Quick Start Guide
For Quartus Prime Lite Software & ModelSim

This Quick Start Guide
will walk you through from
installation to simulation using Modelsim
of your design
in Quartus Prime Lite Software

Mentors
Prasad T
Aditya Gudla
Simranjeet Singh

Interns
Ajay Chaudhari
Chethan T Bhat
Ritvik Tiwari
Karthik A Shet

Table of Content

1 Introduction to Intel Quartus Prime Lite Edition

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.

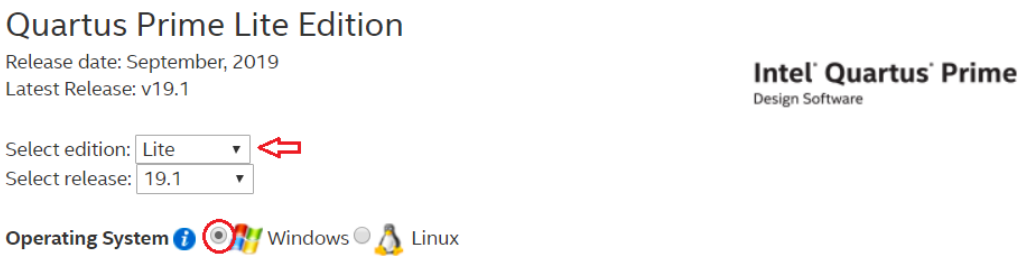
2 Installing Quartus and ModelSim

In this part, we will walk through the process of installing the software Quartus Prime Lite Edition Version 19.1. You can find the link for the installtion here.
System Requirement:

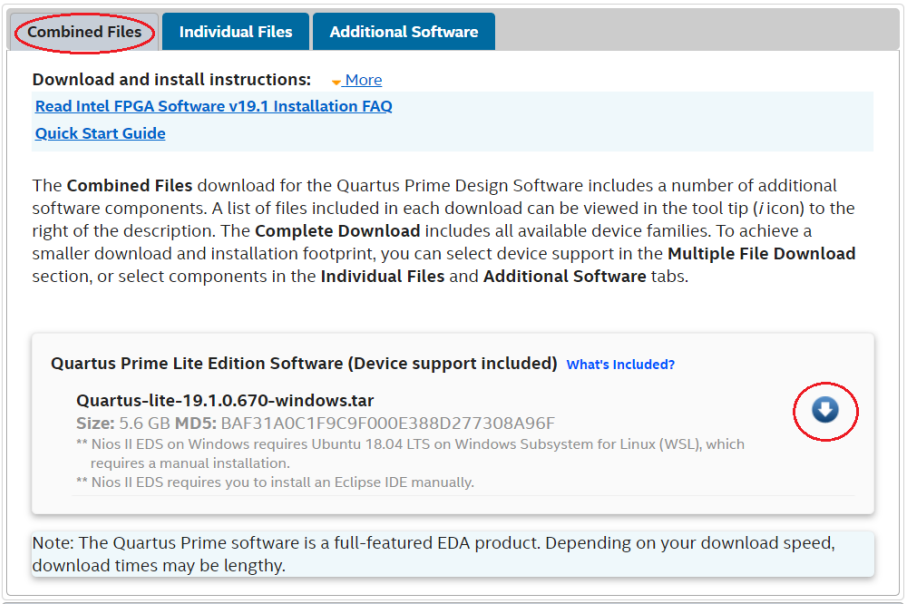
1. A full installation of the Intel FPGA Complete Design Suite v19.1 requires approximately 14GB of available disk space on the drive or partition where you are installing the software.

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.

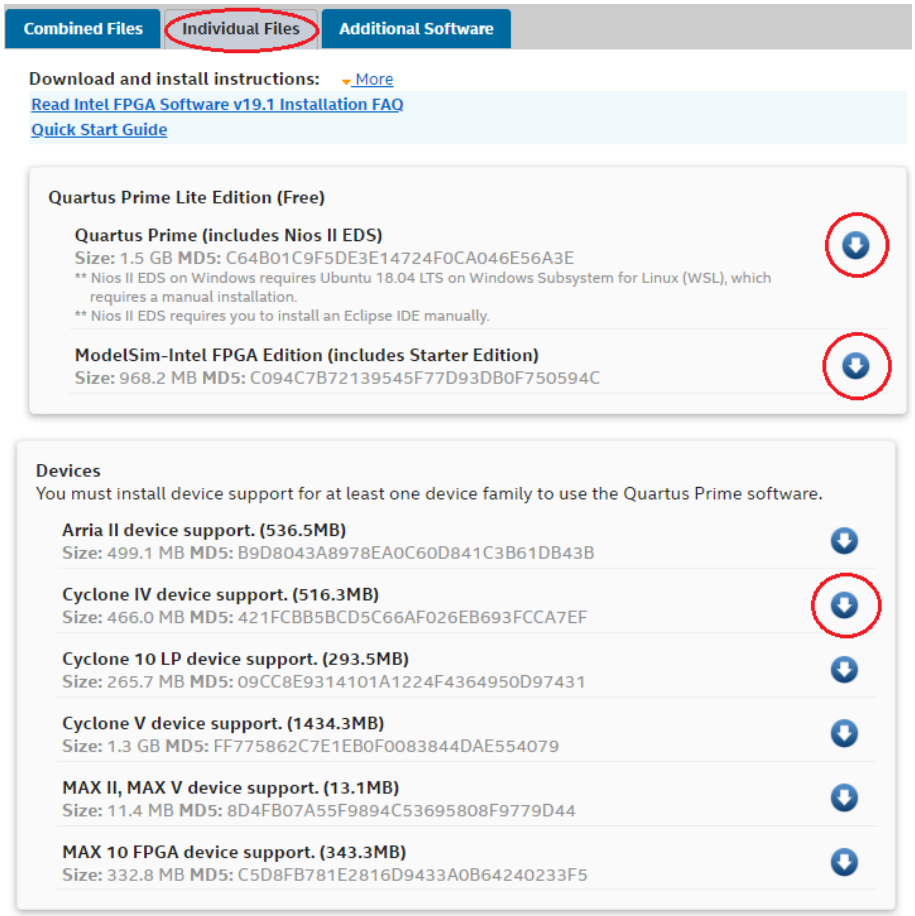


2.1 Download using Complete Software Package



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.

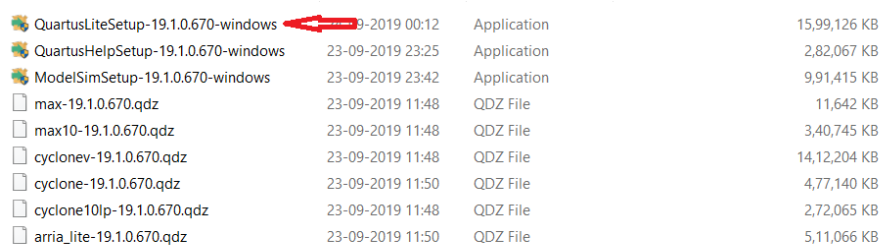
2.2 Download using Individual Software Package



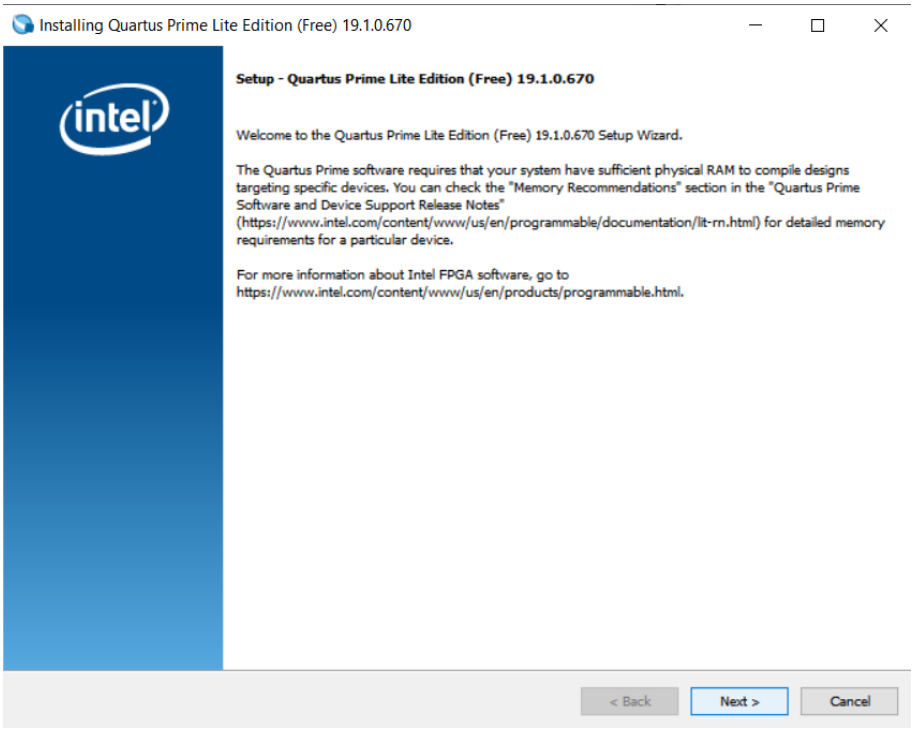
1. In Individual Files download, download both Quartus Prime lite edition and ModelSim, you also need to download files to get support for a particular device family.
2. Select from the below option which are necessary.
   (a) Arria II device support.
   (b) Cyclone IV device support.
   (c) Cyclone 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

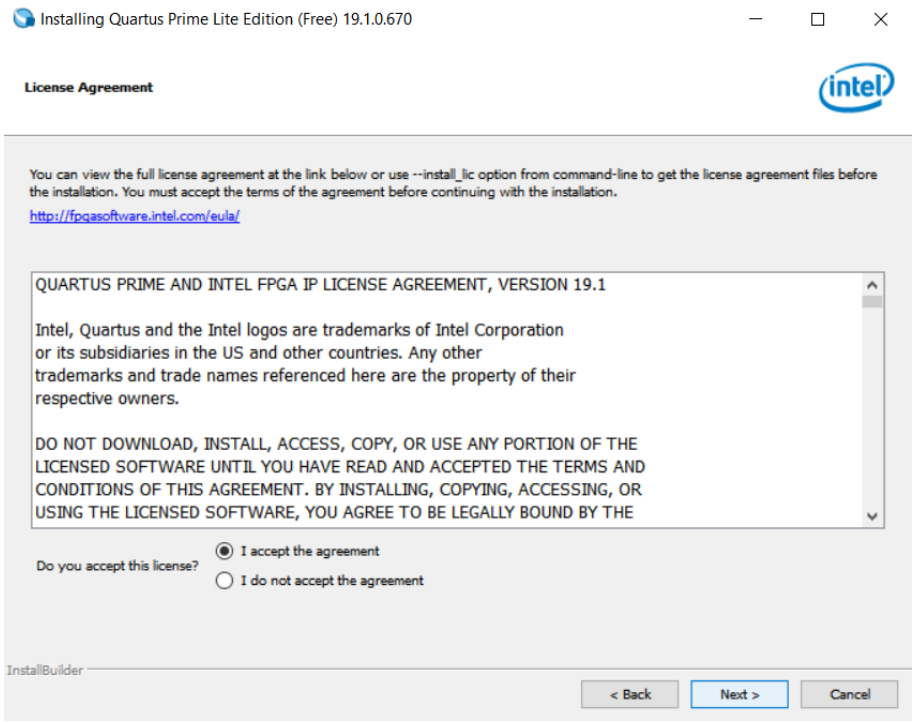
1. After Download necessary files extract them in a single folder and Click on the **QuartusLiteSetup** and **Allow** the application to install it on your device



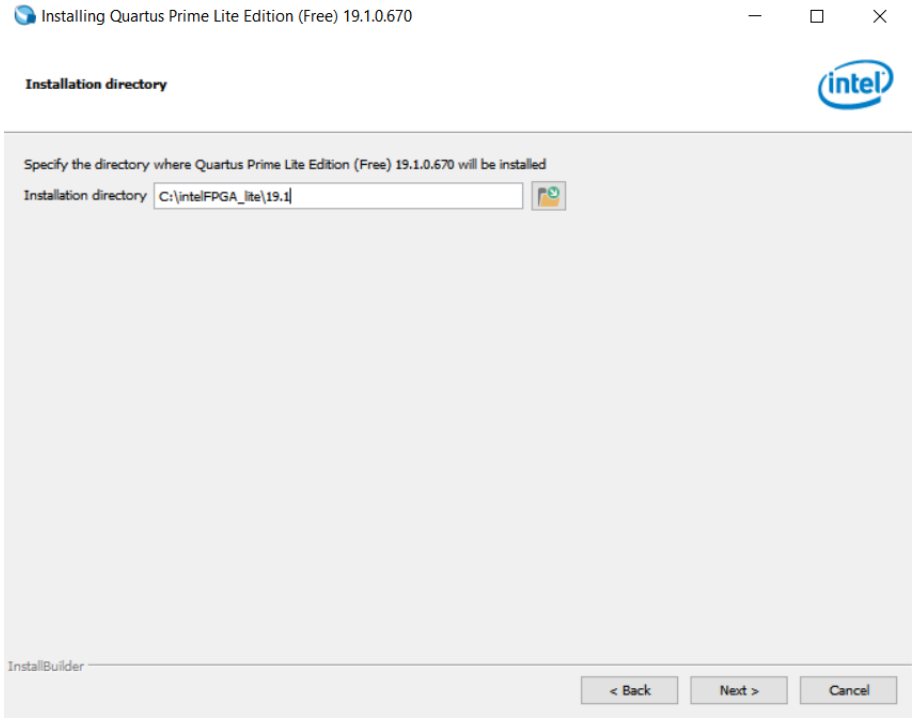
2. Click on **Next** to start with the installation.



