EXPERIMENT 2

Inverting Logic: NOT, NAND, & NOR

OBJECTIVES:

- Examine inverting logic circuits.
- Demonstrate the characteristics of NOT, NAND, and NOR gates.
- Develop truth tables for NOT, NAND, and NOR gates.

MATERIALS:

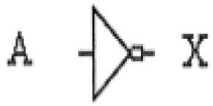
- Xilinx Vivado software, student or professional edition V2018.2 or higher.
- IBM or compatible computer with Pentium III or higher, 128 M-byte RAM or more, and 8 G-byte Or larger hard drive.
- BASYS 3 Board.

DISCUSSION:

The inverter (or NOT gate) represents logical complementation. A NOT gate can have only one input and one output. The output of a NOT gate simply reverses (inverts) the logic value presented at its input. The NOT gate can be combined with AND and OR gates to construct two more basic gates: NAND and NOR gates. Both NAND and NOR gates are universal logic gates, which means that either NAND gates or NOR gates can be used to construct any combinational logic circuit. We will use gate symbols, truth tables, and Boolean equations to demonstrate their characteristics. As with AND and OR gates, NAND and NOR gates can have two or more inputs but only one output.

Gate Characteristics:

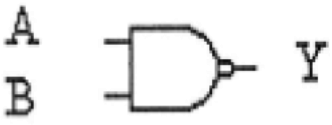
1. The NOT Gate

Symbol	Boolean Equation	Truth Table
	$X = \overline{A}$	Input Output A | X 0 | 1 1 | 0

Because the NOT gate has only one input, the truth table has two rows. Moreover, the output inverts the logic level of the input. In addition to the overhead bar shown above (read as "X = A-bar'), notation for logical inversion includes the following:

!A, /A, ¬A, A*

2. The NAND Gate

Symbol	Boolean Equation	Truth Table
	$Y = \overline{A\,B}$	Inputs Output A | B | Y 0 | 0 | 1 0 | 1 | 1 1 | 0 | 1 1 | 1 | 0

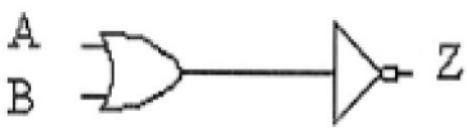
The behavior of a NAND gate can be summarized as follows: The output is LOW only when all the inputs are HIGH. If one or more inputs are LOW (false or logic 0), the output will be HIGH. Comparing the truth table for the NAND gate with that of the AND gate, you will find out that each output of a NAND gate is exactly the opposite (inverted) logic value of the corresponding output of an AND gate. In fact, a NAND gate is functionally equivalent to an AND gate cascaded with a NOT gate as shown below.

3. The NOR Gate

Symbol	Boolean Equation	Truth Table
	$Z = \overline{A + B}$	Inputs / Output

A	B	Z
0	0	1
0	1	0
1	0	0
1	1	0

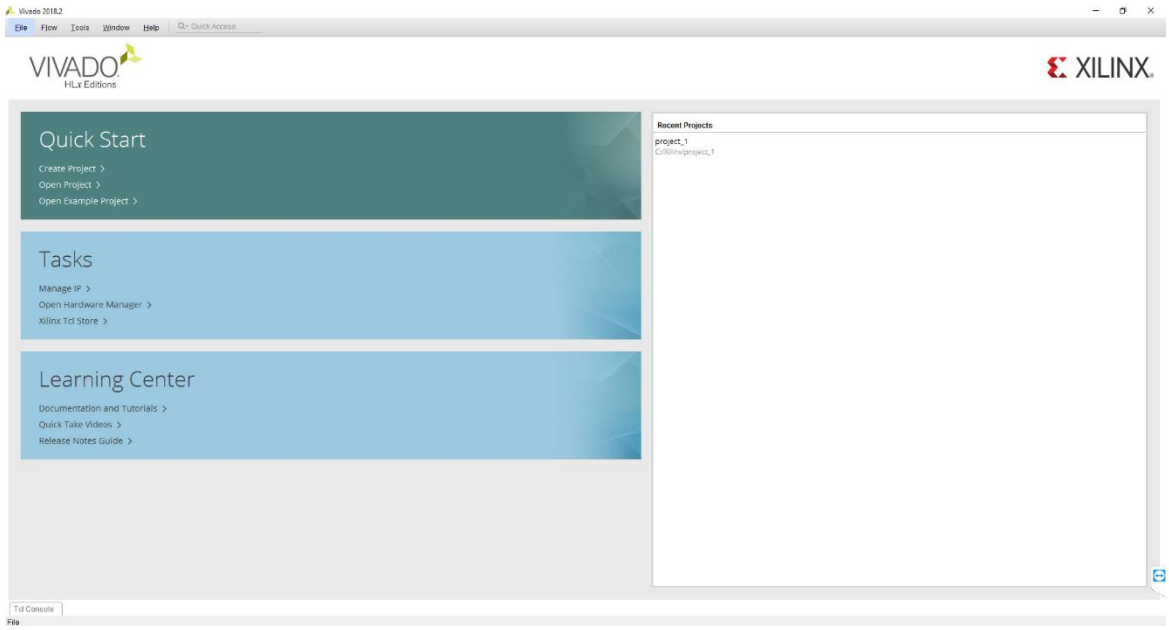
As seen from the above truth table, the output of a NOR gate is HIGH only when ***all*** the inputs are LOW. If one or more of the inputs are HIGH, then the output is LOW. Similarly, a NOR gate can be constructed using an OR gate cascaded with a NOT gate. In other words, a NOR gate is functionally equivalent to an OR gate followed by an inverter.



