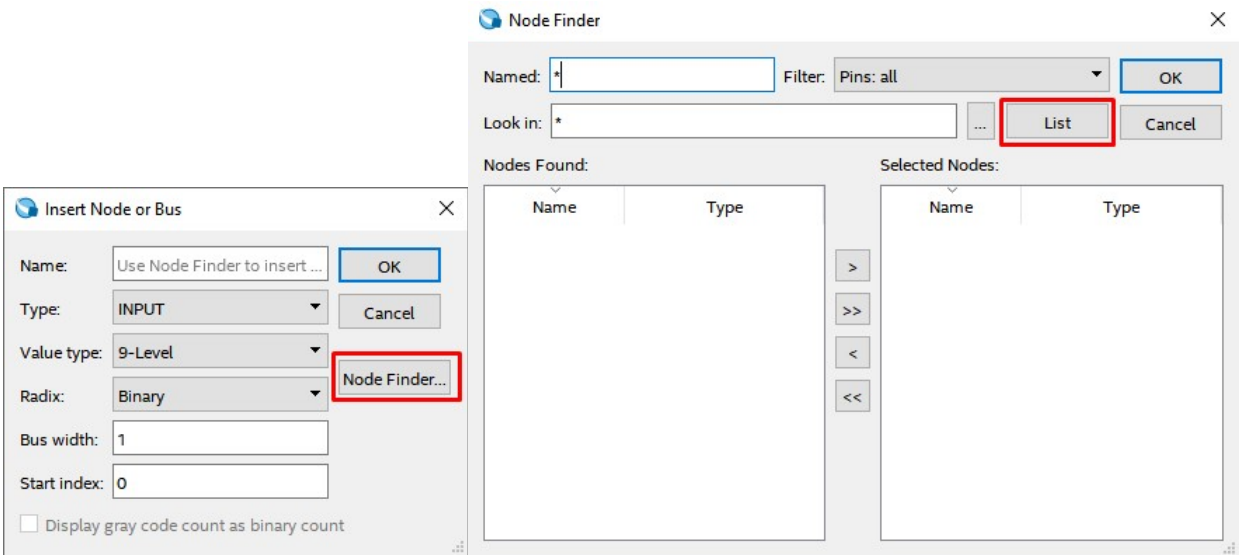
Step 1: Click on file and then click on New. From the list, Choose University Program VWF



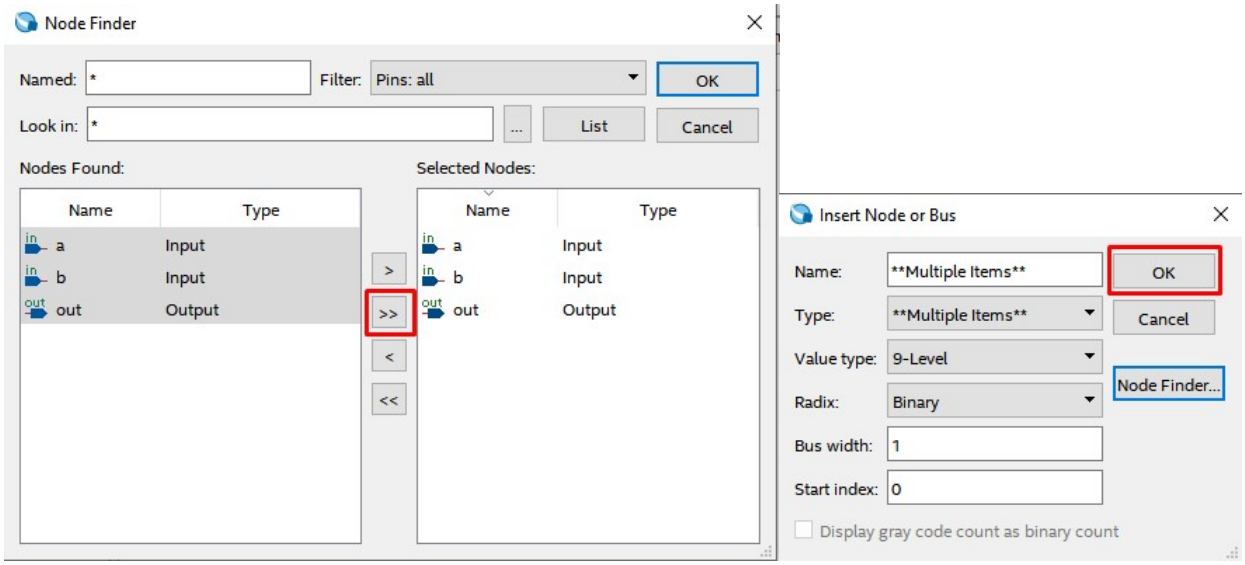
Step 2: In the New window, Go to Edit→Insert→Node or bus



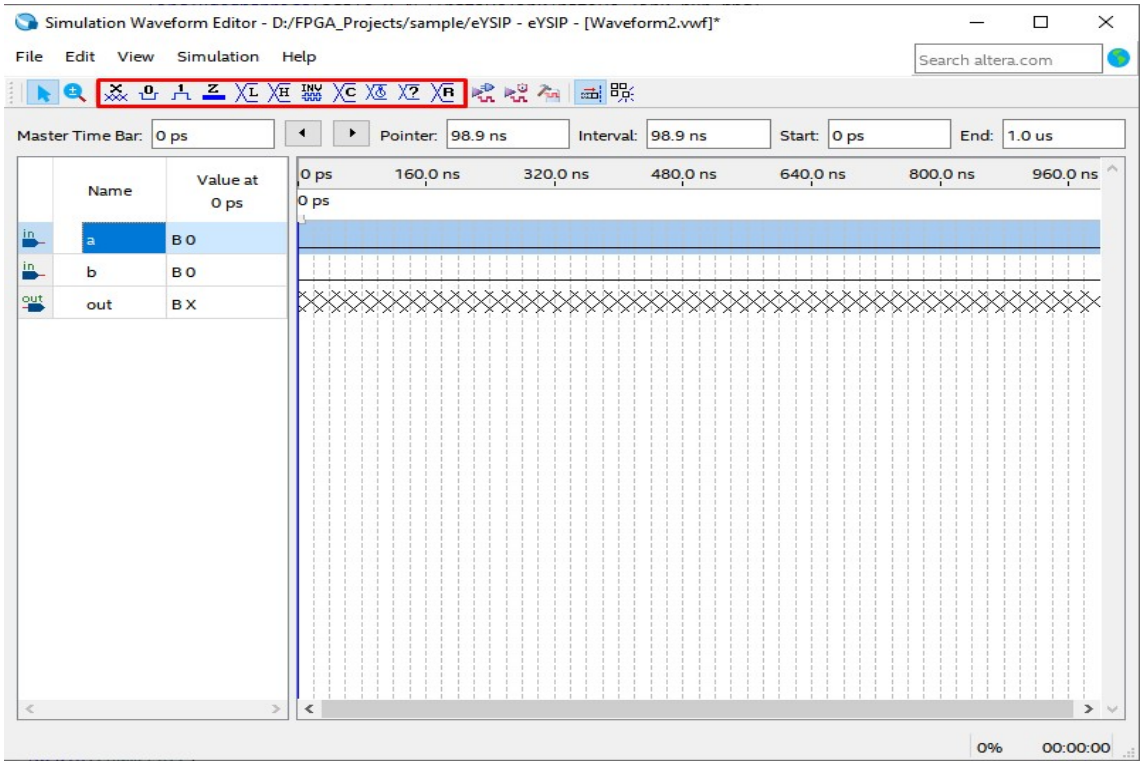
Step 3: Click on Node Finder, in the new window, click on list