3. Click on **I accept the agreement** and proceed

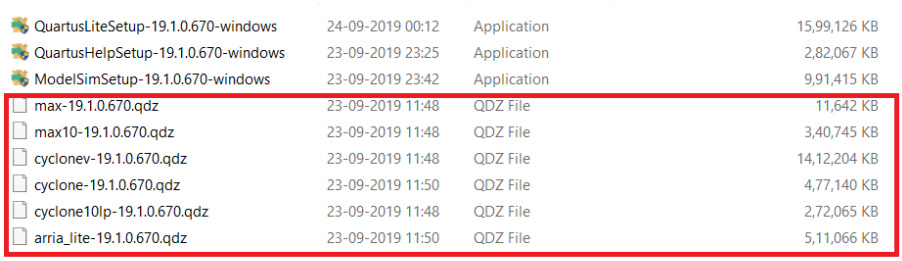


4. Enter the path where you need the software to be installed



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

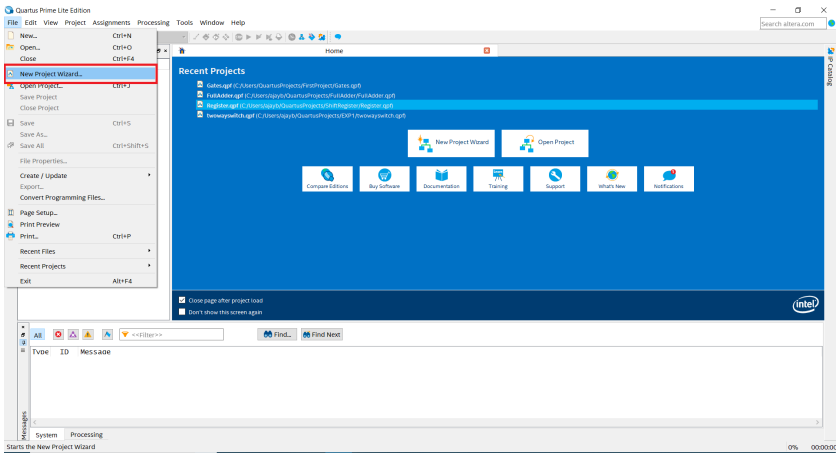


3 Getting started with Quartus prime

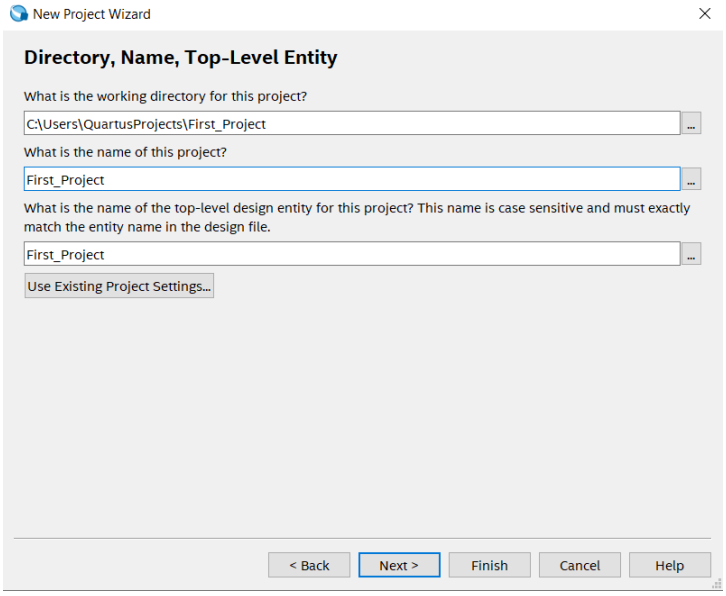
3.1 Creating a New project

In this section, we will learn how to create a new project using the Quartus software. As part of this, we will create multiple files for our designs, for testing our designs, and for downloading our design to the DE0-Nano board etc. To keep things simple, we will implement a simple logical AND gate design and the create Test Bench to simulate our design using ModelSim. To create a new project in Quartus Lite follow the below steps

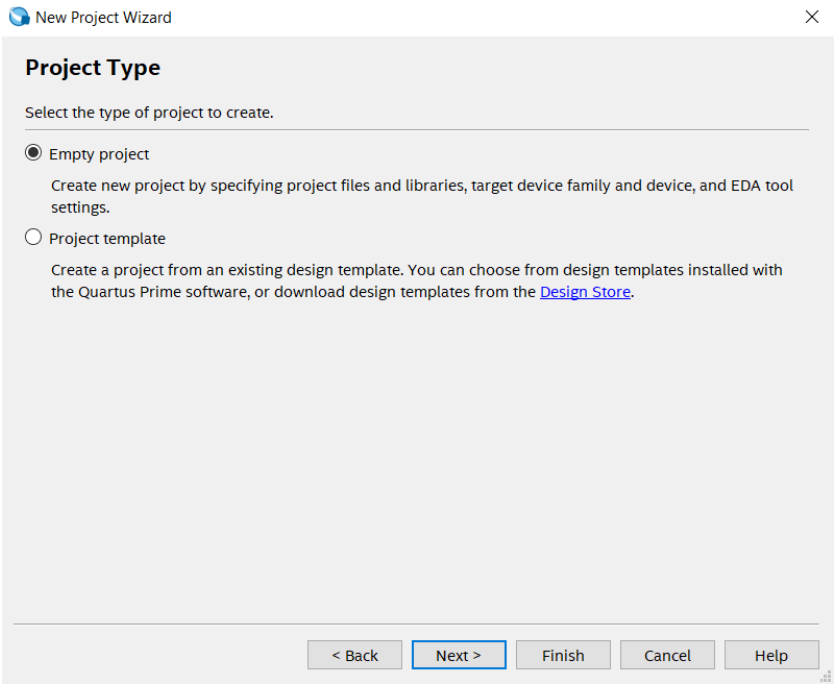
1. Click **File** → **New Project Wizard** to quickly setup and open a new project. Alternatively, **New Project Wizard** can be opened from the **Home Tab** that is seen when Quartus is opened.



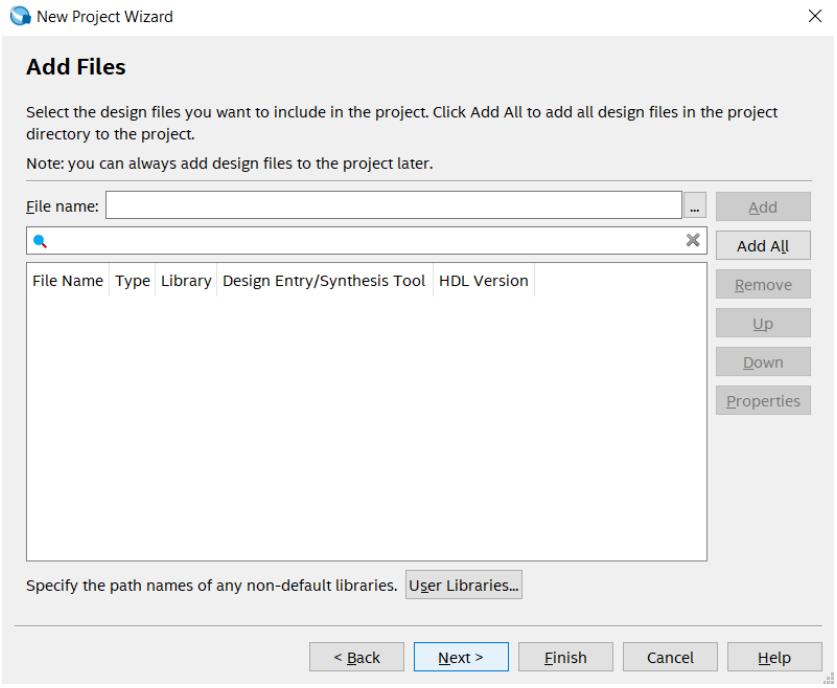
2. Click **Next** on the Dialog Box and select the **Directory** in which the project is to be saved. Give the **Project Name** and Click on **Next**.



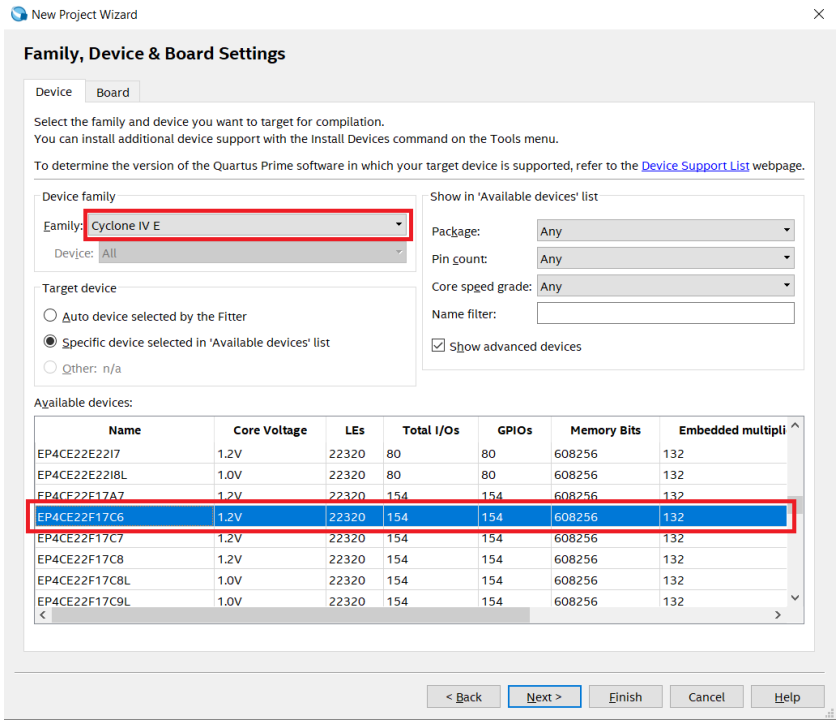
3. Choose a Project Template, Click on **Empty Project**. Selecting this option will allow us to manual specify the additional file and libraries, device family and EDA(Electronic Design Automation) tool setting.



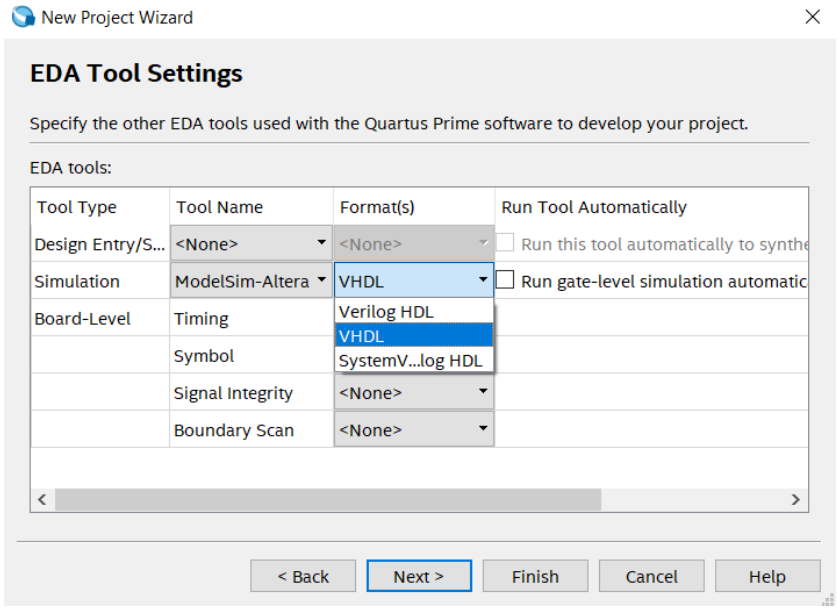
4. Add all the necessary files and Click on **Next**. For this project we don't require any additional files, so now we can continue without adding any file in our project.



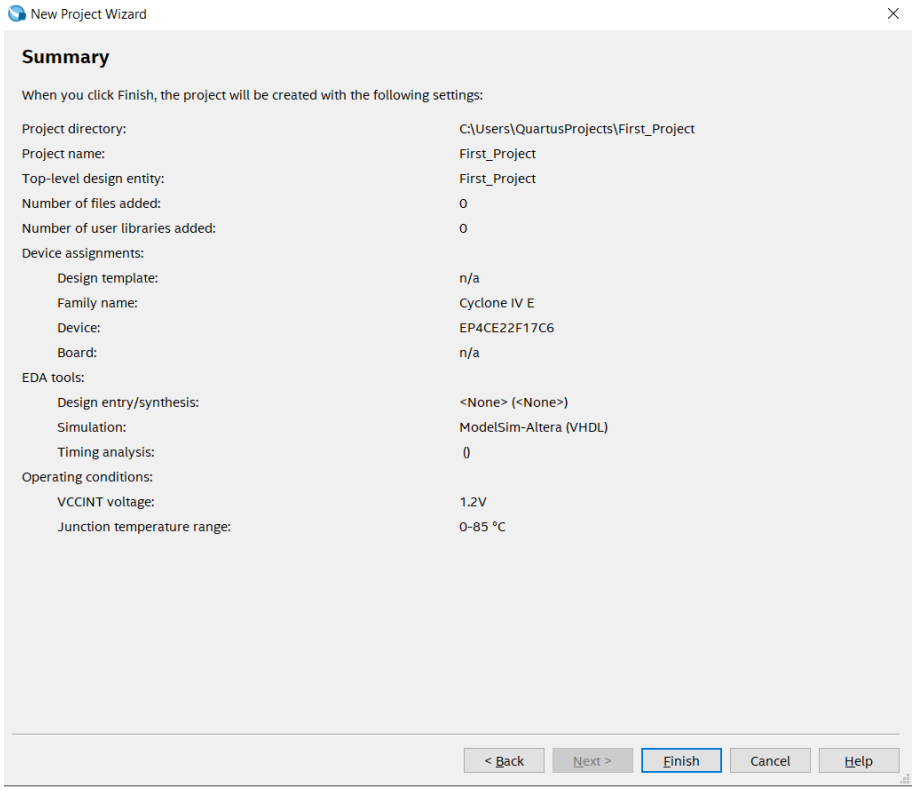
5. Choose the FPGA Device that is being used. Click on **Family** and choose the **Cyclone IV E**. Select **EP4CE22F17C6** from the **Available Device**. We can also search the device using the options provided in right side of the window.



6. Select the tools used in the project. For **Simulation**, choose **ModelSim-Alterra**. Also choose a suitable format(Verilog HDL, VHDL) as per your requirement, We will be providing both Verilog HDL and VHDL code in the tutorial, **you must select any one language throughout this tutorial.**



7. Review the information, you can change the entries by Clicking on **Back** as we cannot change these setting later in our project and make sure all the entries are correct. Click on **Finish** to successfully create a New Project



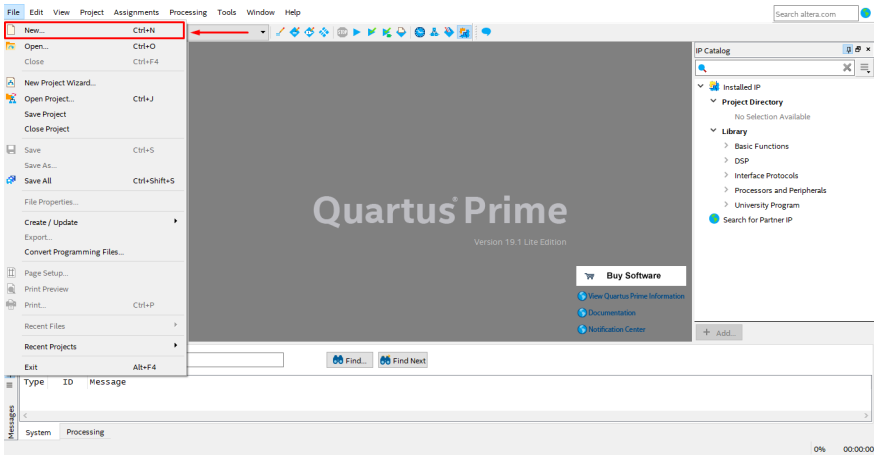
3.2 Creating new files in the project

In the previous steps we created a directory, and selected device families and simulation tool to set up a Quartus project, which told the tool about the board we are using. We now need to add some actual circuitry to the project.
We will create a simple design of a Logical And gate this design will have two data inputs “A” and “B”, and a single output “C”.