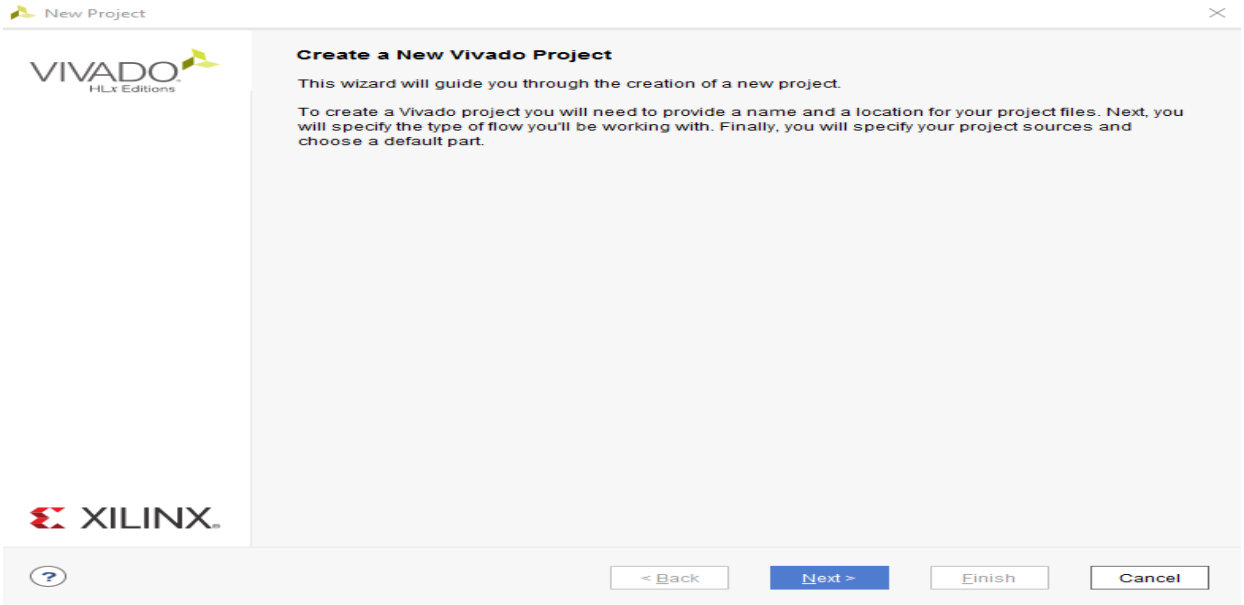
In the later part of this experiment, we will show how NAND and NOR gates can be used to perform some useful functions such as enabling and disabling signals. Also, we will show how to use NAND and NOR gates to perform the function of a NOT gate.

