

LAB # 1:

Introduction to VHDL and Altera Quartus with the design of combinational circuit

Objective

- How to program in hardware descriptive languages in general and VHDL in particular.
- How the software tool Quartus works and its interface
- How the FPGA design flow works from design specification to loading the bit file

Pre-Lab

Part 1 -Familiarize yourself with VHDL and Quartus:

Introduction to VHDL

VHDL is a hardware description language. It described the behavior of an electronic circuit or system, from which the physical circuit or system can then be attained (implemented). VHDL stands for VHSIC Hardware Description Language. VHSIC is itself an abbreviation for Very High Speed Integrated Circuits, an initiative funded by US DOD in the 1980s that led to the creation of VHDL. VHDL is intended for circuit synthesis as well as circuit simulation. However, though VHDL is fully simulated, not all constructs are synthesizable.

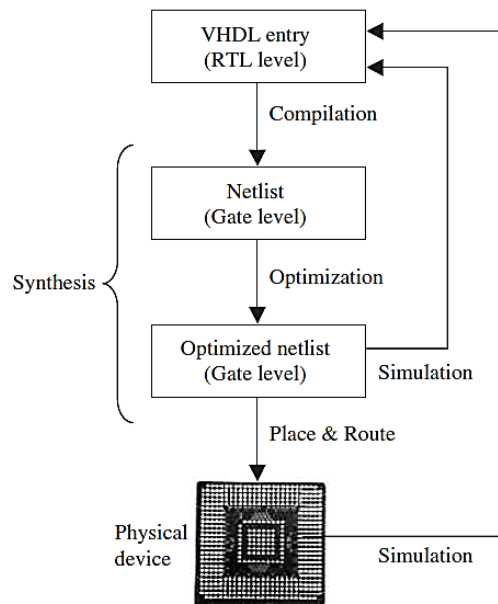


Figure 1-1 Summary of VHDL design flow

A fundamental motivation to use VHDL (or its competitor, Verilog) is that VHDL is a standard, technology/vendor independent language, and is therefore portable and reusable. The two main immediate applications of VHDL are in the field of Programmable Logic Devices (including CPLDs and FPGAs) and ASICs. A final note regarding VHDL is that, contrary to regular computer programs which are sequential, its statements are inherently concurrent (parallel). In VHDL, only statements placed inside a *Process*, *Function* or *Procedure* are executed sequentially.

Fundamental VHDL Units:

A standalone piece of VHDL code is composed of at least three fundamental sections:

- **LIBRARY** declarations: Contains a list of all libraries to be used in the design. For example: *ieee, std, work, etc.*
- **ENTITY**: Specifies the I/O pins of the circuit.
- **ARCHITECTURE**: Contains the VHDL code proper, which describes how the circuit should behave (function)

A **LIBRARY** is a collection of commonly used pieces of code. Placing such pieces inside a library allows them to be reused or shared by other design.

The typical structure of a library is illustrated in figure 1.2. The code is usually written in the form of **FUNCTIONS**, **PROCEDURES**, or **COMPONENTS**, which are placed inside **PACKAGES**, and then compiled into the destination library.

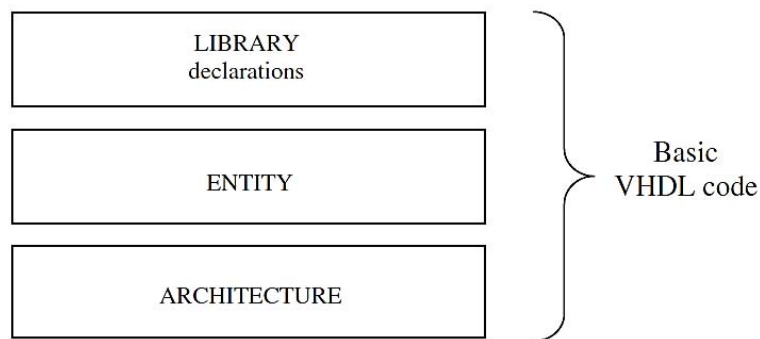


Figure 1-2 Fundamental sections of a basic VHDL code

Entity declaration:

VHDL files are basically divided into two parts, the entity and the architecture. The entity is basically where the circuits (in & out) ports are defined. There is a multitude of I/O ports available, but this lab will only deal with the two most usual ones, the INput and OUTput ports. (Other types

of ports are for example the INOUT and BUFFER ports.) The entity of the circuit in figure 1.3 should look something like below.