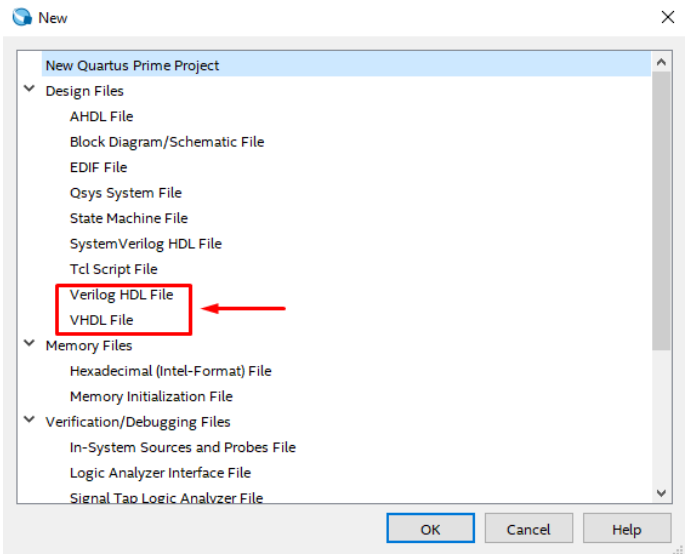
Follow the below steps to add necessary file, in our project.

3.2.1 Verilog HDL/VHDL File

1. Click on **File** → **New**



2. Select any one file type from drop down menu either **Verilog HDL or VHDL file** as per your requirements. You can follow the guidelines using any one language throughout this tutorial.



3. Write down the below code in the newly created file. Both Verilog HDL or VHDL code are provided. Also note that VHDL is not case sensitive but Verilog HDL is case sensitive.

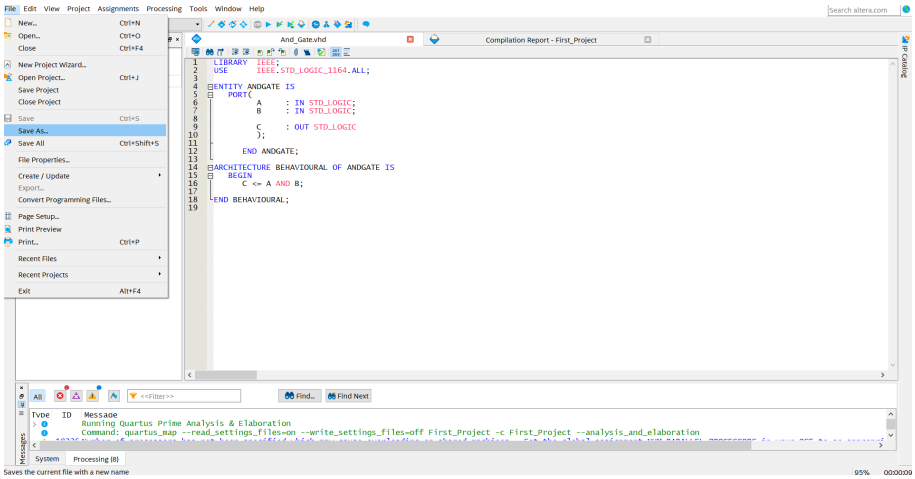
Below is the VHDL Code for AND gate

```
------------------------------------------------------------------
-- File:         And_Gate.vhd
-- Description: This is an implementation of a simple logical
--               And Gate using a dataflow architecture.
------------------------------------------------------------------
-- Libraries
LIBRARY          IEEE;
USE              IEEE.STD_LOGIC_1164.ALL;


-- Entity Declaration
-- This is where all the Inputs & Outputs are specified for
-- our AND gate
ENTITY AND_GATE IS
        PORT(
                A        : IN STD_LOGIC;
                B        : IN STD_LOGIC;

                C        : OUT STD_LOGIC
                        );

                END AND_GATE;

-- Architecture Body
-- This is where we describe our logical expression
-- for AND gate
ARCHITECTURE BEHAVIOURAL OF AND_GATE IS
        BEGIN
                C <= A AND B;

END BEHAVIOURAL;
```

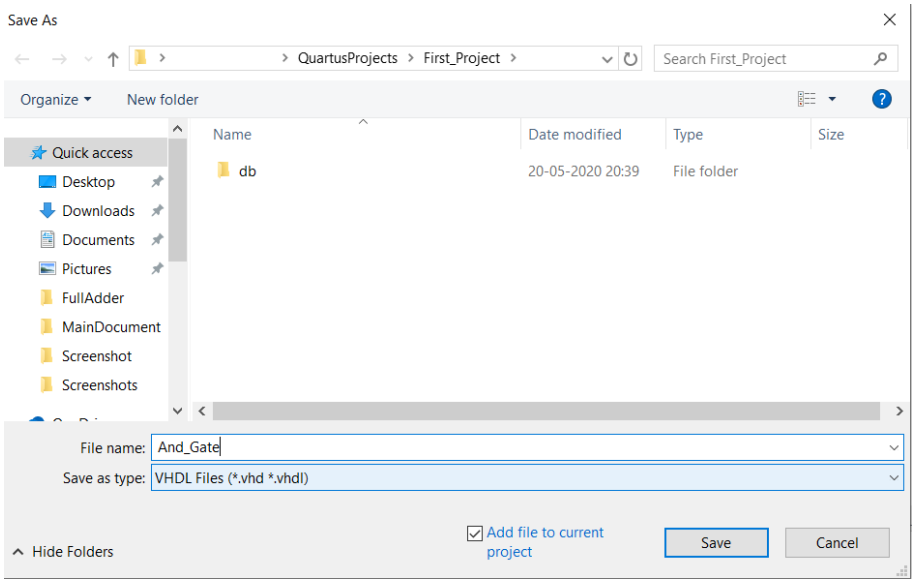
Below is the Verilog Code for AND gate

```
// Verilog code for AND gate
module And_Gate
        (
        input a,b,           // defining inputs A and B of AND gate
        output out           // defining output of AND gate
        );

assign out = a&b;        // Logic implementation
endmodule
```

4. After completing the code click on **File** → **Save As**

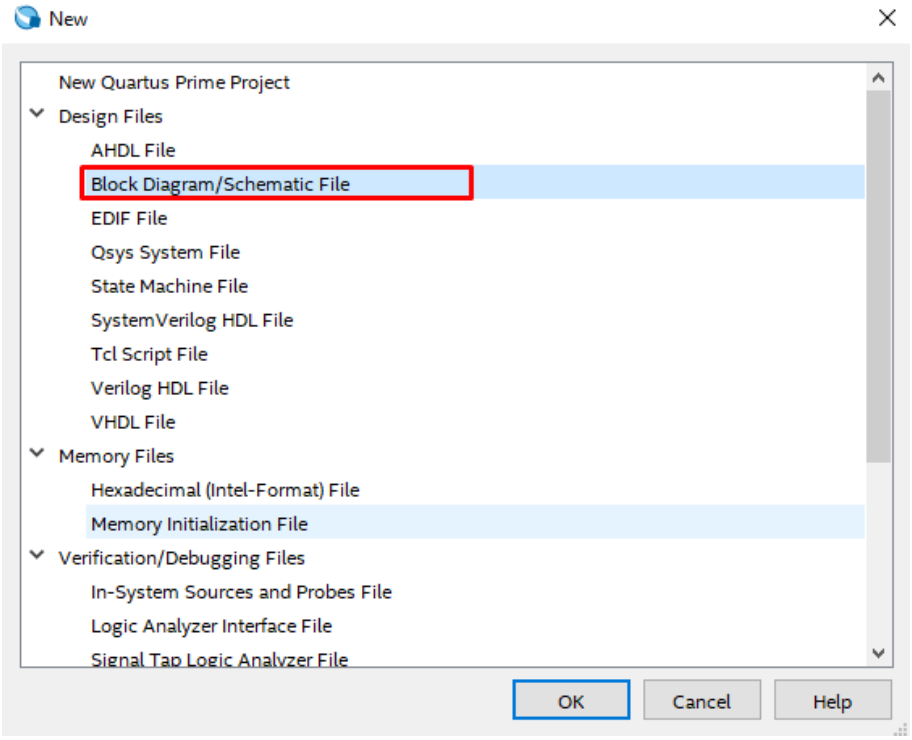


5. Enter the Name of file (it should be same as module/entity name), Enter correct file extension, for VHDL it is **.vhd** and for Verilog HDL it is **.v** and then Click on **Save**



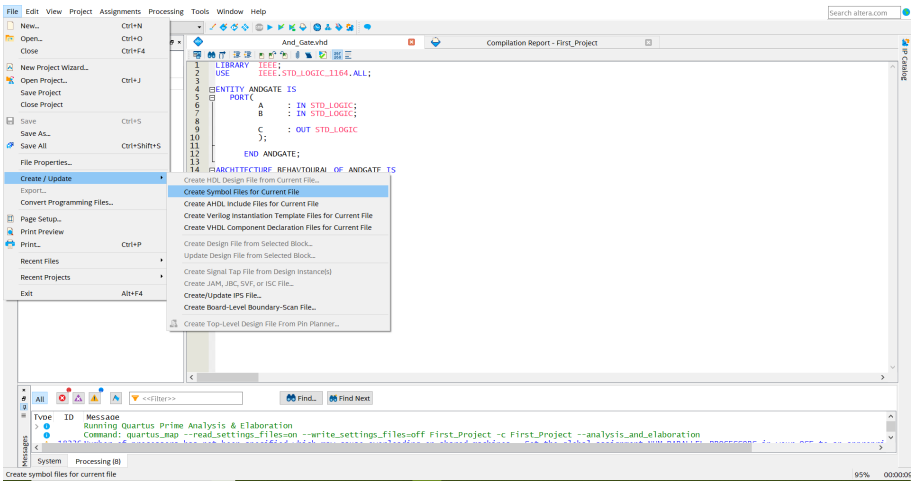
3.2.2 Block Diagram File

1. Click on **File→New** and Select **Block Diagram/Schematic File** from the drop down menu

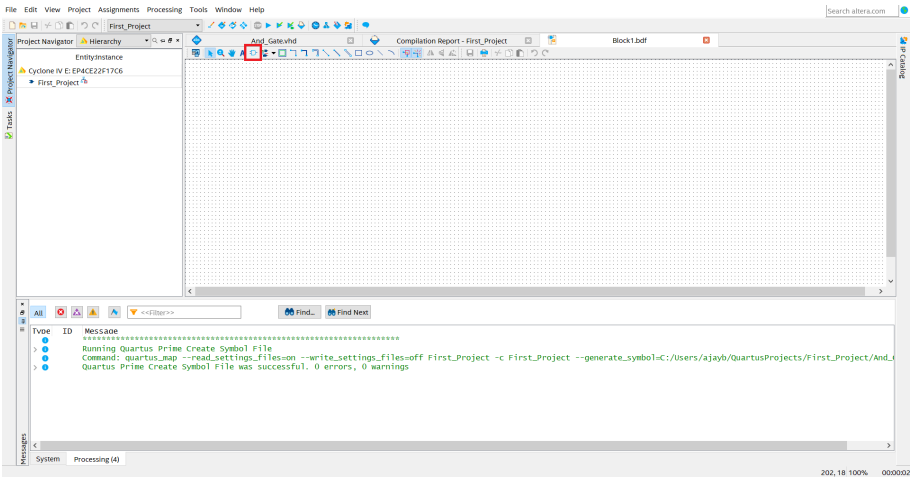


2. Creating Block Diagram. First we need to create Symbol File of our VHDL/Verilog HDL code so that we can add it in the Block Diagram File.

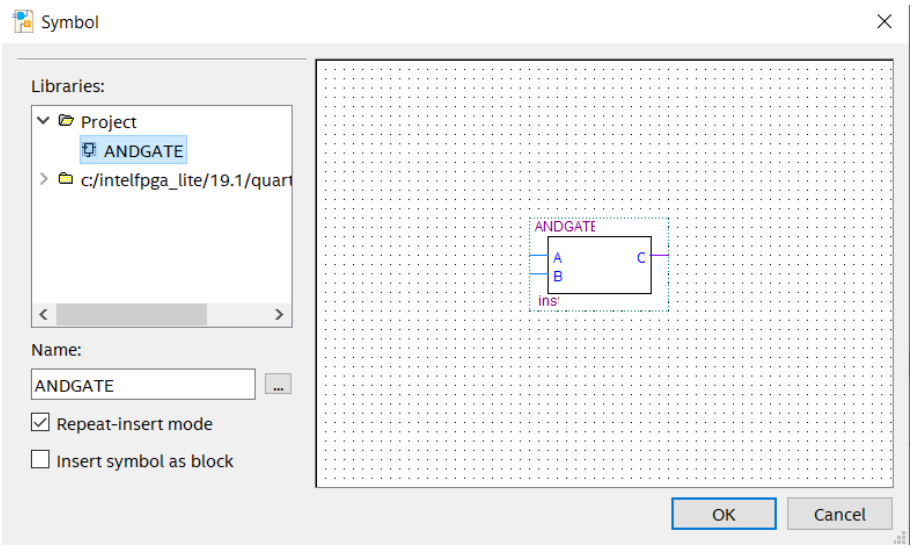
   File→Create/Update→Create Symbol Files for Current File



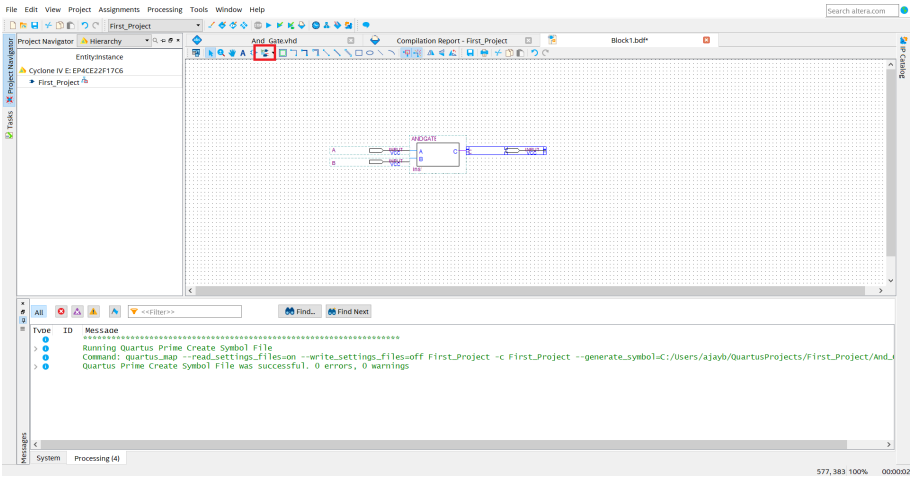
3. Now we will import our Symbol File in your block diagram. Click on the **Symbol Tool** or double click anywhere in the file.