PROCEDURE:

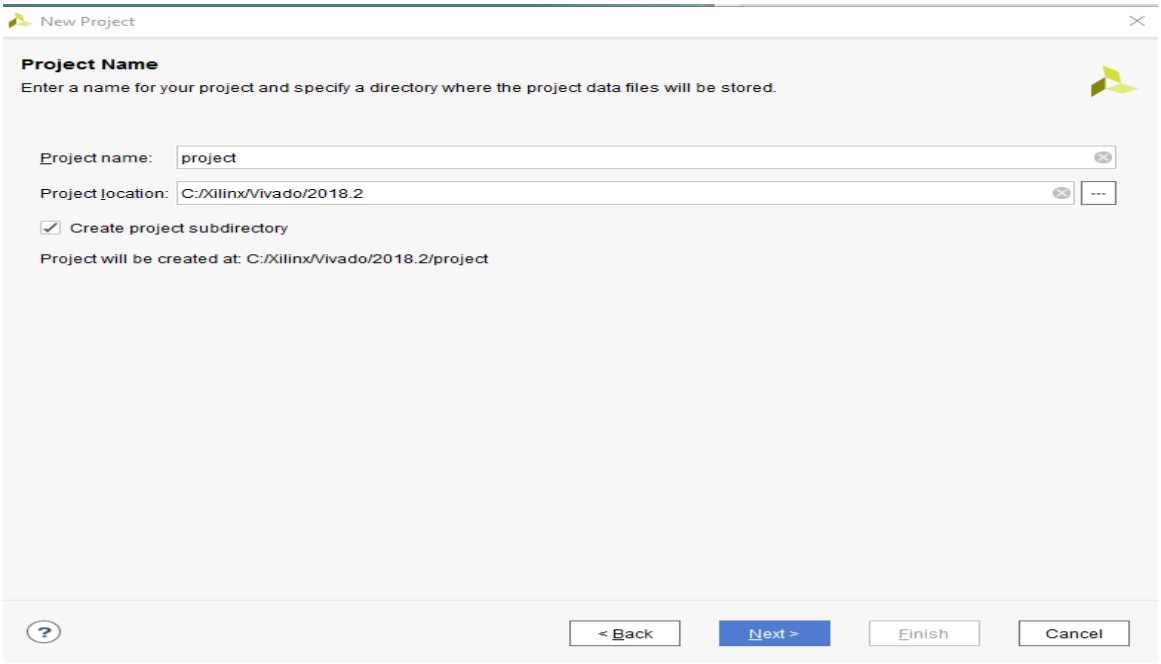
1. Open Xilinix Vivado.



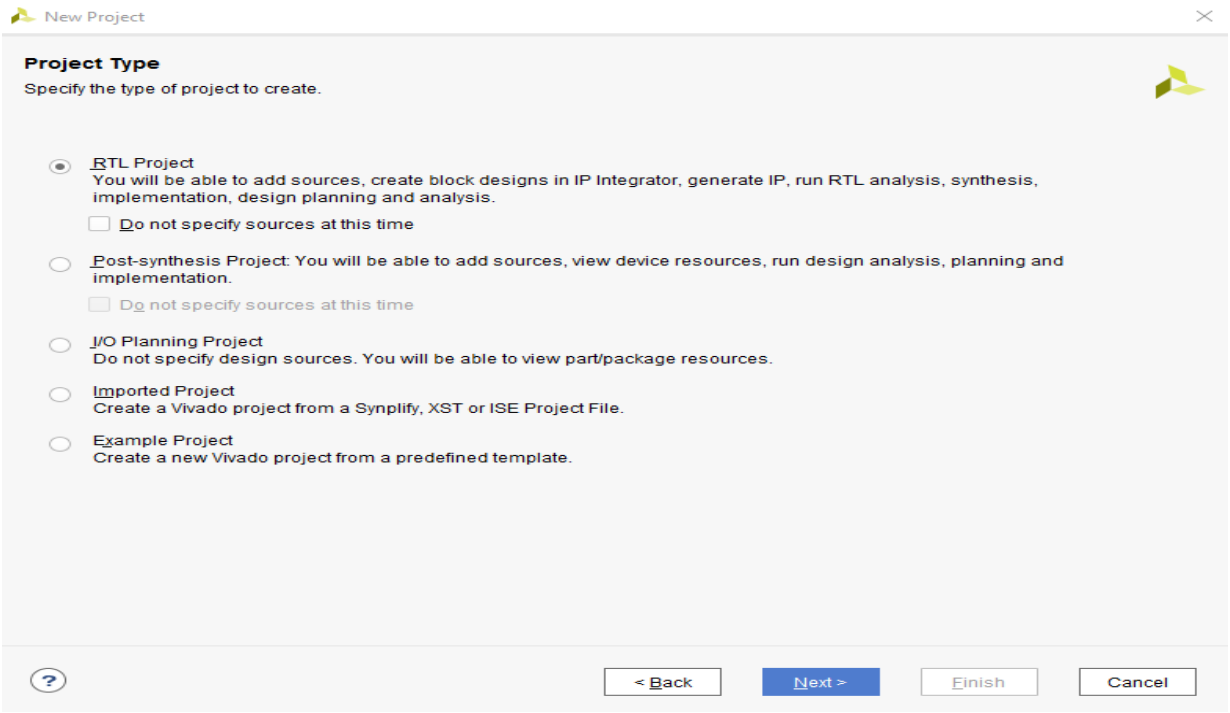
2. In the **Xilinx-Project Navigator** window, Quick start, **New Project**.