Please notice that comments in the code are made with a double-dash (--)

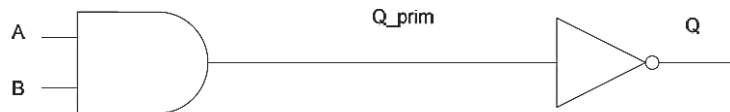


Figure 0-3 NAND gate using AND & NOT gate

Syntax:

```
entity entity_name is
    Port declaration;
end entity_name;
```

For Example:

```
ENTITY nandgate IS
PORT(
    A: IN BIT;
    B: IN BIT;
    Q: OUT BIT
);
END nandgate;
```

-- Note! No ';' here!

Now that we have defined the I/O interface to the rest of the world for the NAND gate, we should move on to the architecture of the circuit.

Architecture:

The entity told us nothing about how the circuit was implemented, this is taken care of by the architecture part of the VHDL code. The architecture of the NAND gate matching the entity above could then be written as something like this...

Syntax:

```
architecture architecture_name of entity_name
architecture_declarative_part;
begin
    Statements;
```

end architecture_name;

For Example:

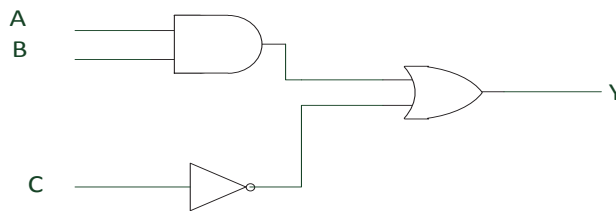
```

ARCHITECTURE dataflow OF nandgate IS
    SIGNAL Q_prim: BIT;
BEGIN
    Q_prim <= A AND B; -- The AND-operation.
    Q <= NOT Q_prim; -- The inverter.
END dataflow;

```

Lab Tasks

Lab Task 1:



Library IEEE;

Use IEEE.STD_LOGIC_1164.all;

Entity circuit is

Port (A, B , C: IN bit; (IEEE standard which defines a nine-value logic type)
Y : OUT bit);

End circuit;

Architecture behavior of circuit is

Begin

Y<= (A AND B) OR (NOT C);

End behavior;

Part 2 – Familiarize yourself with Quartus Tool

Introduction to Quartus ISE:

Altera Quartus II is design software produced by Altera. Quartus II 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 includes an implementation of VHDL and Verilog for hardware description, visual editing of logic circuits, and vector waveform simulation.

Lab Task 2: Create a New Project

1. Start the Quartus II software.

From the Windows Start Menu, select:

All Programs → Other Apps → Altera → Quartus II 13.1 → Quartus II 13.1

(32-Bit)

2. Start the New Project Wizard.

If the opening splash screen is displayed, select: **Create a New Project (New Project Wizard)**, otherwise from the Quartus II Menu Bar select: **File → New Project Wizard**.

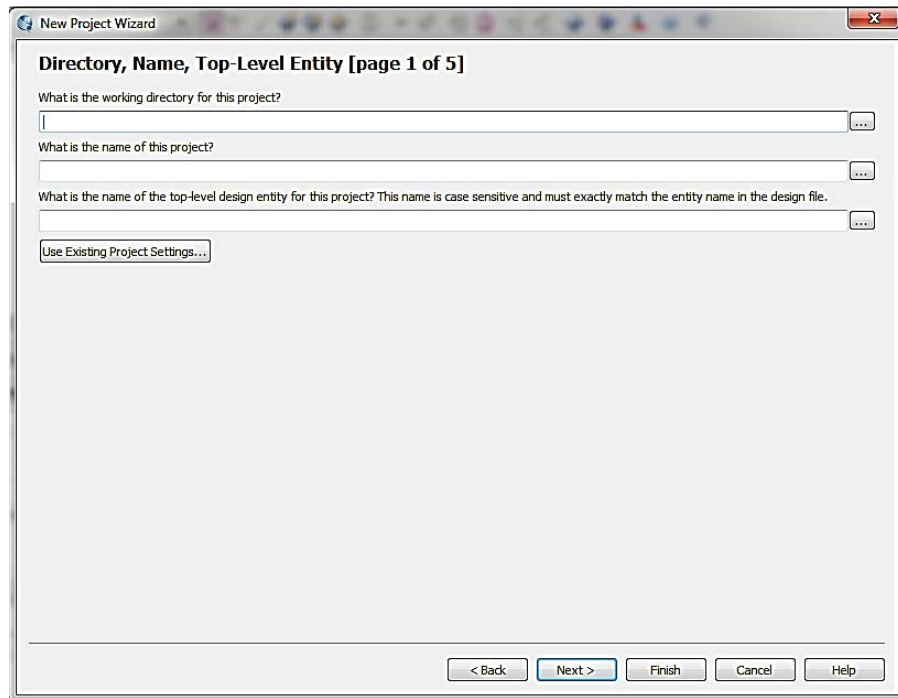
3. Select the Working Directory and Project Name.