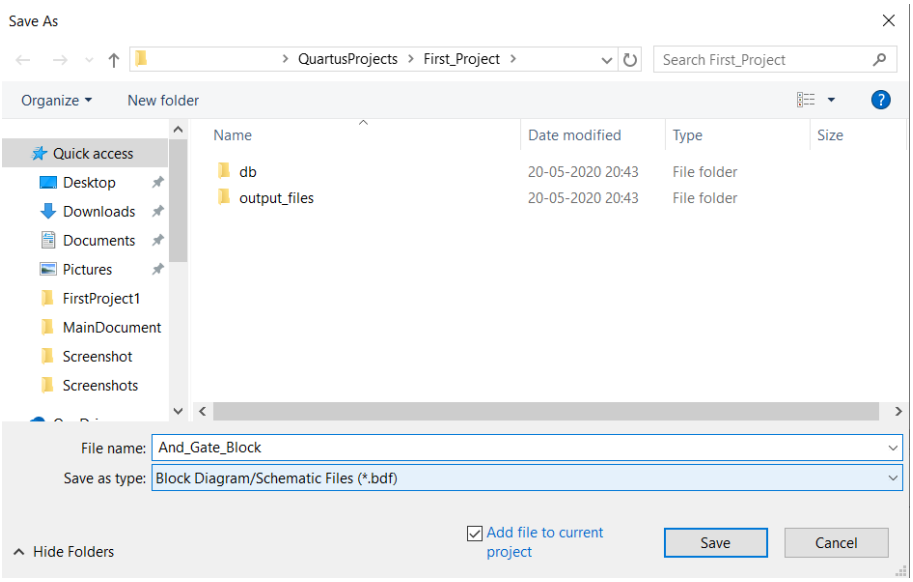
4. Now, Select your **Project** → **And_Gate** and Click on OK. You can place the Symbol anywhere in your layout. Note: You can place as many duplicates of the same symbol by clicking repeatedly, until you press esc.



5. You can use the **Pin Tool** for assigning I/O ports to Inputs A and B, Output C. You can change the I/O ports name by right clicking on it and the changing the **Properties** field.

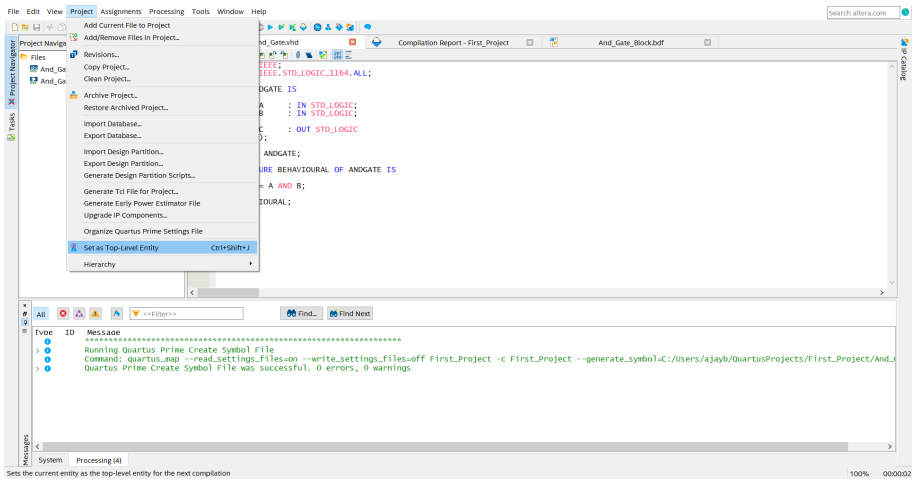


6. To save the file, click on **File→Save as**. Enter the Name of your file as shown in the figure and Click on **Save**.

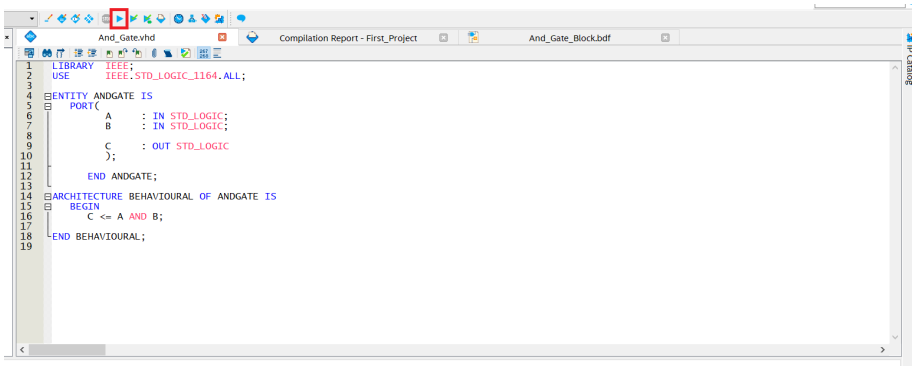


3.3 Compiling the Project Files

1. To compile the file, the file must be the top-level enity of the project. We can assign the current file as top-level entity by **Project** → **Set as Top-Level Entity** option in toolbar



2. Generate synthesis or final compilation results by running the following commands: Click **Start Compilation** shown in the figure below(red box) to generate final compilation results.



4 Pin Assignment and Loading the design on FPGA board

4.1 Pin Description

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".

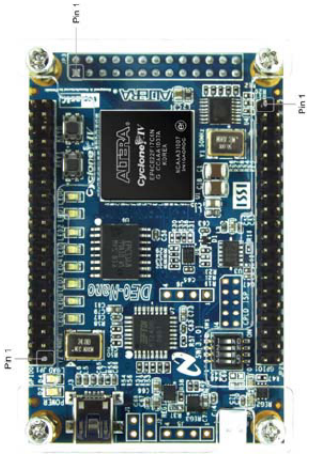
DE0-Nano package contains the following General user I/O pins:

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

Dedicated pins includes

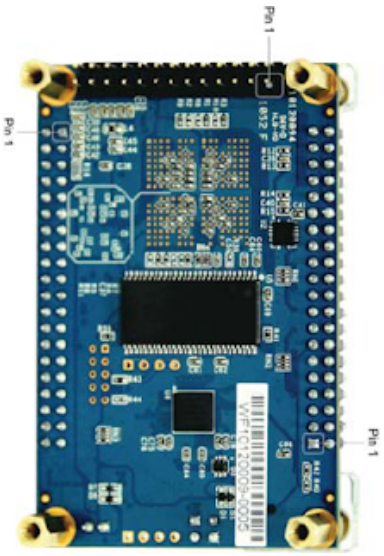
- Two 40-pin Headers (GPIOs) provide 72 I/O pins, 5V power pins, two 3.3V power pins and four ground pins.
- 2x13 header, A/D converter

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



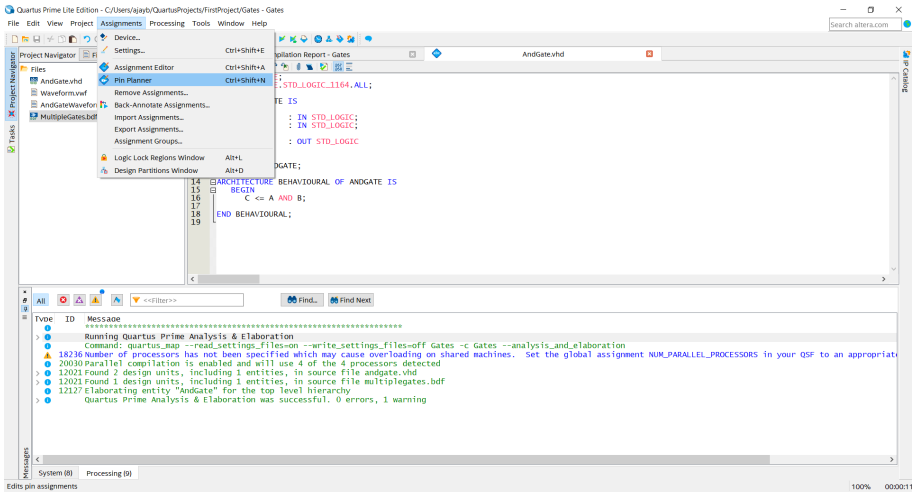
Pin Name for sliding Switches		
Signal Name	FPGA PinNo.	Description
DW[0]	PIN_M1	DIP Switch[0]
DW[1]	PIN_T8	DIP Switch[1]
DW[2]	PIN_B9	DIP Switch[2]
DW[3]	PIN_M15	DIP Switch[3] V
Pin Name for Push Button		
Signal Name	FPGA PinNo.	Description
KEY[0]	PIN_J15	Push-button[0]
KEY[0]	PIN_E1	Push-button[1]
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.

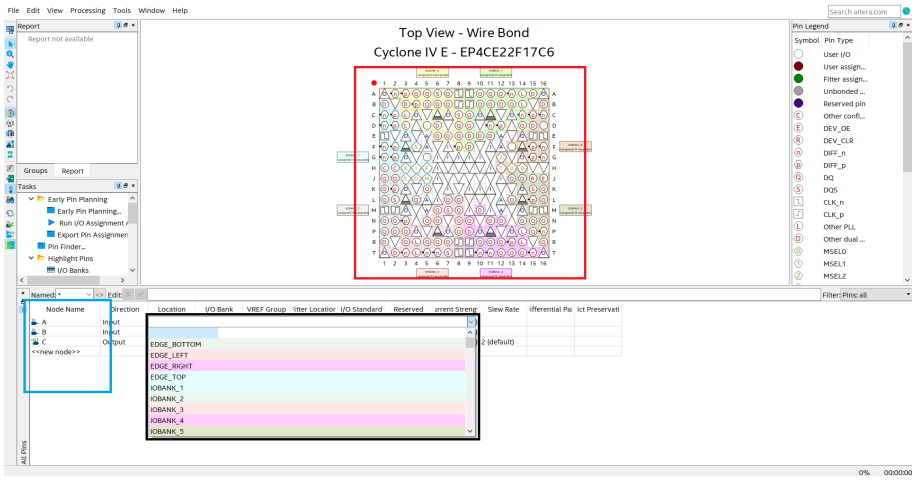
Further in this section we will map the I/O pins created in our design with the actual pins of the DE0-Nano. Before starting pin assignment make sure to do Analysis Elaboration check.

4.2 Pin Assignment

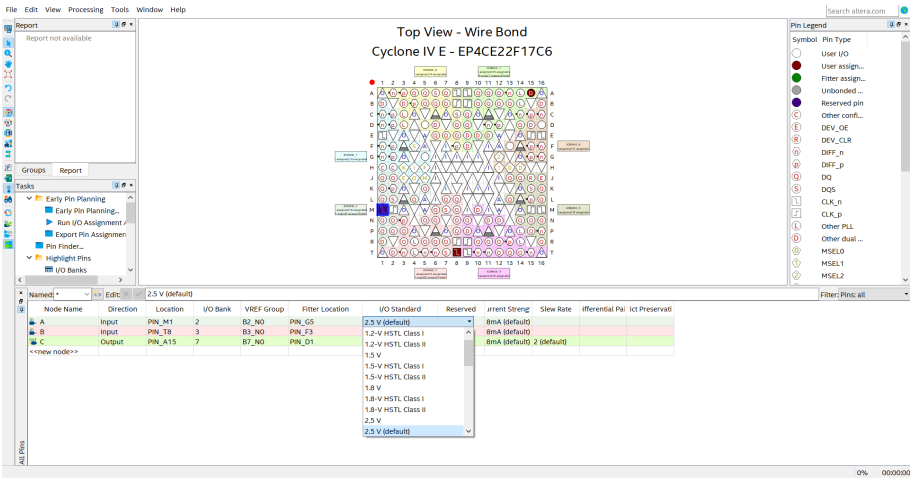
1. Click on **Assignments** → **select Pin Planner**, which opens the Pin Planner, the Pin Planner shows the I/O that we have created in our design.



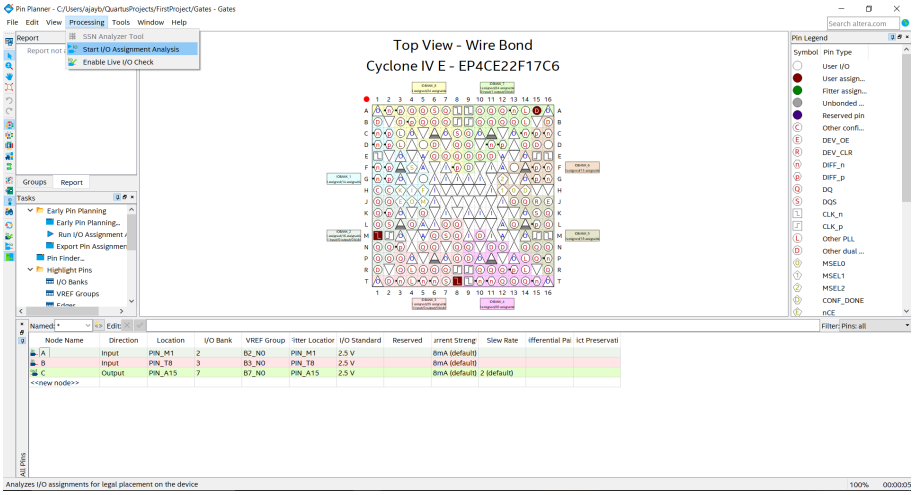
2. Now assign the pins by either dragging and dropping the nodes(Blue box) on the required pins(**Black** box), or click on the drop down menu(Red box) to search and select a pin. We will select LED[0] as the output of the AND gate and Slide switch 0 and 1 for input signal A and B.