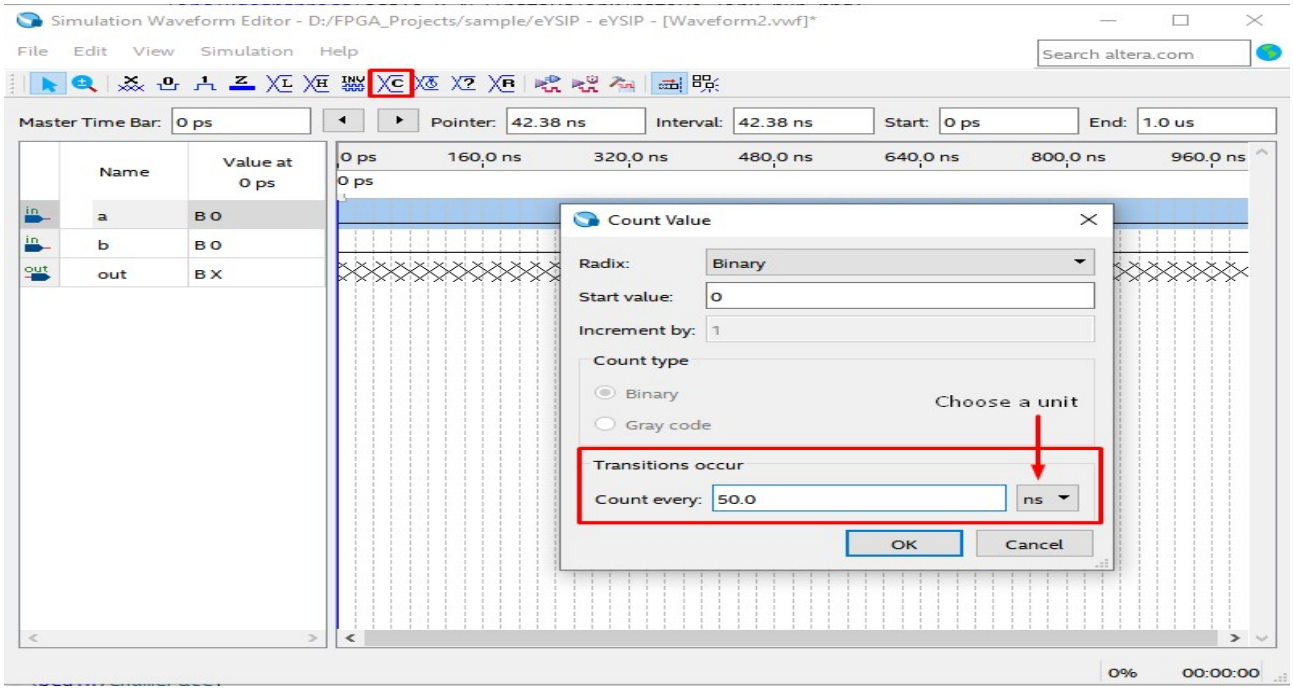
Step 4: Click on the highlighted button to add all the Nodes. Now click on 'OK' and then 'OK' again



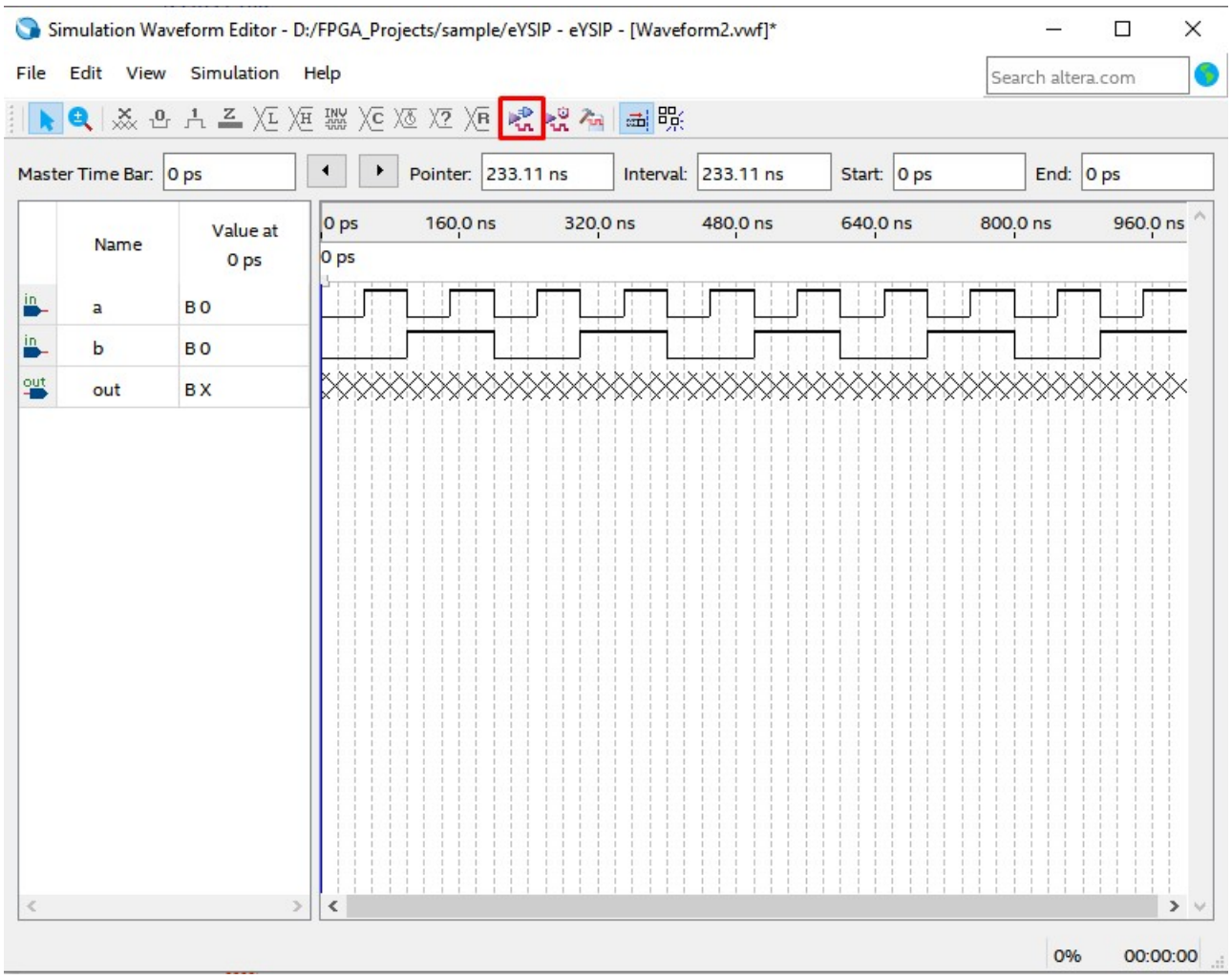
Step 5: Choose any of the input node and specify its value by choosing any of the options that are highlighted