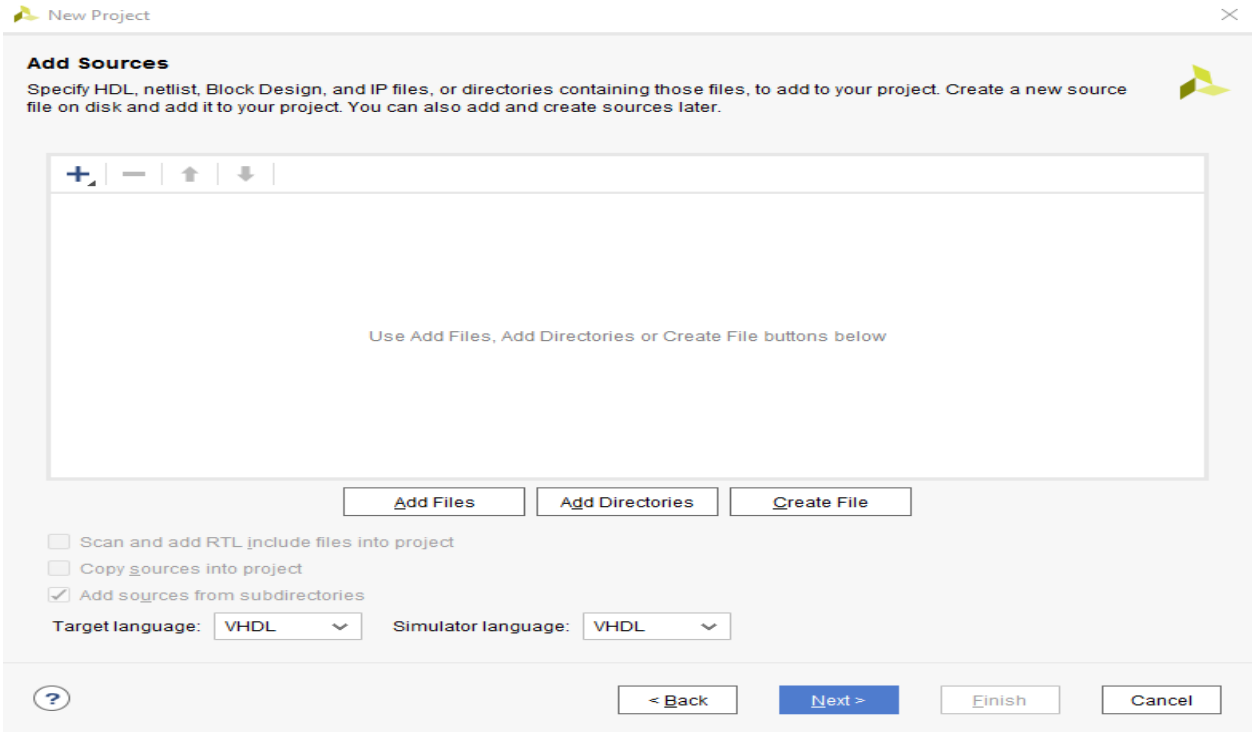
3. Name the project.