3. Make sure to select the appropriate I/O standard by referring to the board manual or go with the default setting.



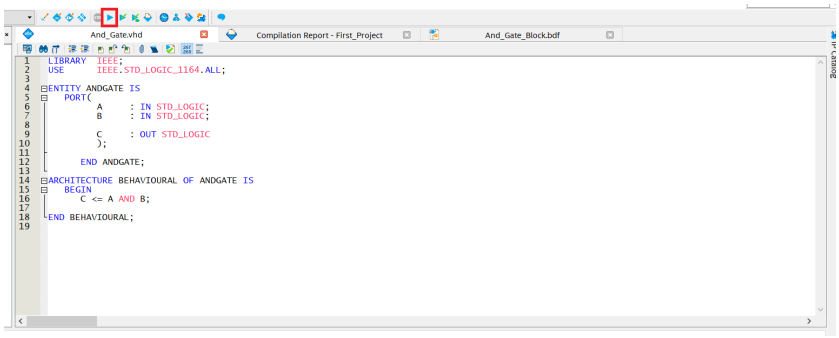
4. Once all the pins are assigned, Go to **Processing→Start I/O Assignment Analysis**. The analysis checks pin assignments and surrounding logic for illegal assignments and violations of board layout rules.



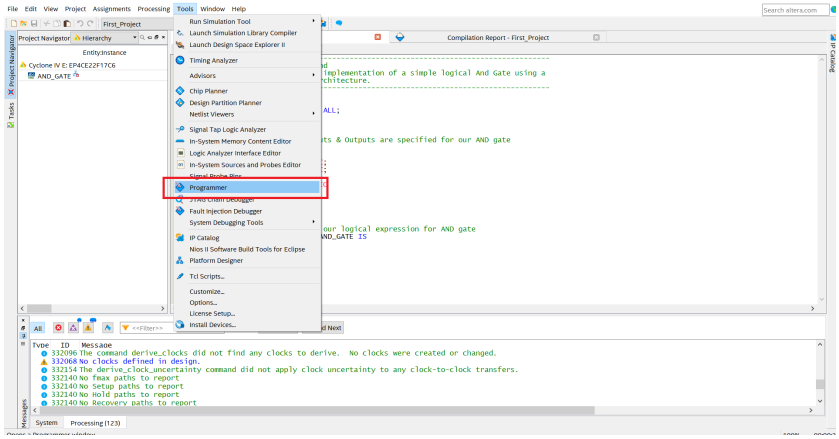
4.3 Downloading the code to DE0 Nano FPGA Board

Before starting, Make sure the board is powered ON and connected to the computer through an USB Cable

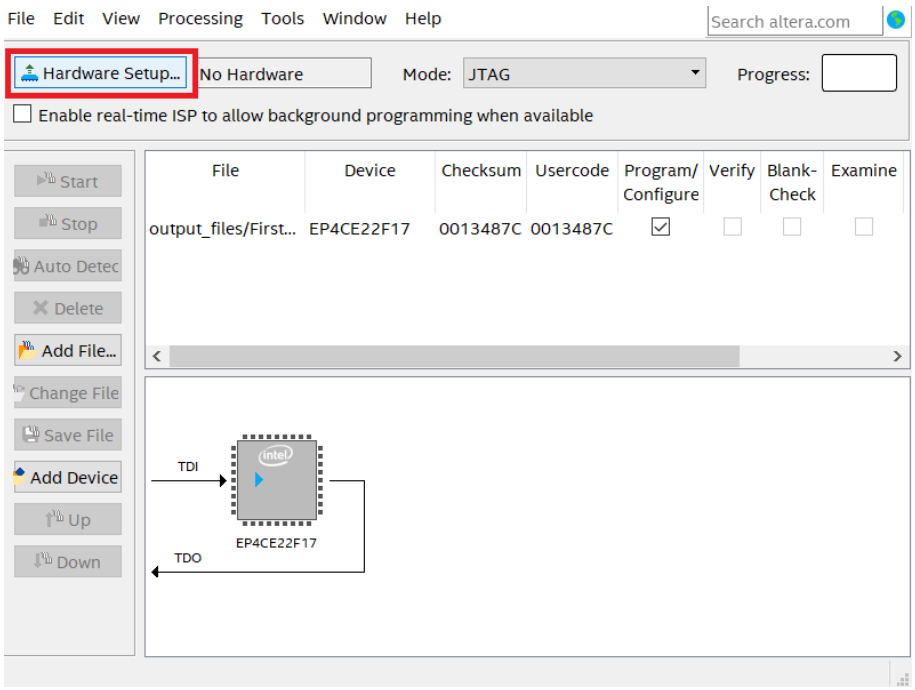
1. Compile the Project by Clicking on **Start Compilation** .This creates an **SRAM object file**(.sof file). This file is used to program the Device



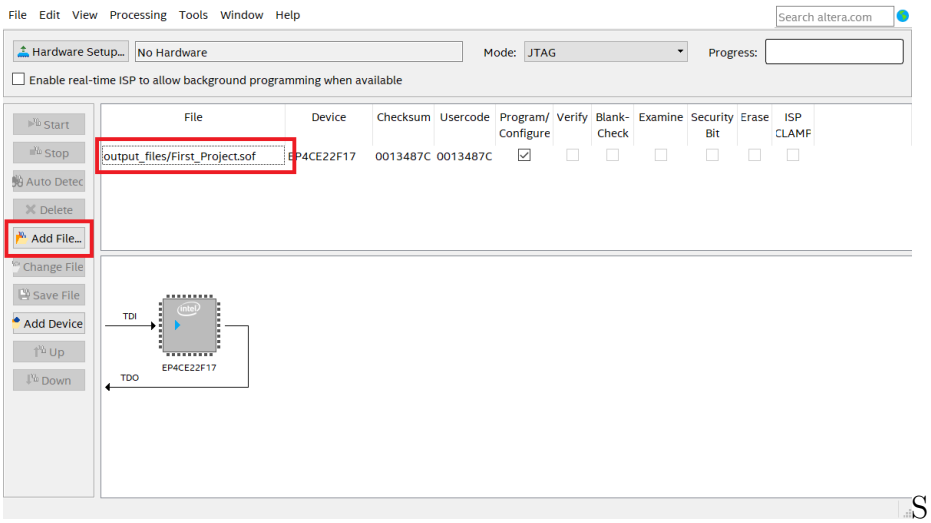
2. Open the programmer by going to **Tools→Programmer**



3. Click on **Hardware Setup**. The Device required must be listed under "Currently available hardware". If not, check if the device drivers are correctly installed. Choose the hardware from the dropdown menu

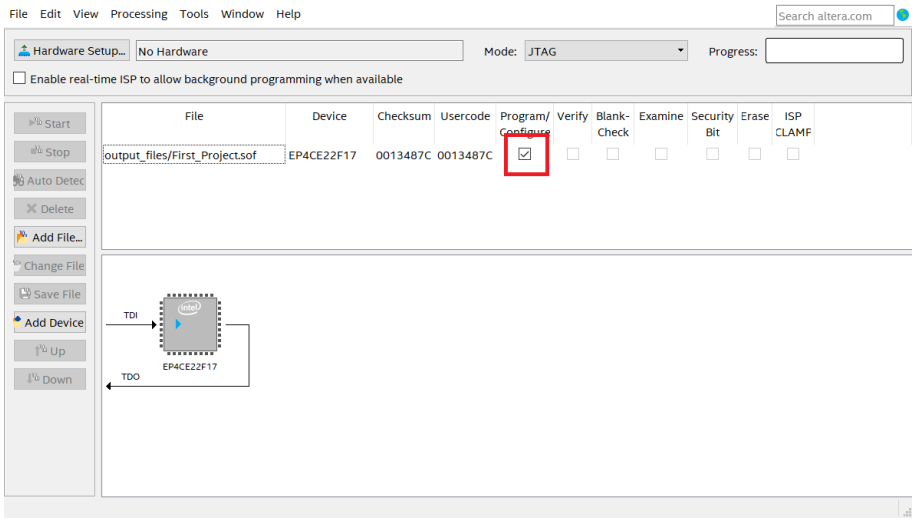


4. If the file is not listed, it can be manually added by clicking on **Add File**. The .sof file can be found the output directory inside the project directory

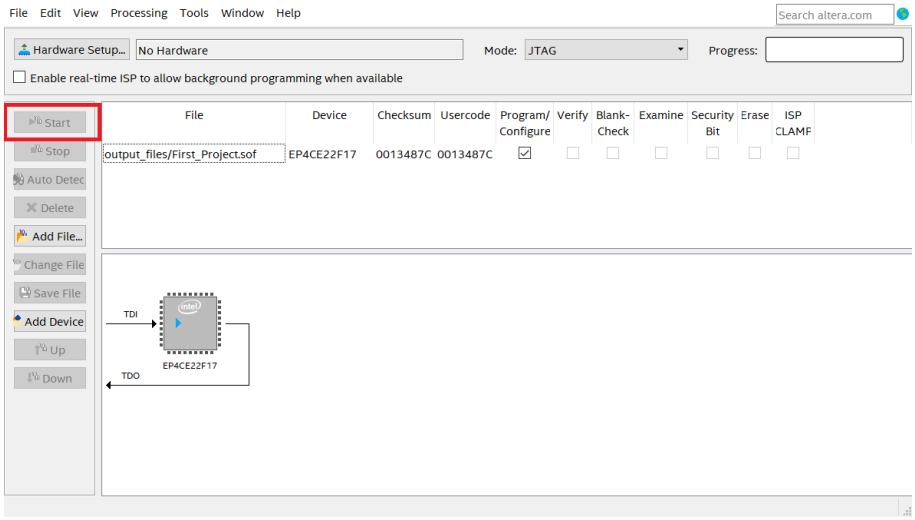


S

5. Make sure the Program/Configure checkbox is ticked



6. Click on **Start** to start the programming process. **Start** button will be enabled when the 'DEO NANO' board is connected to USB port of your device.

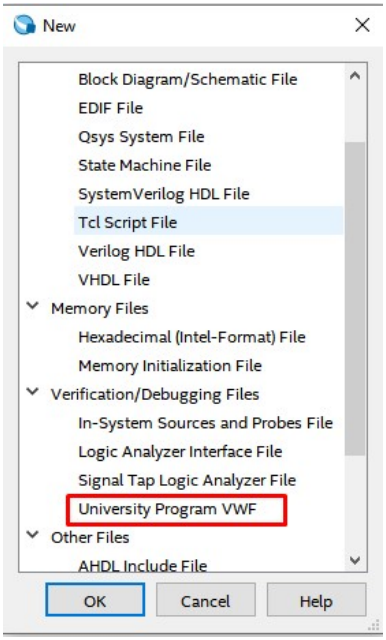


5 Verification by Simulation(using ModelSim)

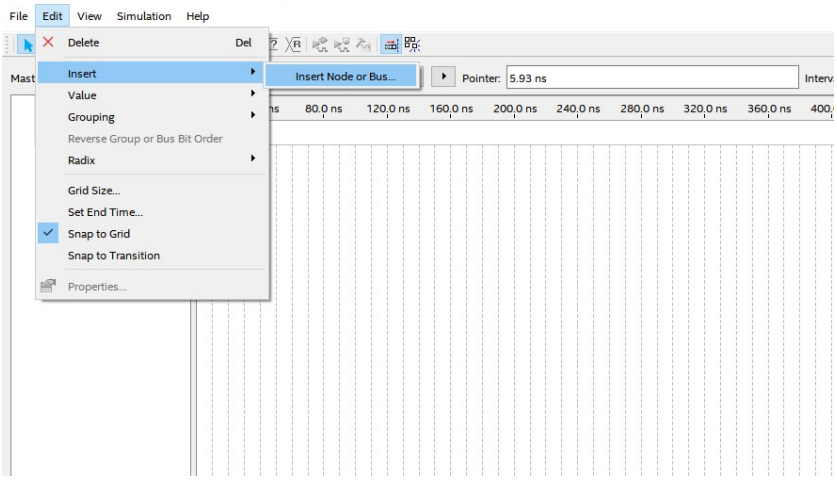
You can perform functional and timing simulation of your design by using EDA simulation tools. We can simulate our design by two methods.These two methods are explained breifly below. In the first method we set the clock for each input "A" and "B" manually by setting clock frequency and run the simulation to get the output "C" . In the second method we will write a test bench file using Verilog HDL or VHDL to simulate our design in ModelSim.

5.1 Without TestBench

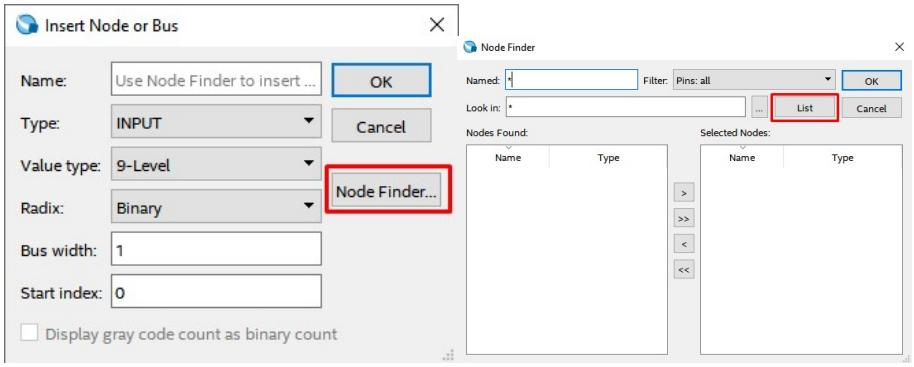
1. Click on **File→New→University Program VWF**



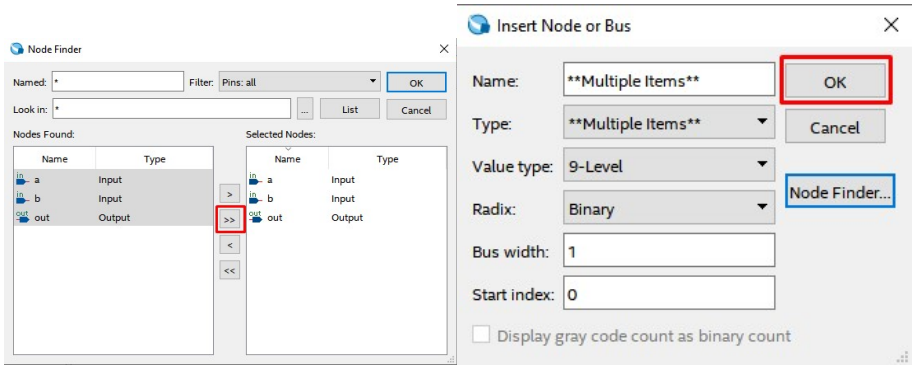
2. In the New window, Go to **Edit→Insert→Node or Bus**