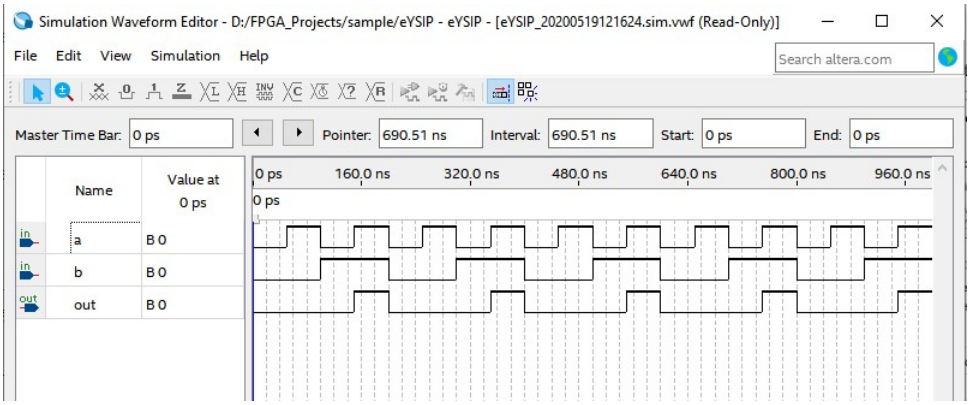


Step 6:For example, to give the alternating 'a' input, we first select the a node, then click on the count input type, and then specify the transition interval



Step 7:Similarly, specify the input patterns for all the inputs and then click on the simulate icon. If a save prompt appears, then save the file with a user defined name. Once the simulation process completes, the waveform will be displayed

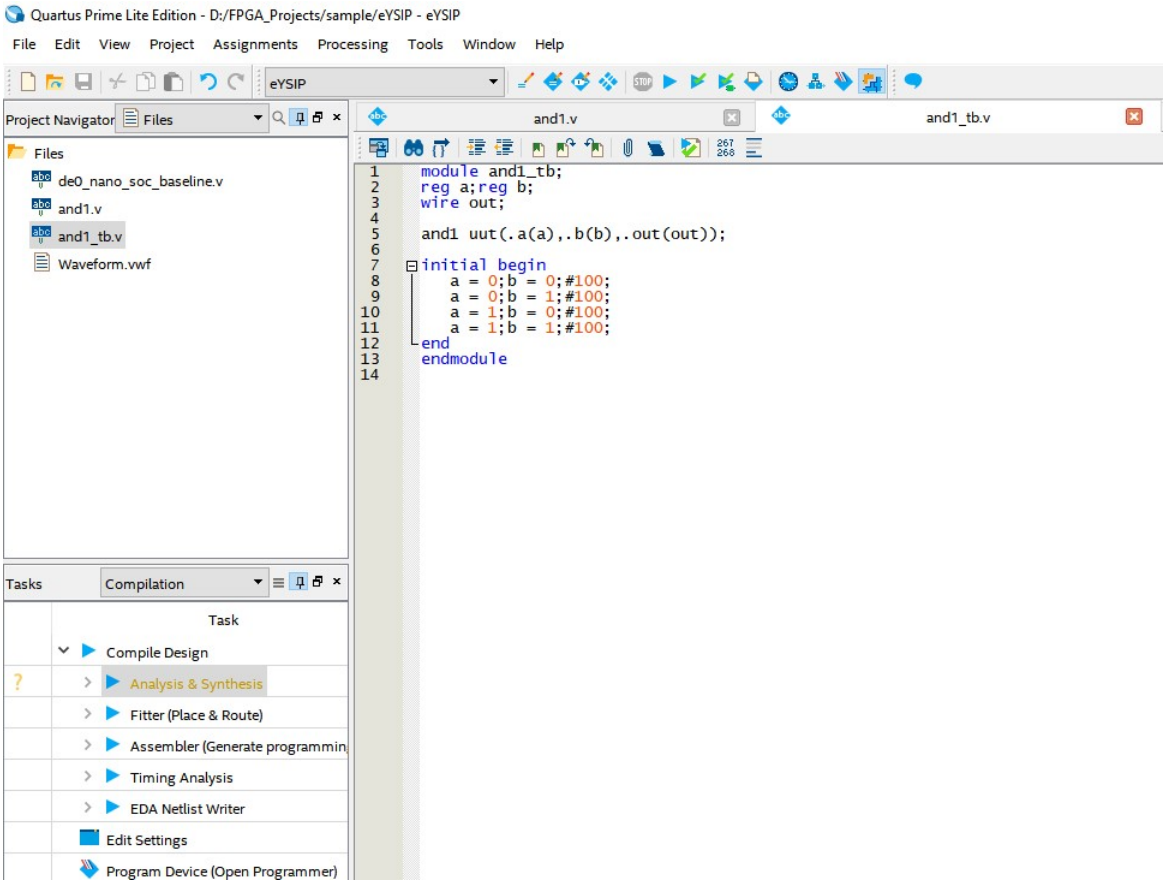




5.2 With Testbench and NativeLink

Altera Quartus II software allows the user to launch Modelsim-Altera simulator from within the software using the Quartus II feature called NativeLink. It facilitates the process of simulation by providing an easy to use mechanism and precompiled libraries for simulation.