Working Directory	H:\Altera_Training\Lab1
Project Name	Lab1
Top-Level Design Entity	Lab1

Click **Next** to advance to page 2 of the New Project Wizard.

Note: A window may pop up stating that the chosen working directory does not exist.

*Click **Yes** to create it*



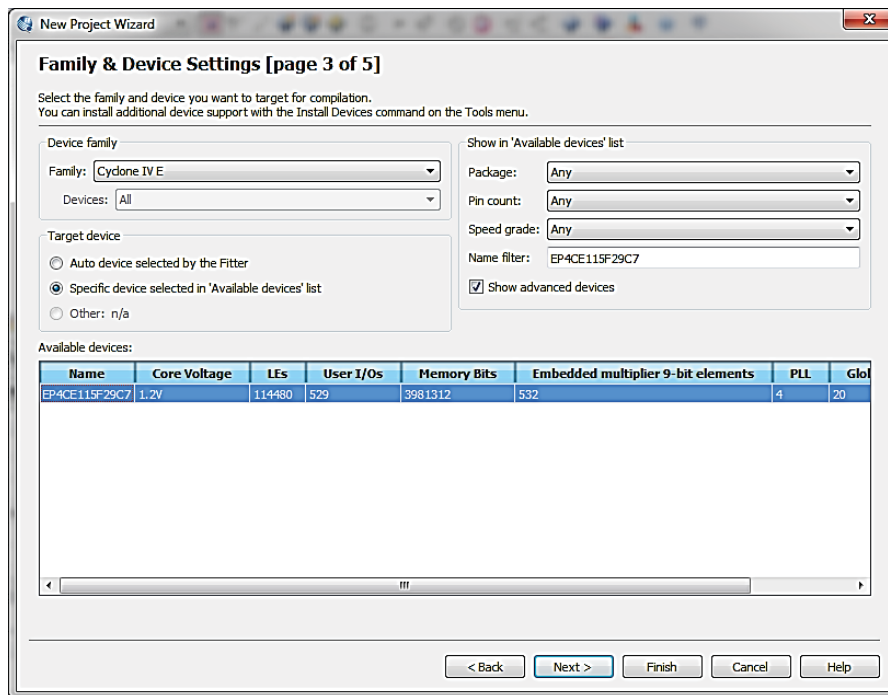
4. Click **Next** again as we will not be adding any preexisting design files at this time.