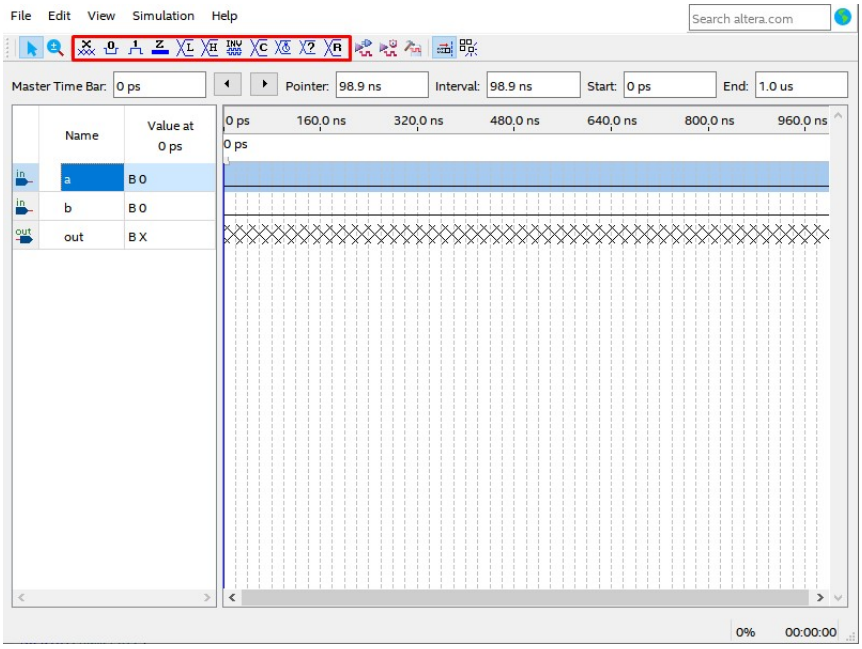
3. Click on **Node Finder**, in the New Window, Click on **List**



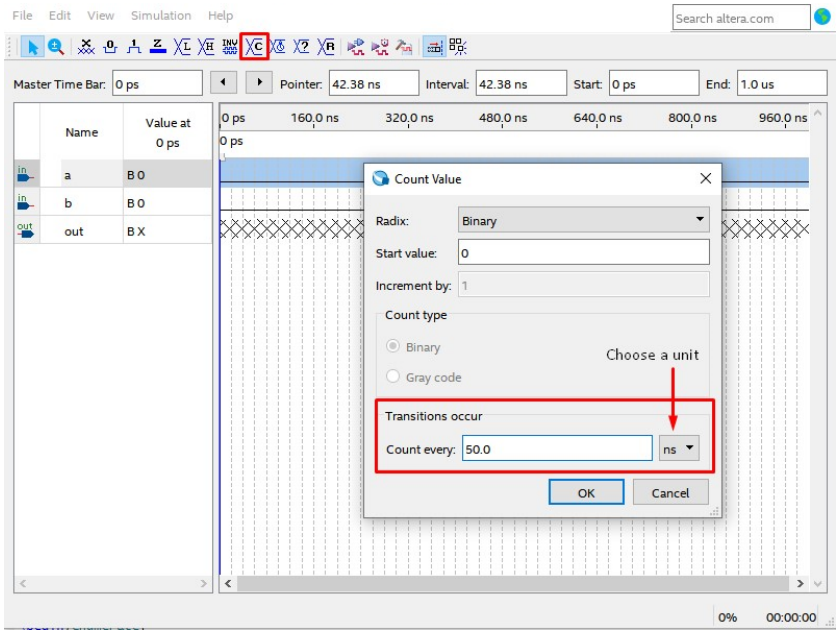
4. Click on the highlighted button to add all the Nodes. Now Click on **OK** and then **OK** again



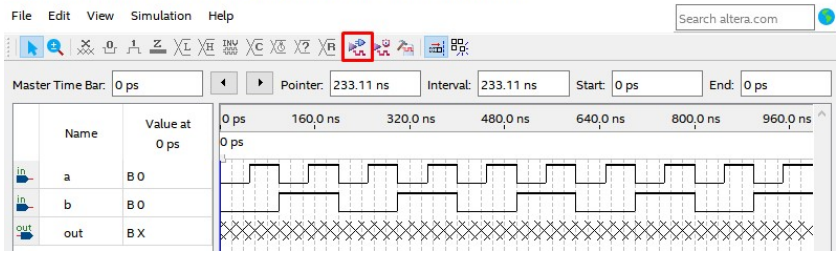
5. Choose input A and specify its value by choosing any of the options that are highlighted



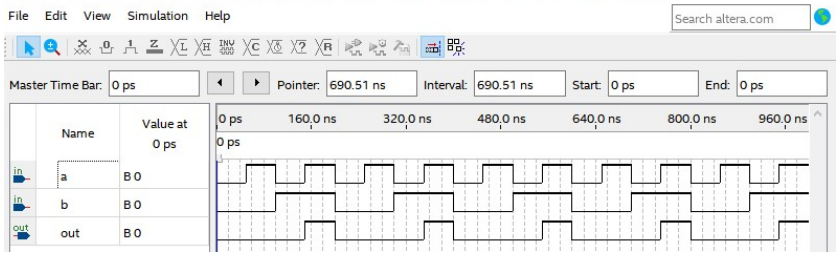
6. For example, to give the Clock pulse to 'A' input, we first Select the a node, then Click on the **Count Value** button highlighted at the top in the below figure, and then specify the Clock Pulse of frequency 50ns



7. Similarly, specify the input patterns for inputs B with Clock Pulse of frequency 100ns 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.
This is the waveform before the simulation starts



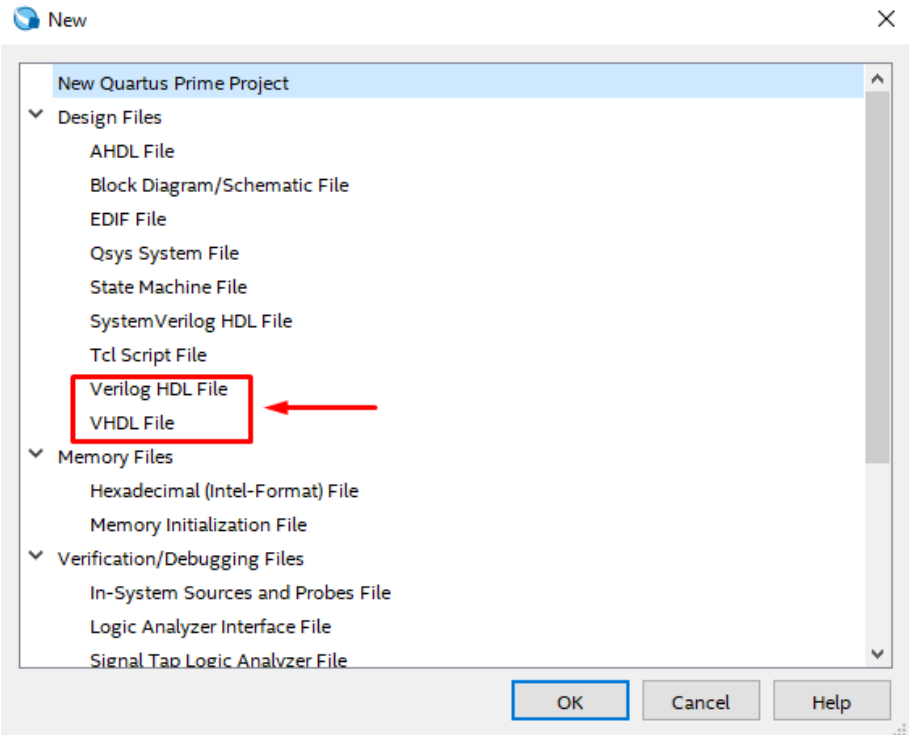
This is the waveform after the simulation ends



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.

1. Create a new Verilog HDL/VHDL file(select only one file), Add the testbench code and save it in the same project directory. Both Verilog HDL and VHDL test Bench code is provided below.



Below is the VHDL Test Bench Code for AND gate

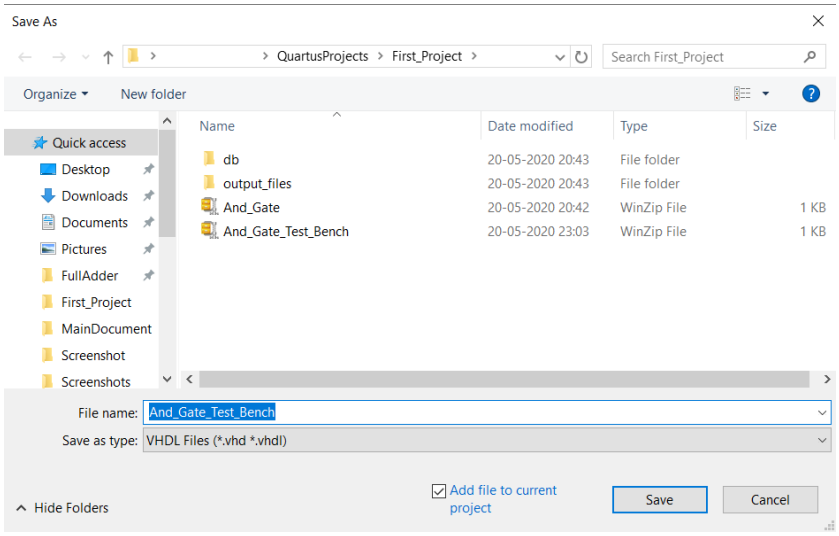
```
LIBRARY          IEEE;
USE              IEEE.STD_LOGIC_1164.ALL;

--defining the entity
ENTITY And_Gate_Test_Bench IS
END And_Gate_Test_Bench;
--defining the architecture
ARCHITECTURE BEHAVIOURAL OF And_Gate_Test_Bench IS
    COMPONENT AND_GATE
                PORT(
                A        : IN STD_LOGIC;          --Switch 1
                B        : IN STD_LOGIC;          --Switch 2
                C        : OUT STD_LOGIC          --LED Output
                );
    END COMPONENT;
```

```
   SIGNAL A : STD_LOGIC := '0'; --Stimulus signal for switch 1
   SIGNAL B : STD_LOGIC := '0'; --Stimulus signal for switch 2
   SIGNAL C : STD_LOGIC;            --Output signal
        BEGIN
--defining unit under test i.e And_gate
        uut: AND_GATE PORT MAP (
                                A => A,
                                B => B,
                                C => C
                                );
                           --begining stimulation process
                           stim_proc: process
                            BEGIN
                                    A <= '0';
                                    B <= '0';
                                    WAIT FOR 50 ns;
                                    A <= '0';
                                    B <= '1';
                                    WAIT FOR 50 ns;
                                    A <= '1';
                                    B <= '0';
                                    WAIT FOR 50 ns;
                                    A <= '1';
                                    B <= '1';
                                    WAIT FOR 50 ns;
                                    WAIT;
                             END PROCESS;
END BEHAVIOURAL;
```

Below is the Verilog HDL Test Bench Code for AND gate

```
// Verilog Test Bench code for AND gate
module And_Gate_Test_Bench;
reg a;reg b;
wire out;
//Defining  unit  under  test i.e  And_gate
And_Gate uut(.a(a),.b(b),.out(out));
// Assigning all possible states for input A and b
initial begin
        a = 0;b = 0;#100;
        a = 0;b = 1;#100;
        a = 1;b = 0;#100;
        a = 1;b = 1;#100;
end
endmodule
```

Save the file with correct file extension, for VHDL it is **.vhd** and for Verilog HDL it is **.v** and then Click on **Save**

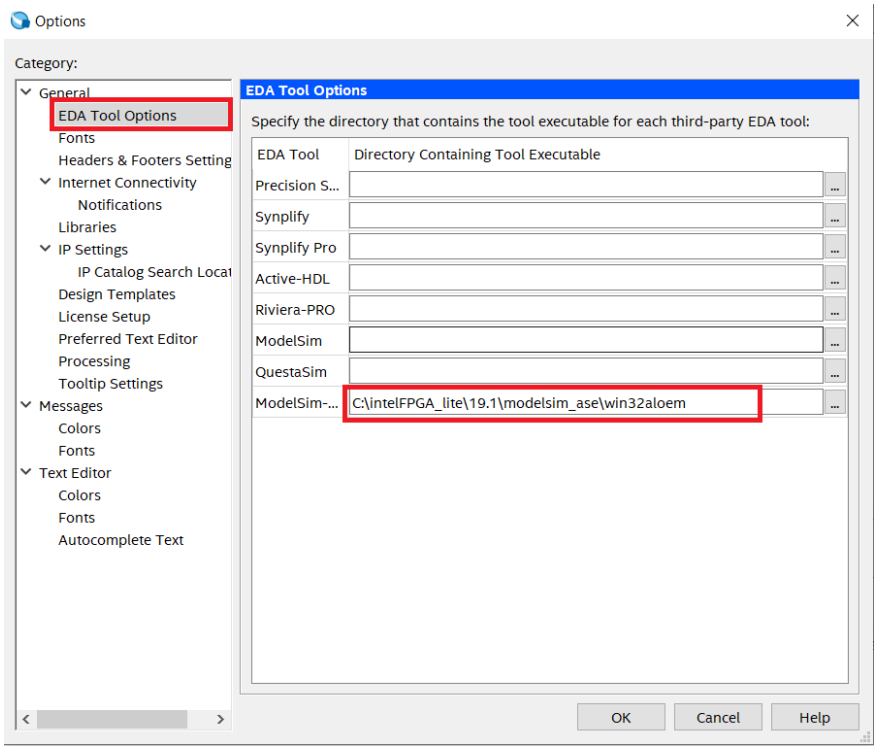


2. **Specify the path to Modelsim Altera**

   (a) Go to the menu **Tools** → **Options**

   (b) In the **General** category, Select **EDA Tool Options**.

   (c) A dialogue box appears, where you can specify the location path of the Modelsim-Altera executable file.And Click **OK**.