Step 1 : Create a new verilog/VHDL file ,Add the testbench code and save it in the same project directory.



Testbench code for an AND gate:

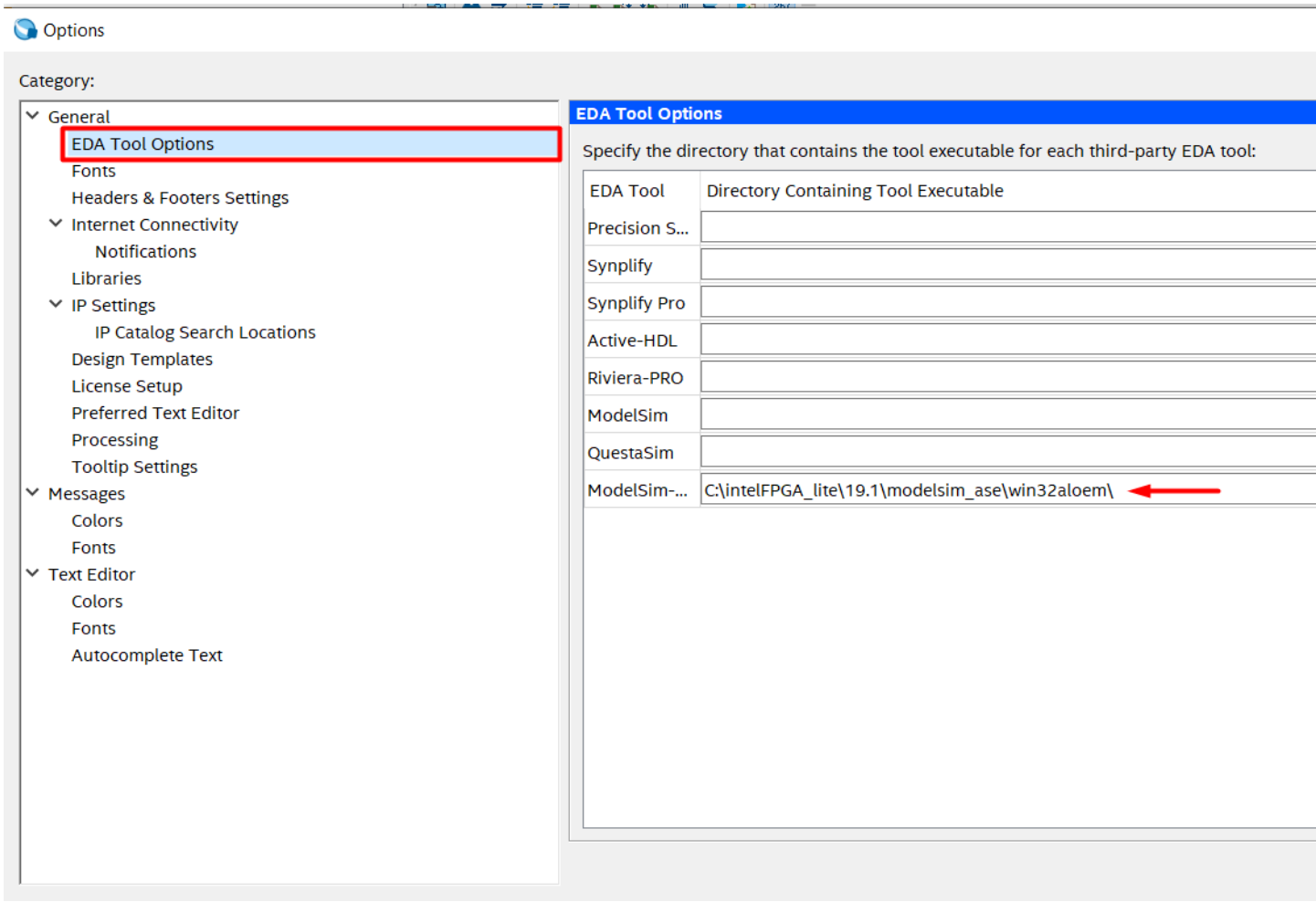
```
module and1_tb;
reg a;reg b;
wire out;

and1 uut(.a(a),.b(b),.out(out));

initial begin

    a = 0;b = 0;#100;
    a = 0;b = 1;#100;
```

a = 1;b = 0;#100;
a = 1;b = 1;#100;

end
endmodule

Step 2: Specify the path to Modelsim Altera:

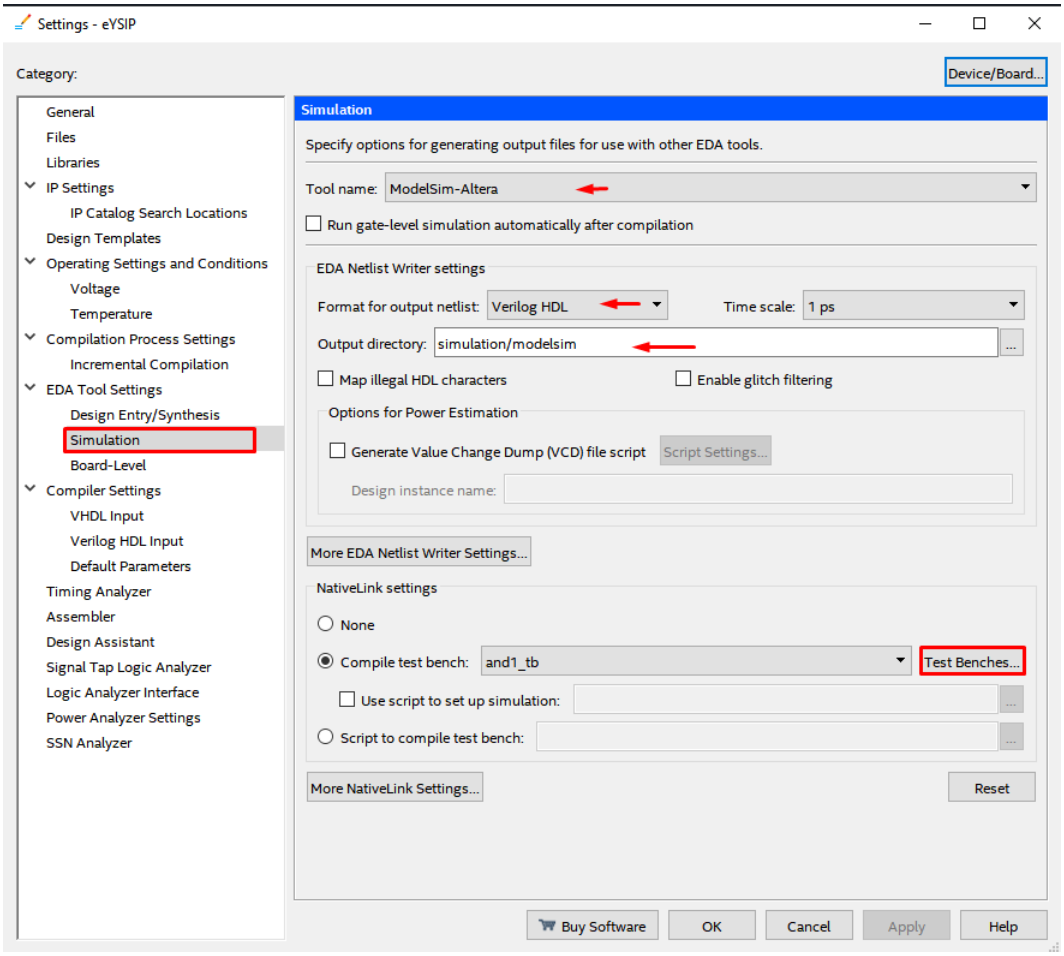
1. Go to the menu Tools → Options
2. In the 'General' category, select 'EDA Tool Options'.
3. A dialogue box appears, where you can specify the location path of the Modelsim-Altera executable file.And click 'OK'.