5. **Select the family and Device Settings.**

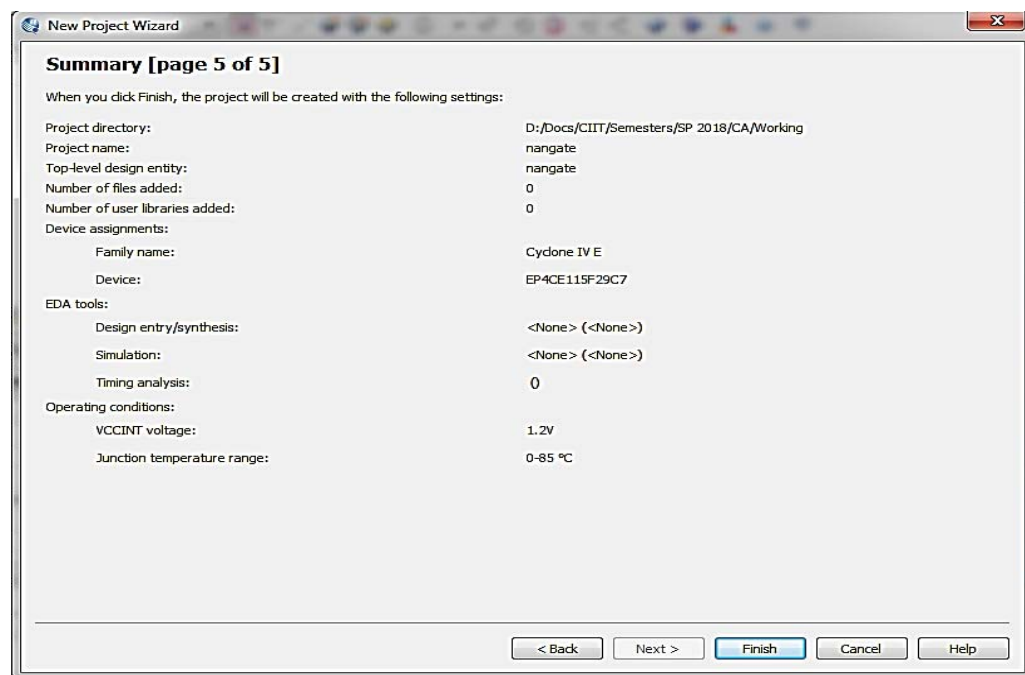
From the pull-down menu labeled **Family**, select **Cyclone IV**.

In the list of available devices, select **EP4CE115F29C7N**.

Click **Next**.



6. Click **Next** again as we will not be using any third party EDA tools
7. Click **Finish** to complete the New Project Wizard

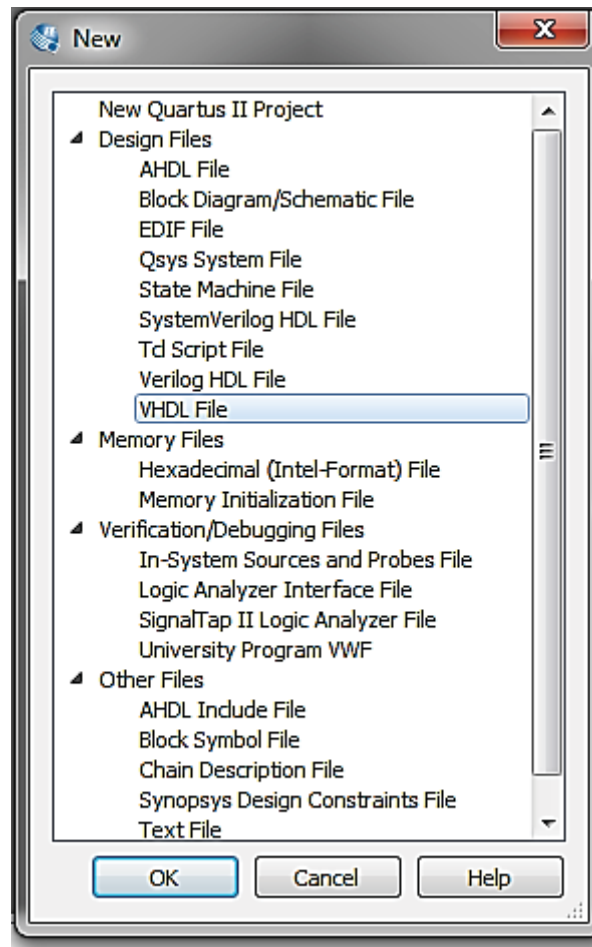


Task 2: Create, Add, and Compile Design Files

1. Create a new Design File.

Select: **File**→ **New** from the Menu Bar.

Select: **VHDL File** from the **Design Files** list and click **OK**.



- Copy and paste the following code into your new VHDL file, then save it by selecting **File** → **Save**.

Name the file **nandgate** and click **Save** in the **Save As** dialog box.

```
library IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
```

```
entity nandgate is
port
(
    A: in std_logic;
    B: in std_logic;
    cout : out std_logic
);
end nandgate;
```