3. **NativeLink Settings to configure Modelsim-Altera**

   (a) Go to the menu **Assignments → Settings.**

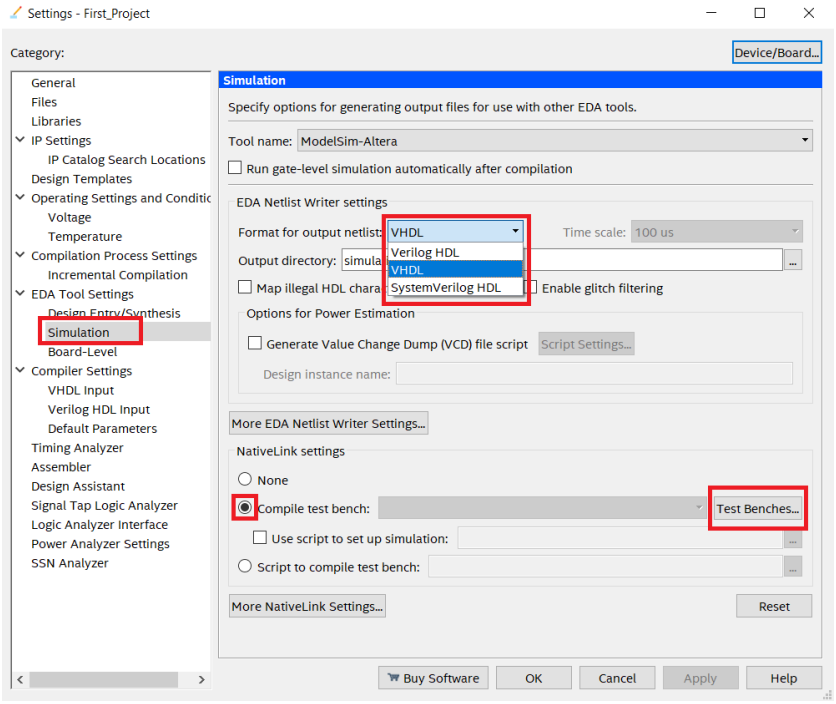
   (b) Under **EDA Tool Settings → Simulation**. The dialogue box for simulation appears.

   (c) For Tool Name, Choose **Modelsim-Altera**.

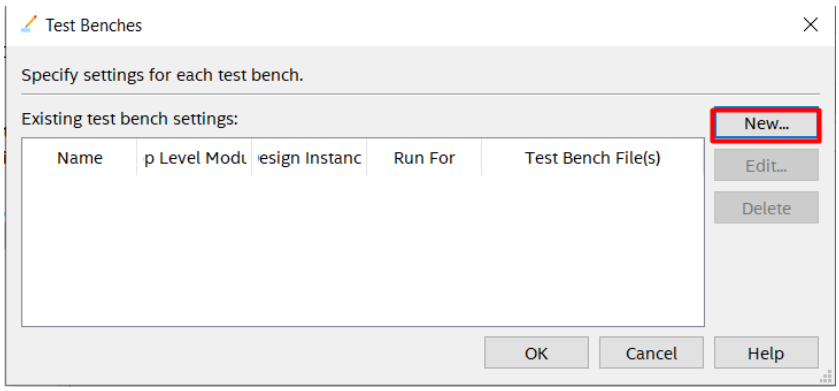
   (d) Select **VHDL or Verilog** as the Format for Output Netlist.

   (e) Select **simulation/modelsim** as the Output Directory.

   (f) Under NativeLink Settings, Choose **Compile Test Bench→ Test Benches**.All these changes are indicated in the below figure.



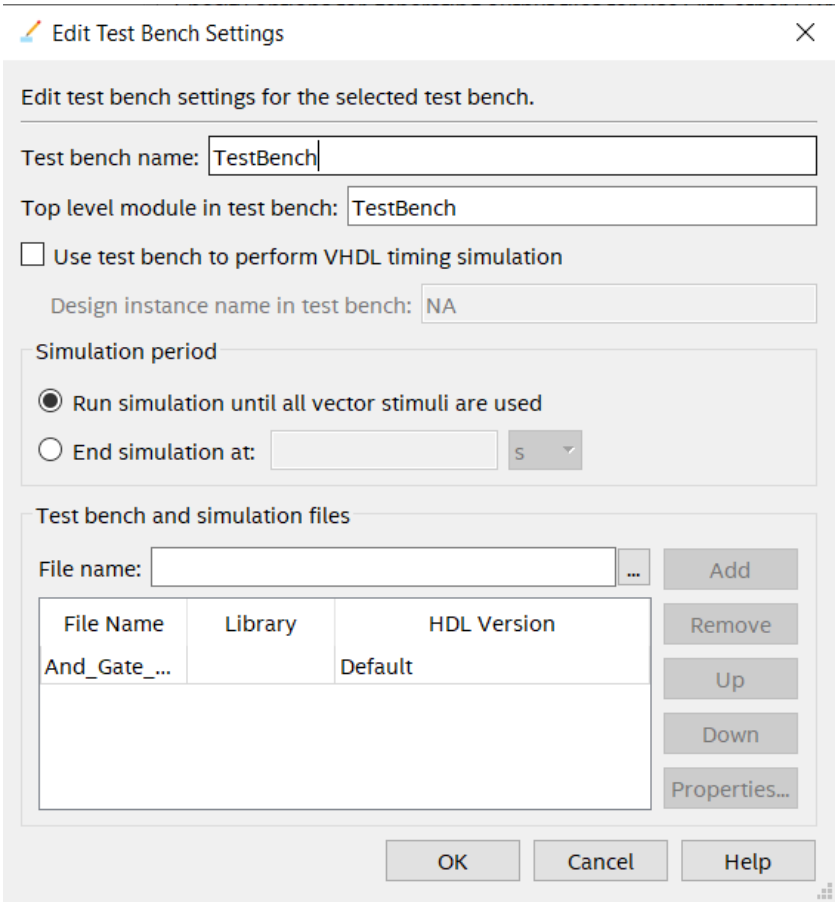
   (g) A new window appears Select **New**.



(h) Another window appears with the name **New Test Bench Settings**.

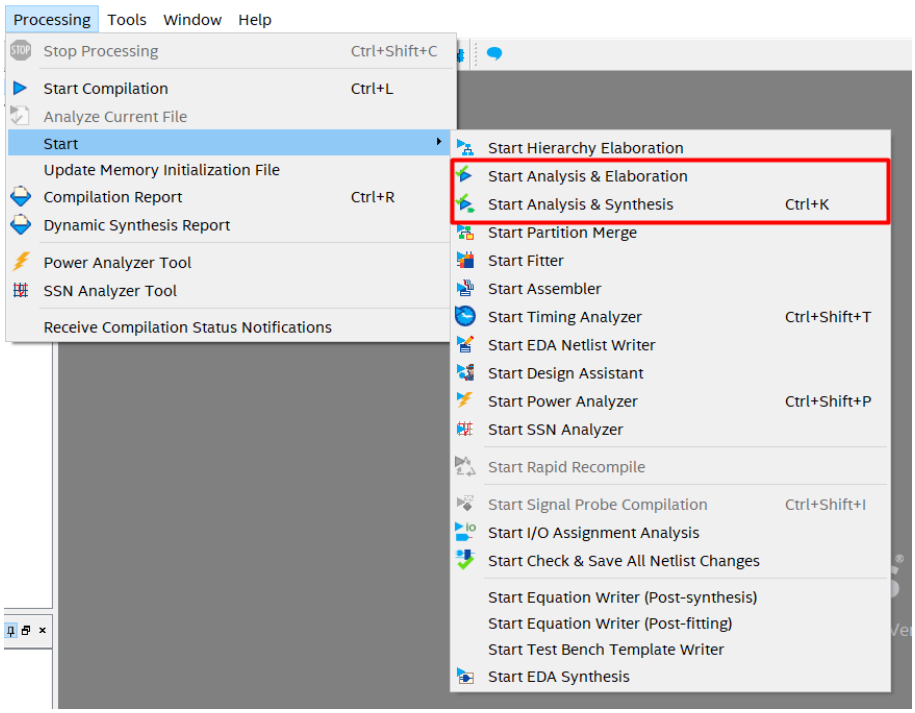
(i) Enter the Test Bench Name and Top Level Module in test bench as shown below.

(j) Add the Test Bench File that we created above in the previous section and then click **OK**.

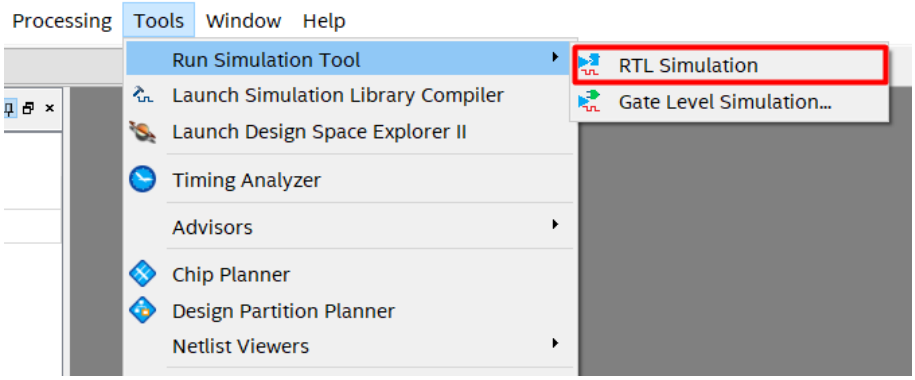


5.2.1 Functional Simulation using NativeLink Feature:

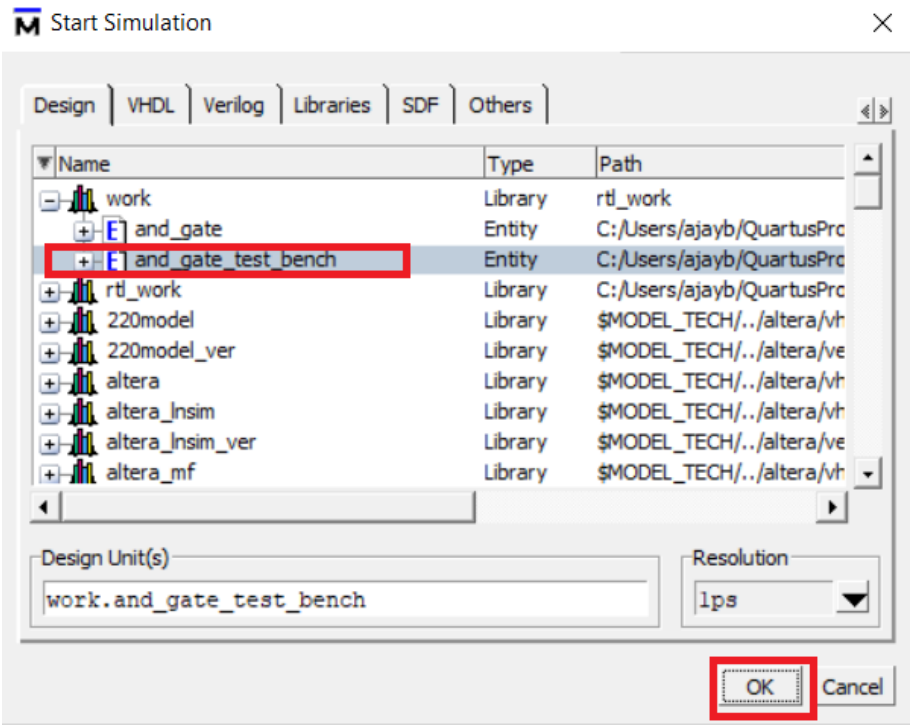
1. Goto menu **Processing →Start→Start Analysis & Elaboration.**
   After this Click on **Processing →Start→Start Analysis & Synthesis** on the same drop box .
   These step checks the error and collects all file name information and builds the design hierarchy for simulation.



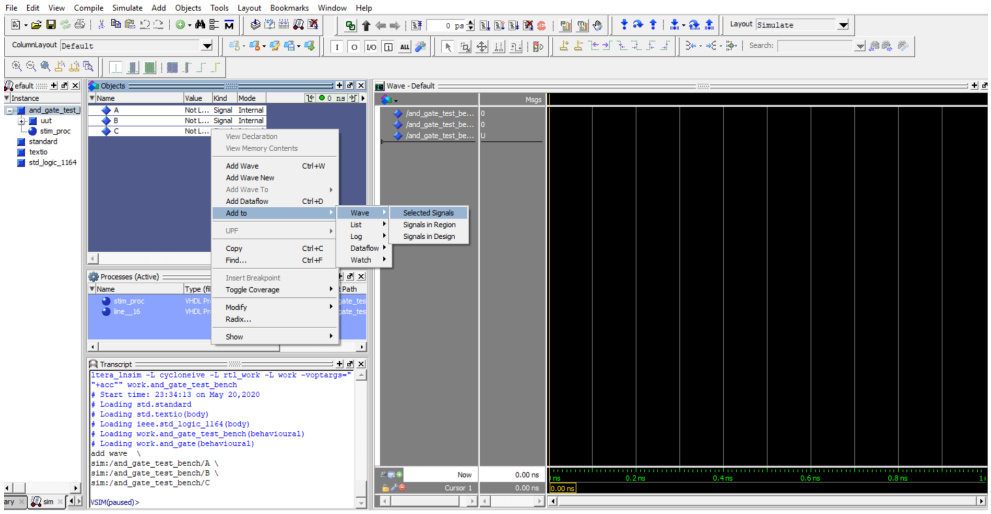
2. Goto menu **Tools→Run Simulation Tool→RTL Simulation** it automatically run the EDA simulator(ModelSim-Altera) and to compile all necessary design files.



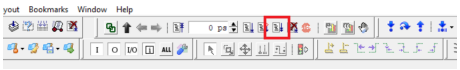
3. Finally ModelSim-Altera tool opens,
   In the Toolbar Click on **Simulate→Start Simulation** then a window will pop-up Add the Test bench file as shown in the below figure.

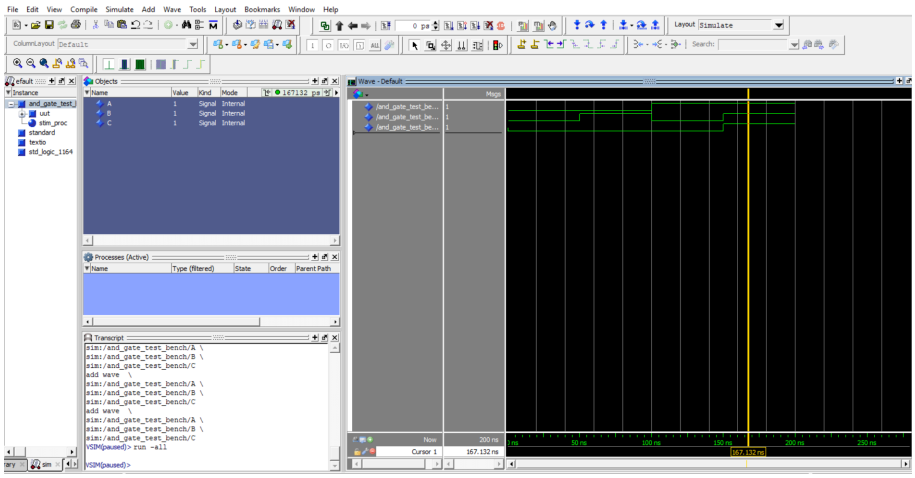


Add the I/O pins as shown in the below figure. Select all the signals (A , B and C) and add it to 'wave'section.