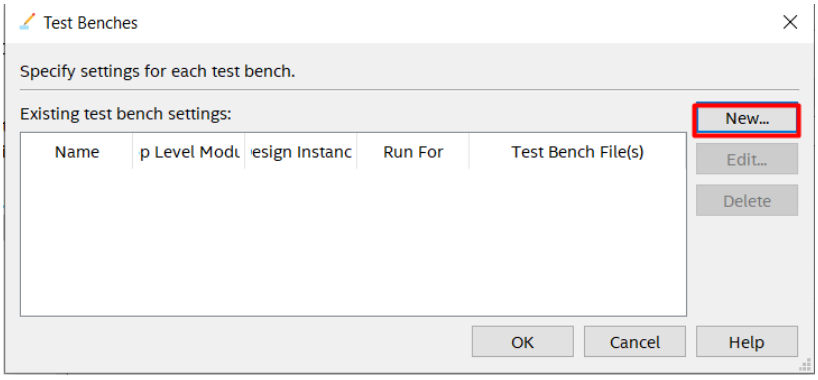
Step 3 : NativeLink Settings to configure Modelsim-Altera:

1. Go to the menu Assignments → Settings.
2. Under 'EDA Tool Settings' choose 'Simulation'. The dialogue box for simulation appears.

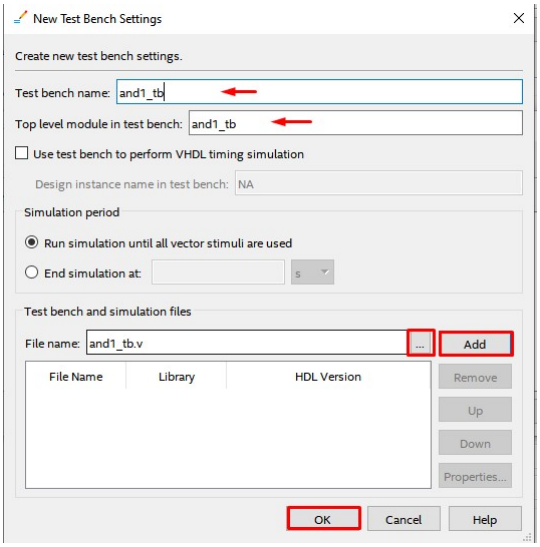
3. For Tool Name, choose 'Modelsim-Altera'.
4. Select 'VHDL'(or Verilog/systemVerilog) as the 'Format for Output Netlist'.
5. Select 'simulation/modelsim' as the 'Output Directory.
6. Under NativeLink Settings, Choose 'Compile test bench'. Then click on 'Test Benches'.All these changes are indicated in the below figure.



7. A new window appears select 'New'

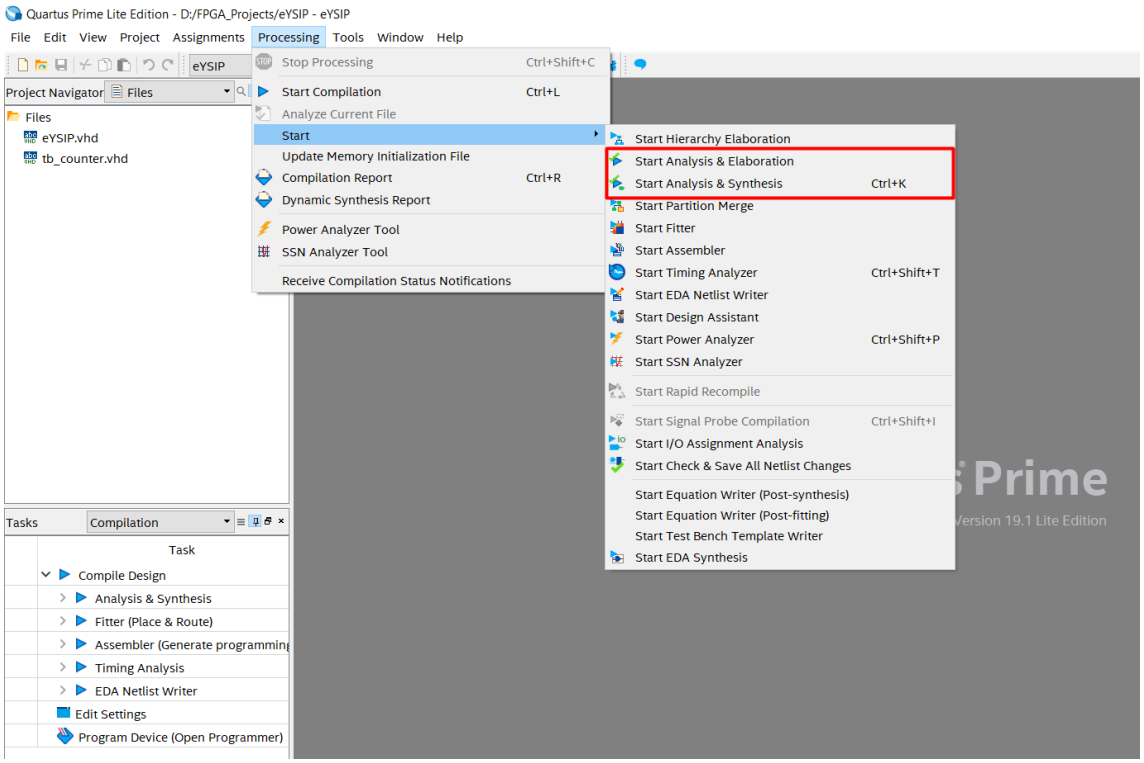


8. Another window appears with the name 'New Test Bench Settings'.
9. Enter the 'Test Bench Name' and 'Top Level Module in test bench'.
10. Add the test bench file and then click 'OK'.