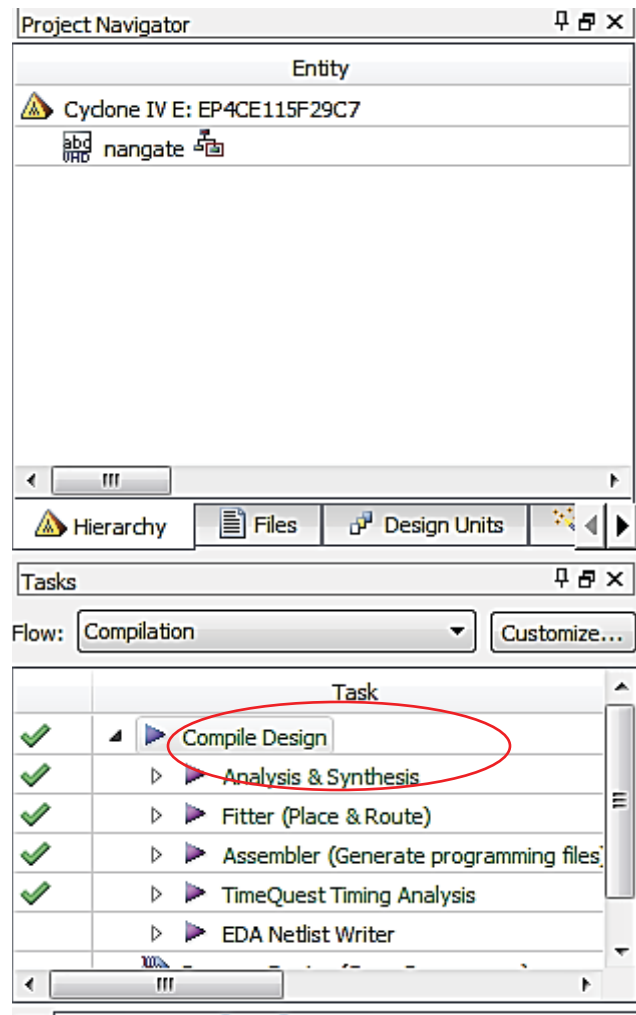
```
architecture behavioral of nandgate is
```


begin

cout <= not (A and B);

end behavioral;

3. Now Click on **Compile Design** to look for errors in code

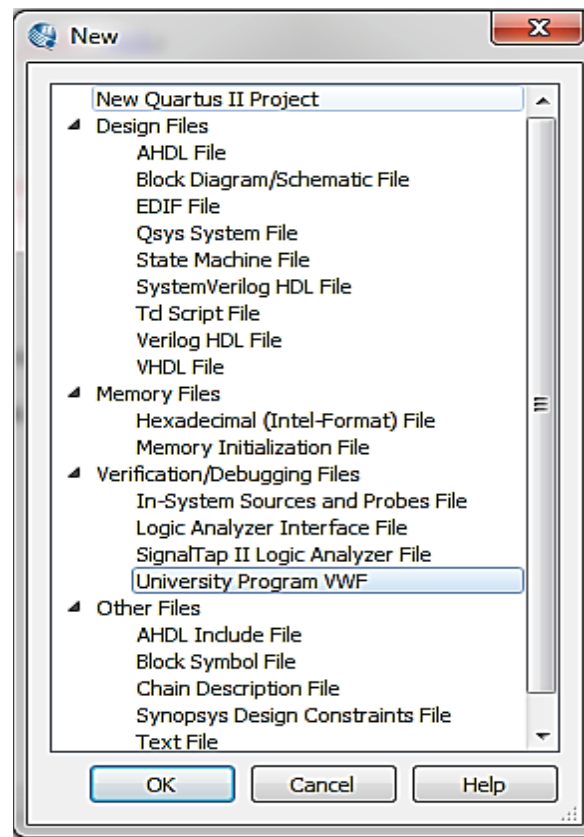


Task 3: Simulate Design using Quartus II

1. Create a Vector Waveform File (vwf)

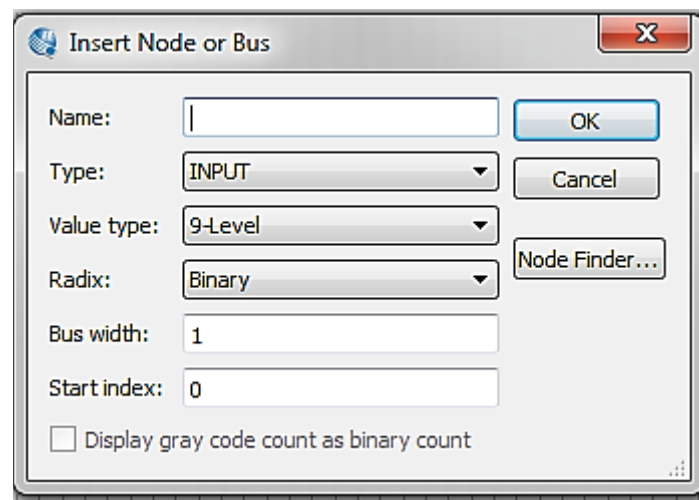
Click the **New File** icon on the Menu Bar

Select **Vector Waveform File** under the **Verification/Debugging Files** heading.