4. Start the simulation by clicking on the **Run All** button in Toolbar





5. **Navigating the simulation** At this point you should have successfully run the simulation, but the waveform window is rather small and hard to see. We can move around in the simulation and see the value of the signals. Look for the cursor, a yellow vertical line in the waveform viewer, with the time in yellow at the bottom. You can use this line to move left or right in the waveform viewer and also zoom-in and zoom-out.

- Move the yellow vertical line across each set of inputs and output in the waveform, so that you can see the level of each signal and compare them with the truth table of the And gate.
- By doing this you will notice that when both the inputs are high then only output is high or else it is low. which is actual logic for and gate.
- Follow this procedure for each experiment you perform to verify your design.

6 Timing Analysis using TimeQuest

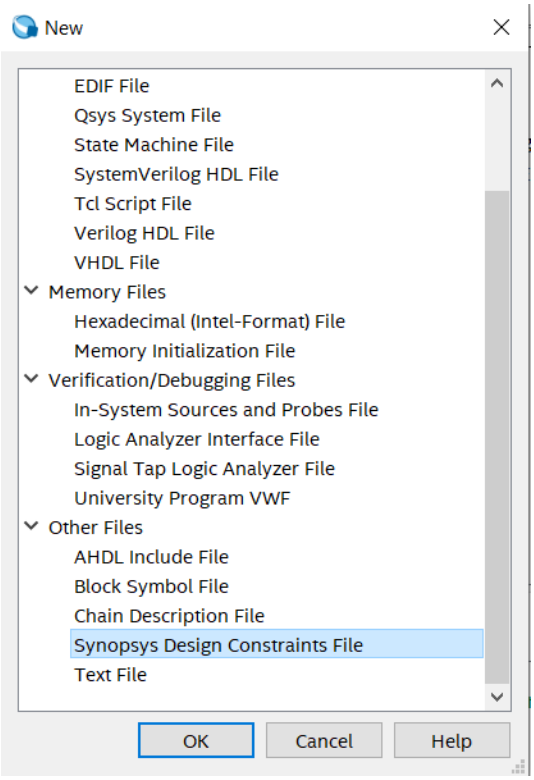
6.1 Introduction to timing analysis

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.
This section is not linked with the above part as this is just a walk through of how to add .sdc file and generate a timing analysis report.

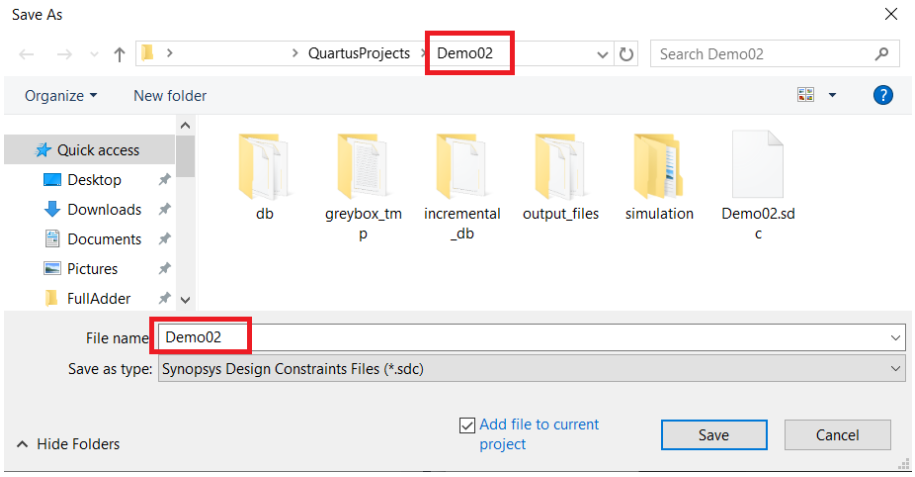
6.2 Creating a Synopsis Design Constraints file

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.



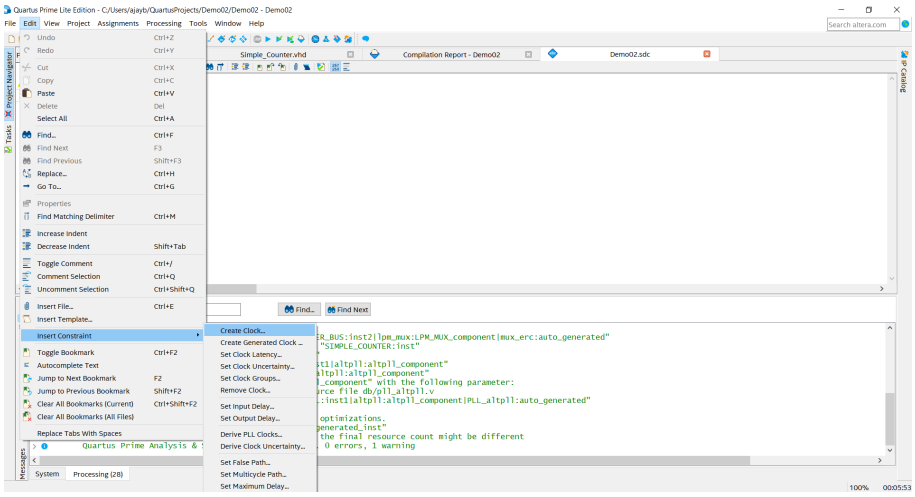
3. Click ok then save this file with the same name as our top-level entity file name So that this file will include in the main compilation process automatically, you can name the file differently but then you have to manual add it to Settings page.



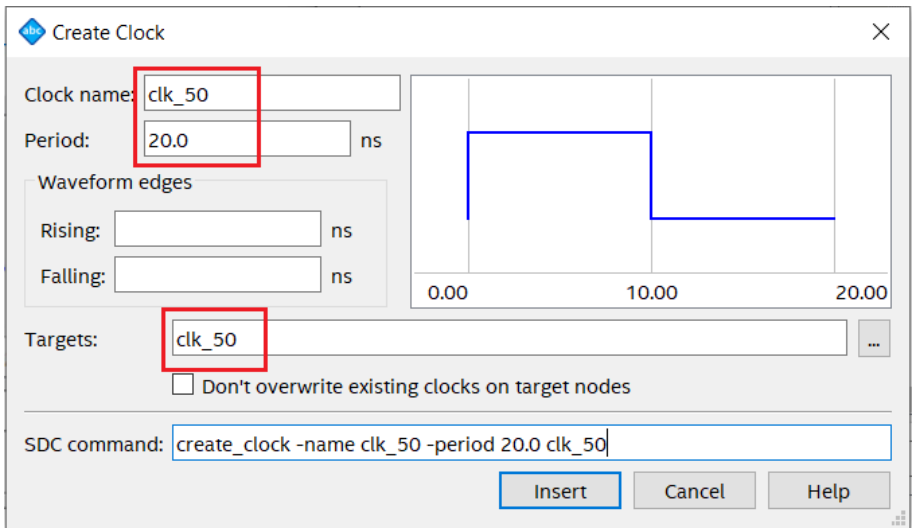
6.3 Adding Timing Constraints

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.

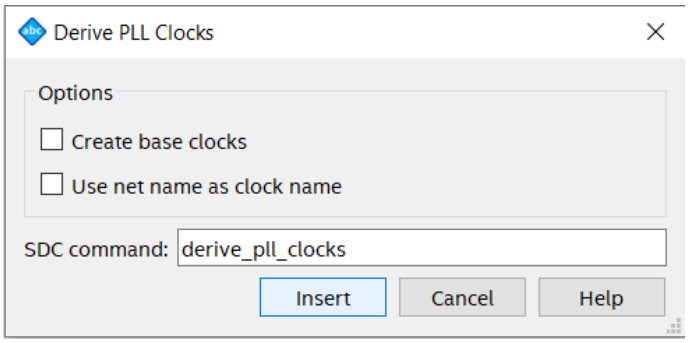


2. There are various types of constraints that could be used in our design, because all the timing related activities involves a clock.
   To create a clock for timing analysis, Select Create Clock from within Insert Constraint under the Edit menu.

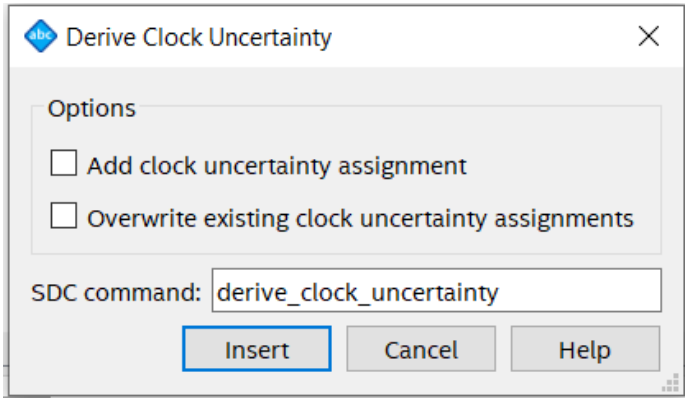


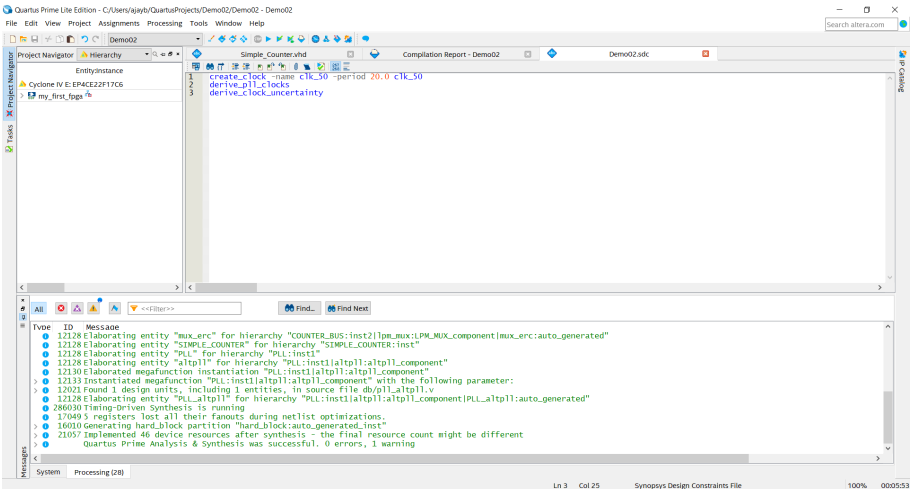
Enter the clock frequency in the period section, for example for a 50MHz clock period is 20ns.Then Click on Insert.

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



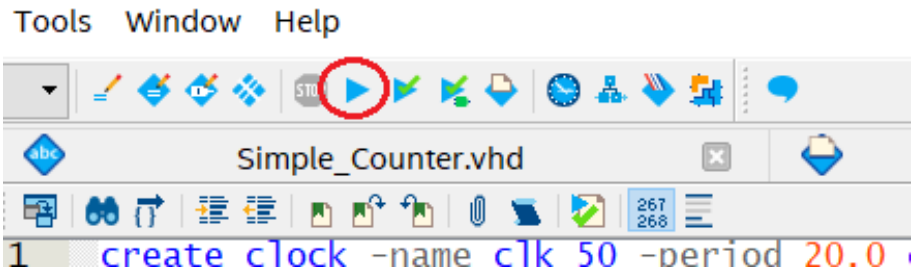
4. Insert Constraint again and this time select Derive Clock Uncertainty and save the file.



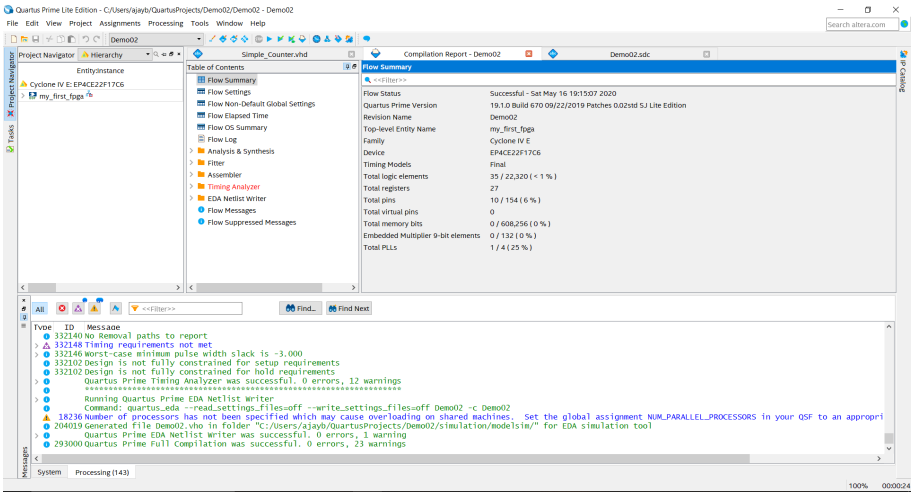


6.4 Running Full Compilation and TimeQuest Analysis

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

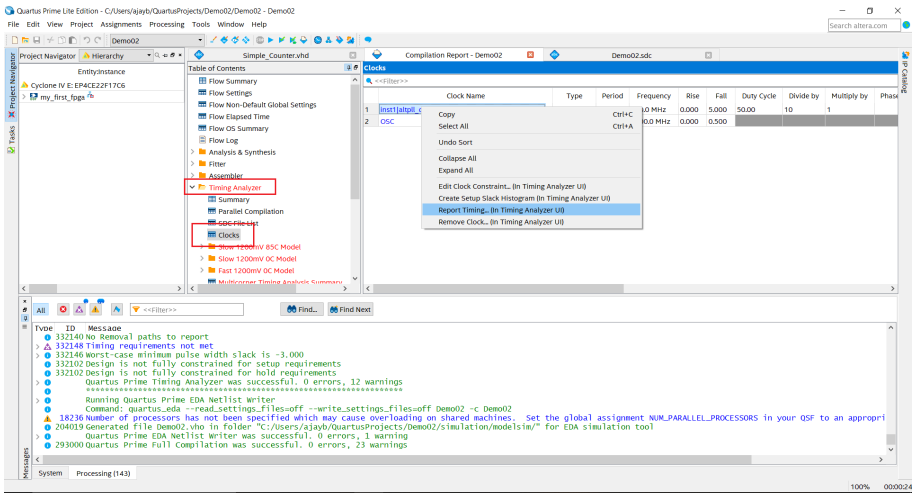


This step can take some time depending upon the complexity of your design and your system specification.



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)



- Applying the proper timing constraints is a necessity if you want your logic to perform properly.
- To ensure accurate data transfer between logic blocks that operate at very high frequency, timing analysis needs to play a more prominent.
- This analysis can also be used to find out the Maximum path delay of a circuit which can be used to find out the maximum clock frequency at which the circuit can operate.