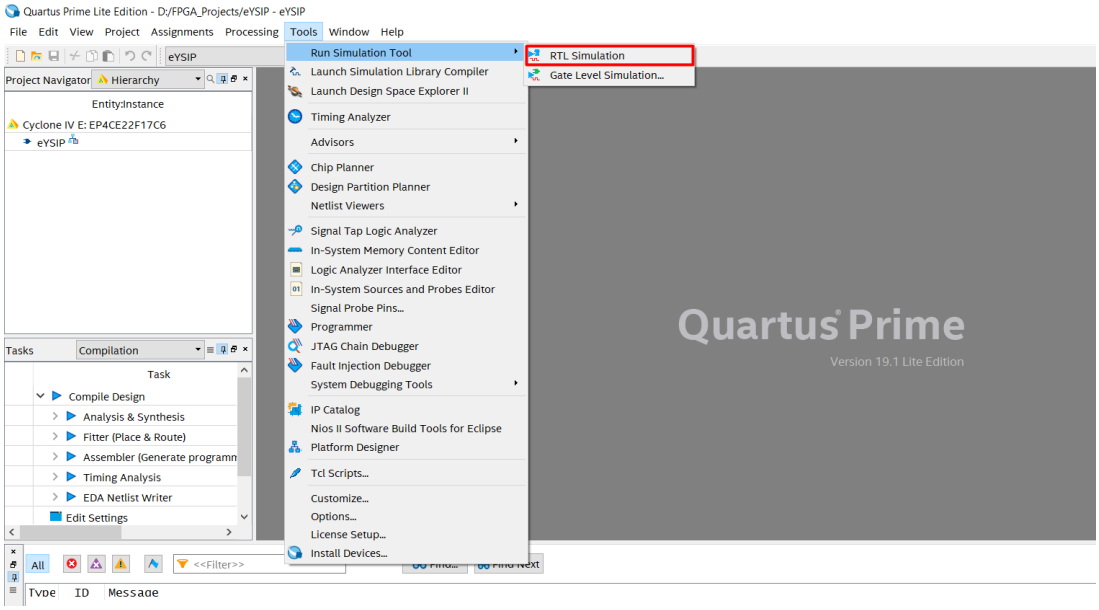


Step 4 : Functional Simulation using NativeLink Feature:

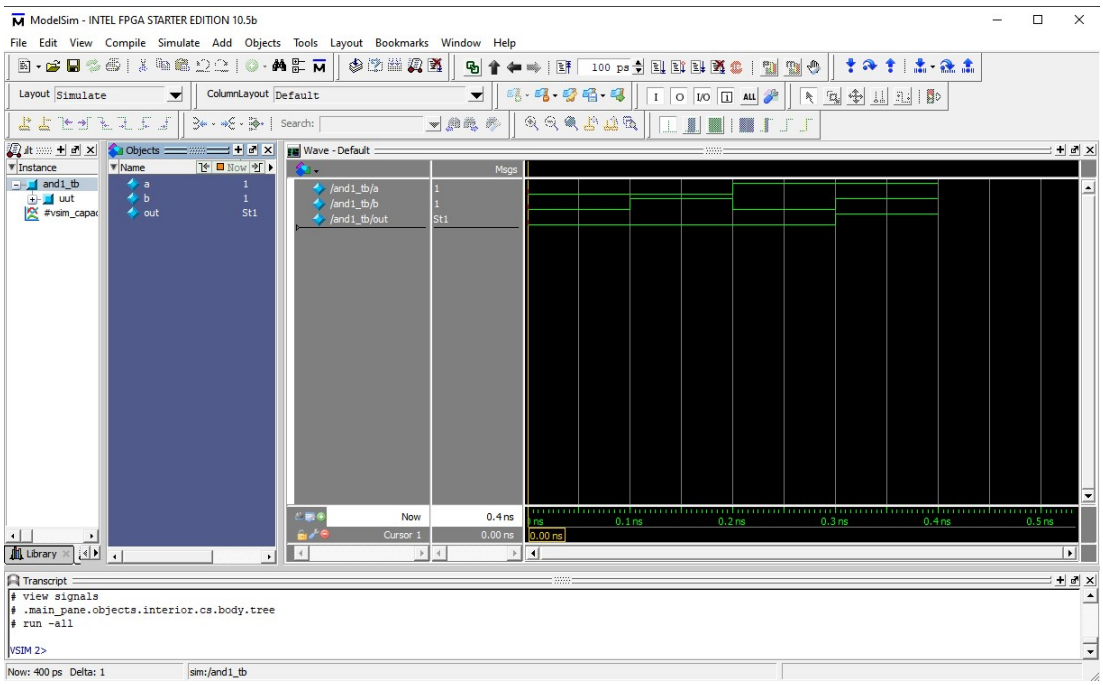
1. Goto menu 'Processing', select 'Start', and then click 'Start Analysis & Elaboration'.After this click on 'Start Analysis & Synthesis 'on the same drop box .These step checks the error and collects all file name information and builds the design hierarchy for simulation.



2. Goto menu 'Tools', click on 'Run Simulation Tool', and then click "RTL Simulation" to automatically run the EDA simulator(ModelSim-Altera) and to compile all necessary design files.



3. Finally ModelSim-Altera tool opens with simulated waveforms.



6 Timing Analysis using TimeQuest

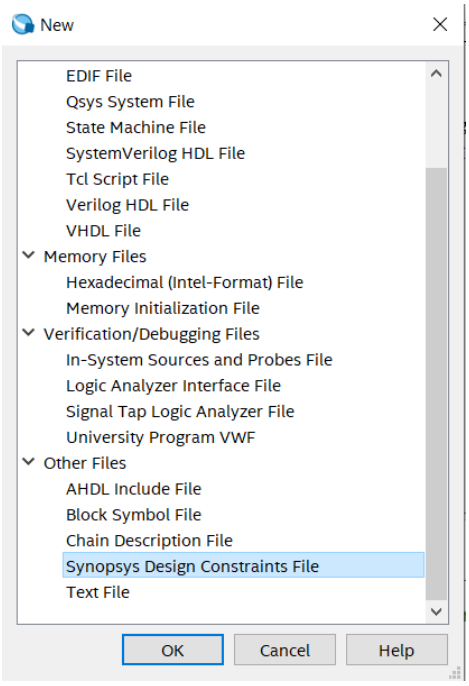
6.1 Introduction to timing analysis

The process of analyzing delays of a logic circuit to determine the condition under which the given circuit operates reliably is known as Timing Analysis. It can be used to calculate delay for every possible logical path in the design. It can also be used to determine the maximum operational frequency for the circuit.
We can use Timing Analyzer in Quartus to perform a detailed analysis of our FPGA design. This ensures that the specified timing constraints were properly passed to the implementation tools.