2. Add Signals to the Waveform


Select **Edit** → **Insert** → **Insert Node or Bus...** to open the **Insert Node or Bus** window.

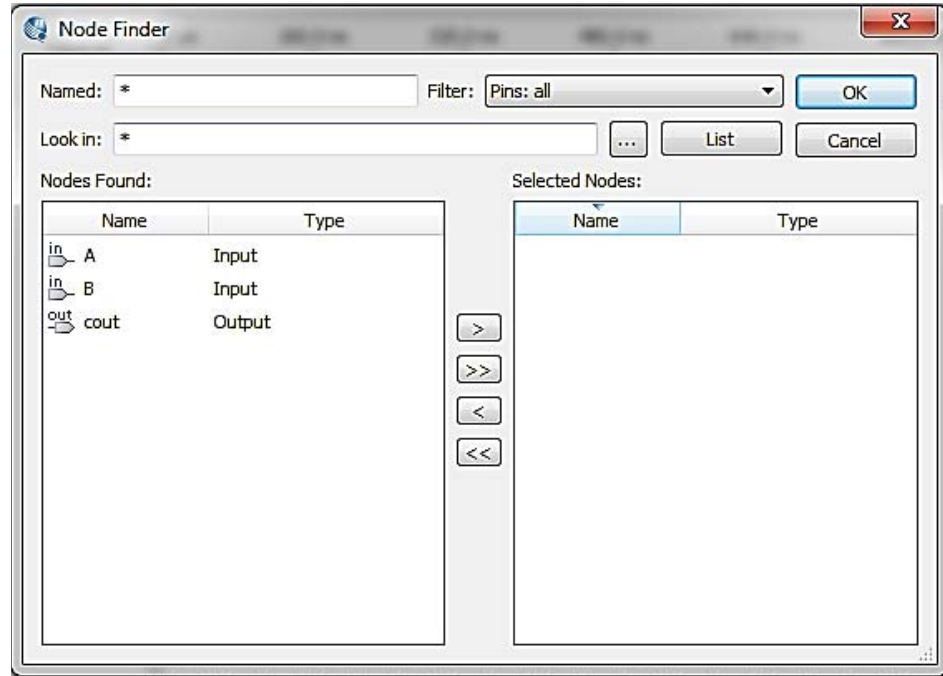


- Click the **Node Finder...** button in the **Insert Node or Bus** window to open the **Node Finder** window.

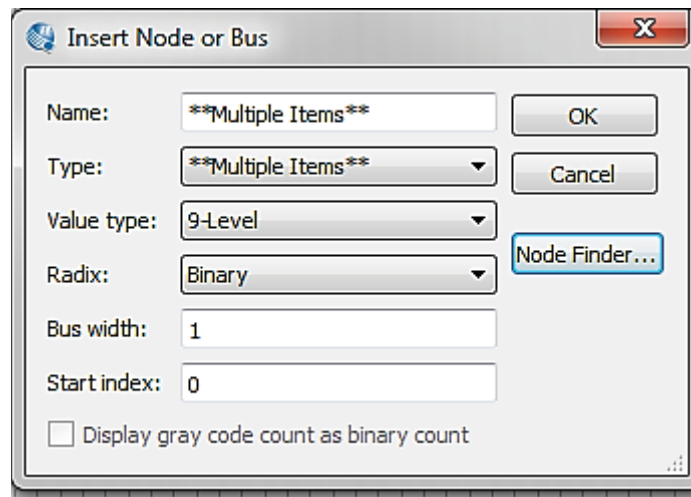
From the **Filter** drop-down menu, select **Pins: all** and click the **List** button. This will display all the inputs and outputs to the circuit.