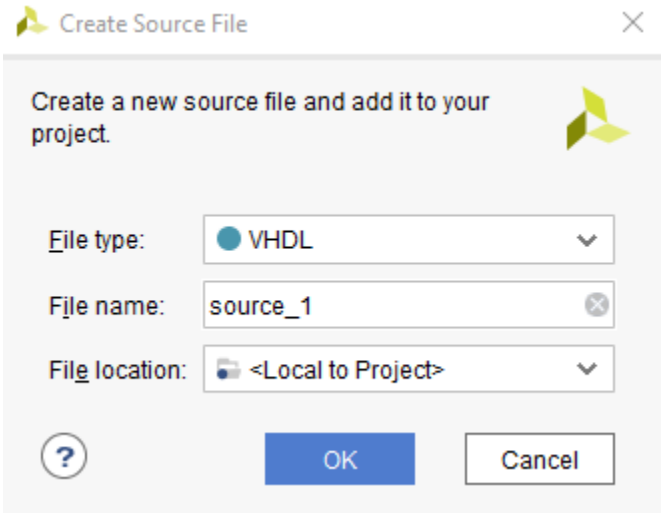


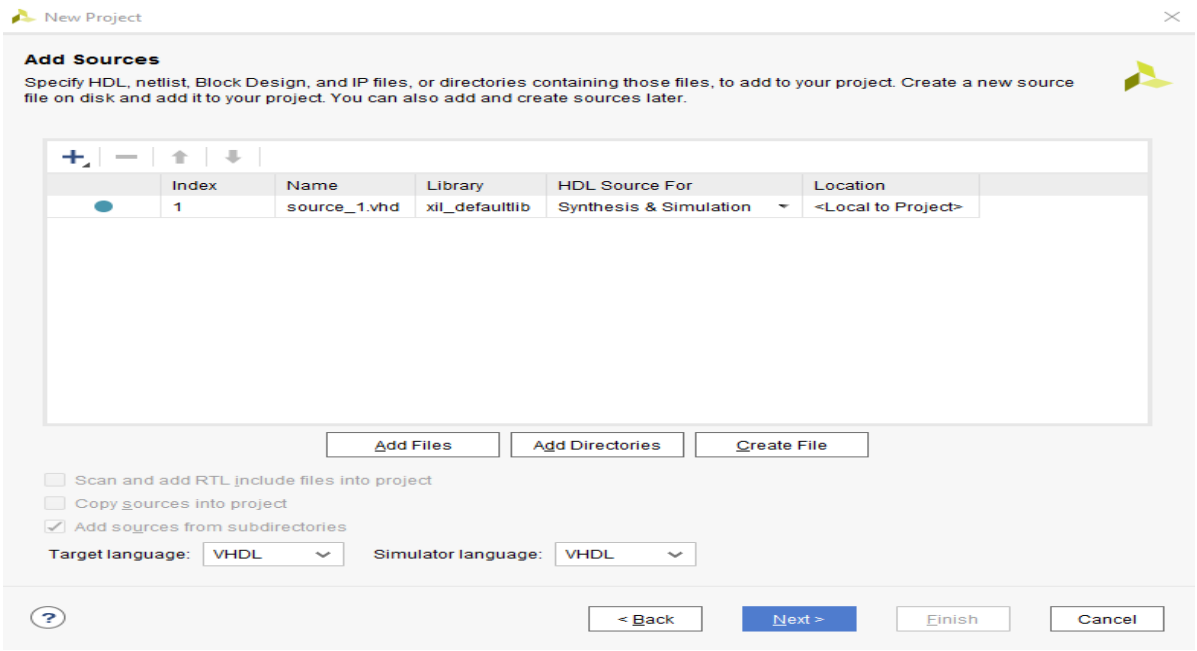
4. Choose "RTL Project" and check the "Do not specify sources at this time" as we will configure all the settings manually through the navigator from inside the project.