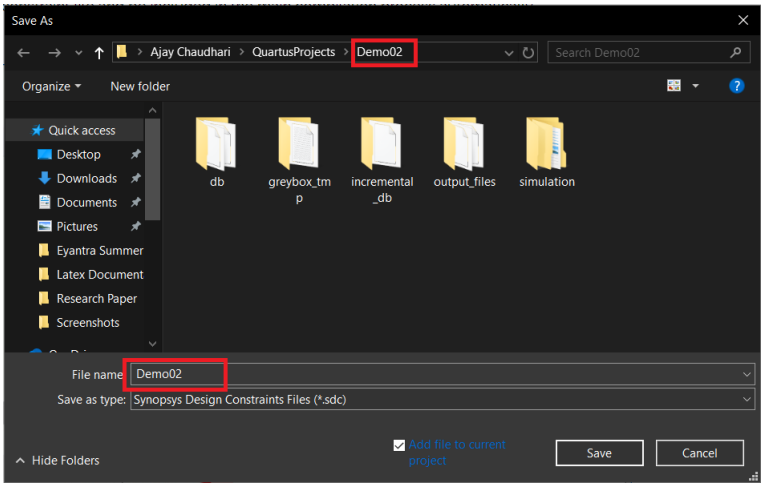
6.2 Creating a Synopsis Design Constraints file

Now that we know the importance and what actually timing analysis,let's create a Synopsis Design Constraints or an SDC file and then we have to add it to our project as the main timing constraints file.
To create a new *.sdc file

1. Click on New file under the File menu
2. Select the Synopsis Design Constraints File from under the Other Files section.



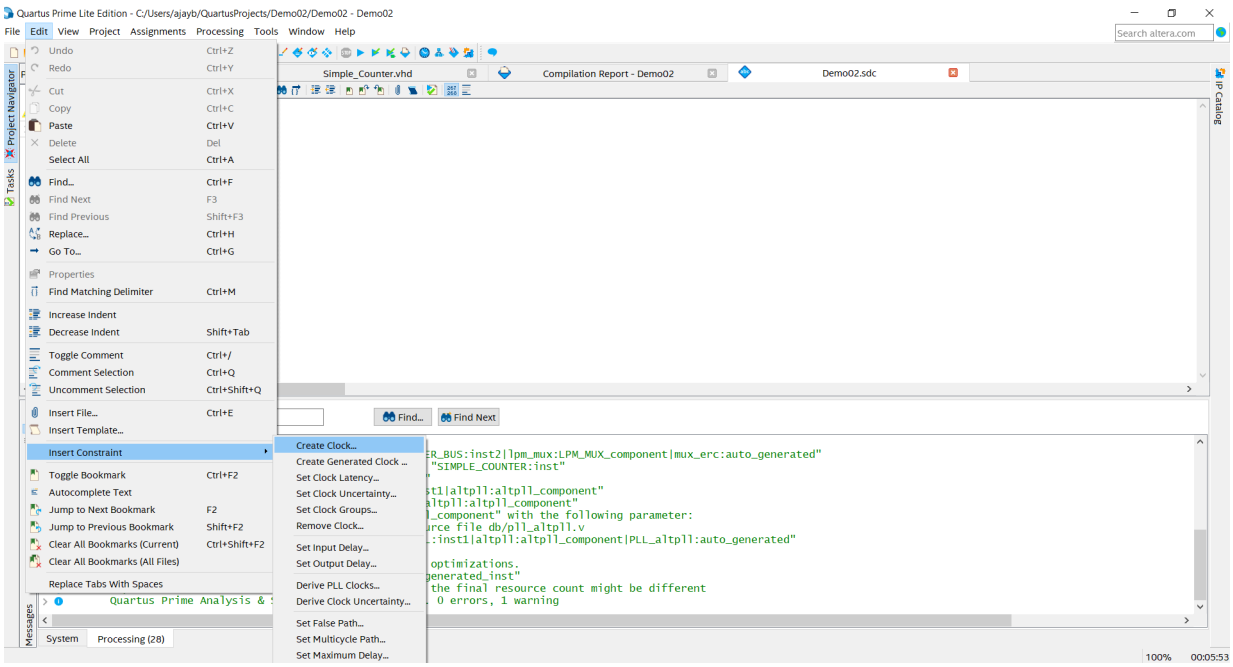
3. Click ok then save this file with the same name as our top-level entity file name So that this file will include in the main compilation process automatically, you can name the file differently but then you have to manual add it to Settings page.



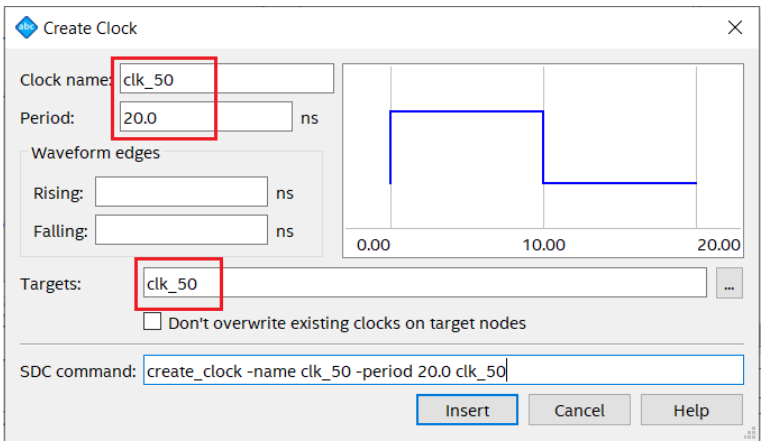
6.3 Adding Timing Constraints

Adding a time constrainst is a necessary step but simple design doesn't require timing constraint but it is recommended to add one to avoid unnecessary warning message from our compilation process.
To add a timing constraint follow the steps

1. Clicking on Insert Constraint under the Edit menu.

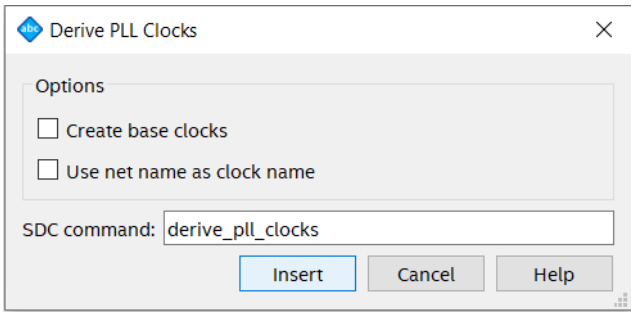


2. There are various types of constraints that could be used in our design, because all the timing related activities involves a clock.
   To create a clock for timing analysis, Select Create Clock from within Insert Constraint under the Edit menu.