Highlight all the **Input Groups A, B, and Output Group Sum** in the **Nodes**

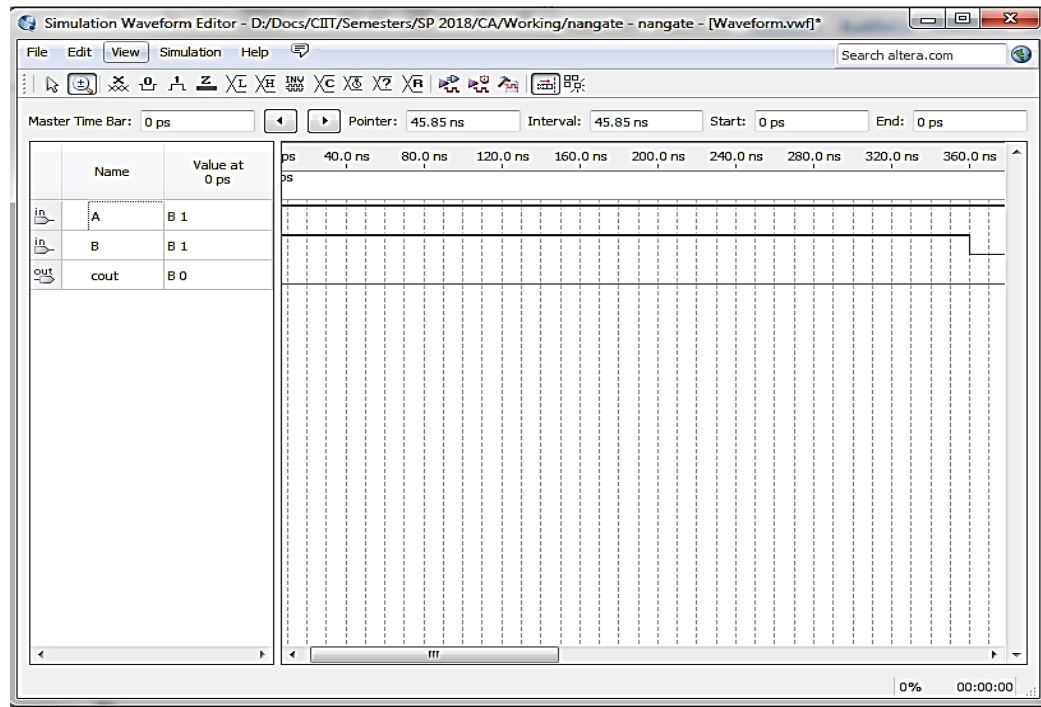
Found window and click the right arrow  to select them. Then click **OK** to close the **Node Finder** window.



4. In the **Insert Node or Bus** window, change **Radix** to **Binary**. This will make it easier for us to enter values and interpret the results.
- 5.



6. Input waveforms can be drawn in different ways. The most straightforward way is to indicate a specific time range and specify the value of a signal. To illustrate this approach, click the mouse on the A waveform near the 0-ns point and then drag the mouse to the 400-ns point. The selected time interval will be highlighted in blue. Press **Ctrl + Alt + B** to open the **Arbitrary Value** window to enter the values manually. We will use this approach to create the waveform for B, which should change from 0 to 400 ns.
- 7.



Leave **Sum** with a value of undefined (X), as values for this will be generated when the simulation runs.



Click the **Save** icon on the Menu bar and save the vwf file with the filename **nandgate.vwf**.

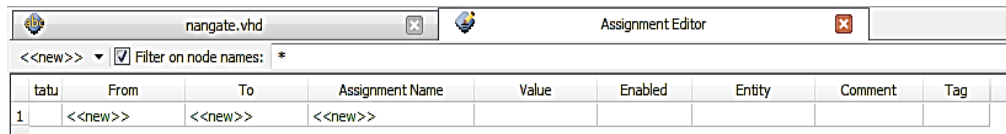
8. In the Main window, select *Simulation / Options* and then select *Quartus II Simulator*.
Select *OK*.
9. In the Main window, select *Simulation* and then select *Run Functional Simulation*.
10. Now you should see your simulation output with the outputs defined.

Note: The file will indicate "read-only" meaning you can't edit it.

Task 4: Implementing the Design on the DE2 Board

1. From the Menu Bar select: **Assignments** → **Assignment Editor**

- Double-click <<new>> in the **To** column and select **Node Finder** button, List down all the input and output pins and click OK.



- Double-click the adjacent cell **Assignment Name** and select **Location (Accepts wilds cards/groups)**.

Double-click the adjacent cell **Value** and assign the pin number.

Continue to assign the pins as seen in the table below.

	To	Location	DE2 Board Description
1	A	PIN_AB28	SW0
2	B	PIN_AC28	SW1
3	Cout	PIN_G19	LED0

- Save the pin assignments by selecting **File**→ **Save** from the Menu Bar, or by clicking the Save

button  on the toolbar.

- Compile the design again by clicking the Start Compilation button on the toolbar



- Plug in and power on the DE2 board. Make sure that the **RUN/PROG Switch for JTAG/AS Modes** is in the **RUN** position.

- In the Quartus II window click the **Programmer** button on the Toolbar to open the

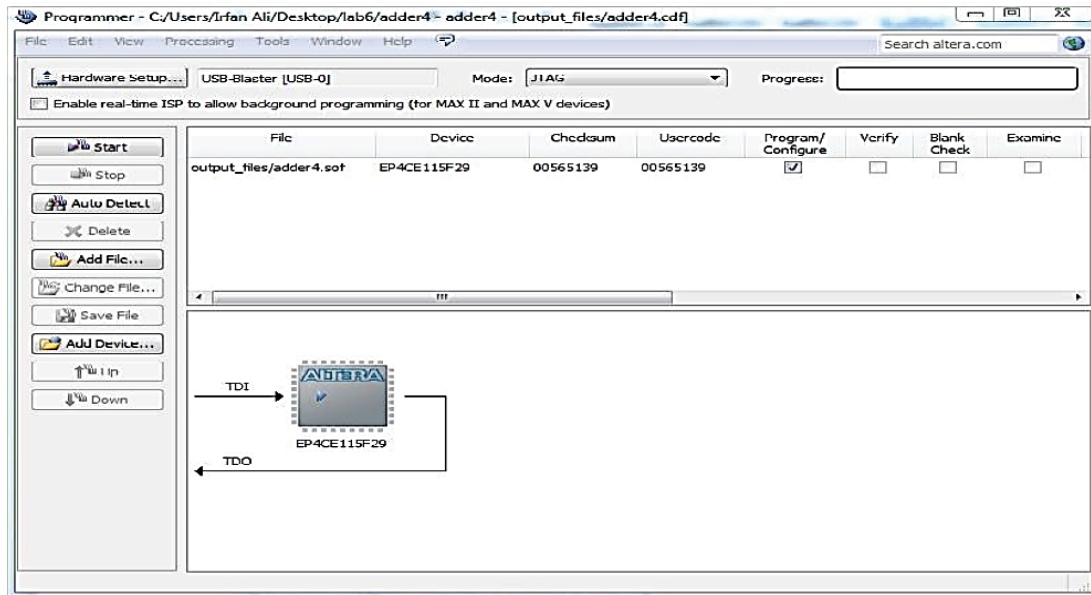


Programmer window

The **Hardware Setup...** must be **USB-Blaster [USB-0]**. If not, click the **Hardware Setup...** button and select **USB-Blaster [USB-0]** from the drop-down menu for currently selected hardware.

Mode should be set to **JTAG**.

Make sure that the **File** is **adder4.sof**, **Device** is **EP4CE115F29C7N** and the **Program/Configure** box is checked.



Then click the **Start** button to program the DE2 board.

When the progress bar reaches 100%, programming is complete.

8. You can now test the program on the DE2 board by using the toggle switches located along the bottom of the board.

SW0 bit of data for A.

SW1 bit of data for B.

The output of the operation is displayed on the first LED (LED0) located above the blue push buttons on the DE2 board.

Rubric for Lab Evaluation

		Exceeds Expectations	Meets Expectation	Developing	Unsatisfactory
Hardware		[4]	[3]	[2]	[1]
	Setup of experiment and implementation (hardware/simulation)				
	Follow the procedure/design process				
	Experimental results				
	Safety				
	Viva				
	Hardware Score (Average)				
Software		Exceeds Expectations	Meets Expectation	Developing	Unsatisfactory
		[4]	[3]	[2]	[1]
	Ability to use software				
	Ability to follow procedure and/or design a procedure for experiment				
	Ability to troubleshoot software				
	Q & A				
	Software Score (Average)				
Written Lab Report		Exceeds Expectations	Meets Expectation	Developing	Unsatisfactory
		[4]	[3]	[2]	[1]
	Data Presentation				
	Data Analysis				
	Writing Style				
	Written Lab Report Score (Average)				
Total Score					
		Signature & Date:			