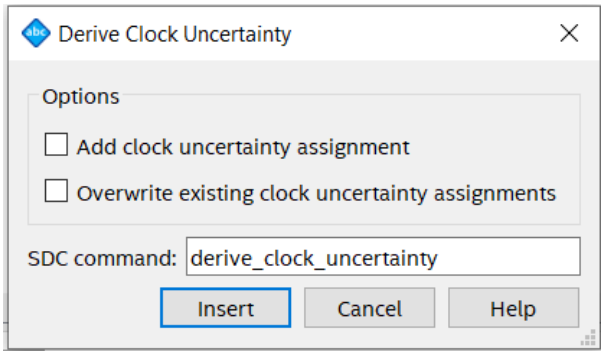


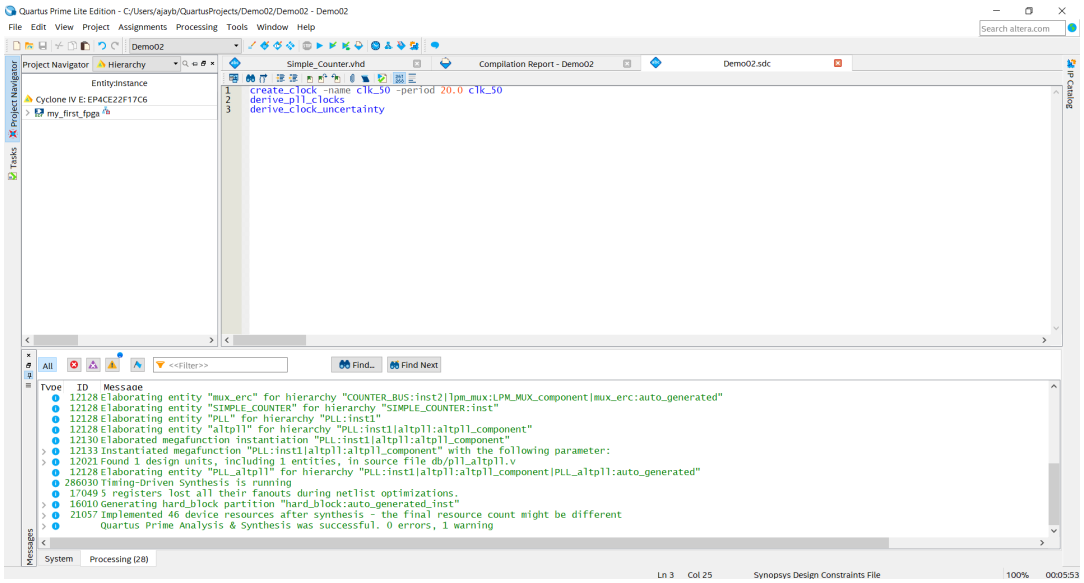
Enter the clock frequency in the period section, for example for a 50MHz clock period is 20ns.Then Click on Insert.

3. Insert Constraint again and this time select Derive PLL Clocks to generate a constraint for the PLL clock that has been derived from our main input clock. Uncheck the box as shown in the figure



4. Insert Constraint again and this time select Derive Clock Uncertainty and save the file.

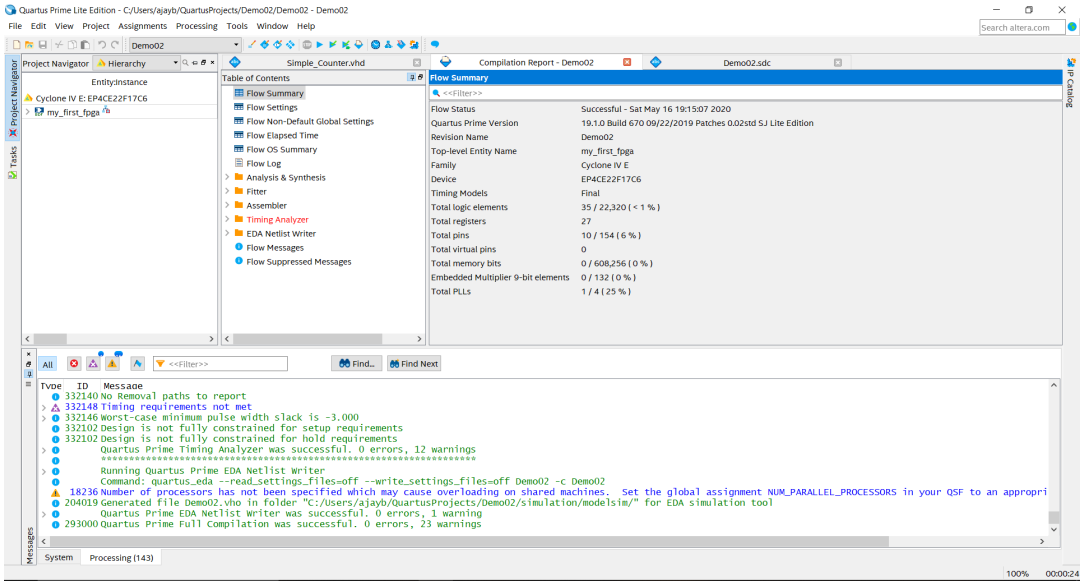




6.4 Running Full Compilation and TimeQuest Analysis

Now all the steps to set timing constraint are done. Now we can run the full compilation process. This can be done by clicking on

This step can take some time depending upon the complexity of your design and your system specification.



TimeQuest Timing Analysis uses Synopsis Design Constraint timing analysis file that we just created to generate a report. To start analysis click on the Timing Analyzer in table of

content then click on the clocks and then right click on the clock on which analysis has to be done and select Report Timing (in Timinng Analyzer UI)