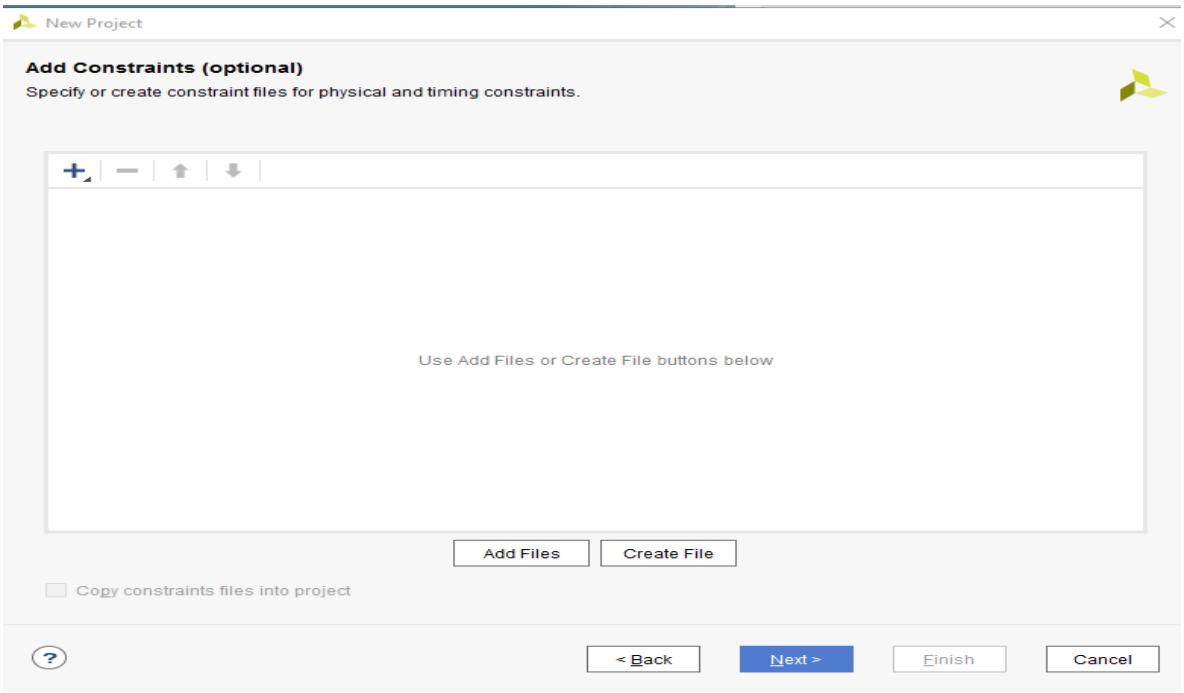


5. Select **New Source…** and the **New** window appears. In the New window, choose Schematic, type your file name (such as ***source_1***) in the File Name editor box, click on OK, and then click on the Next button.



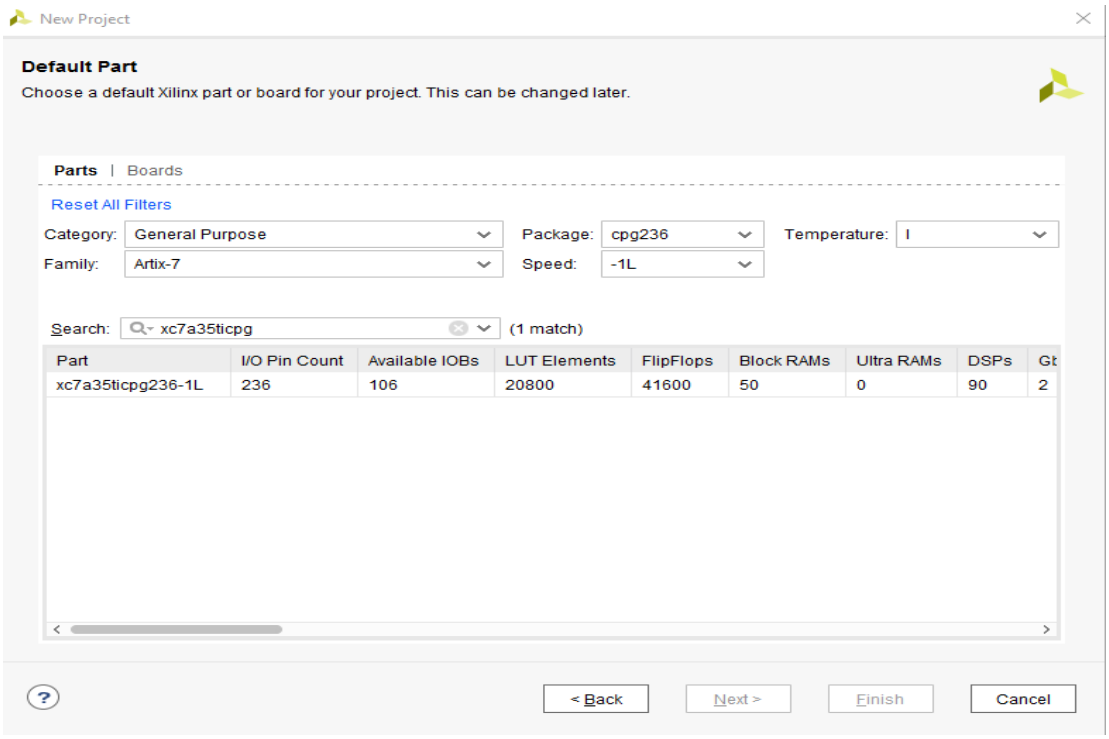


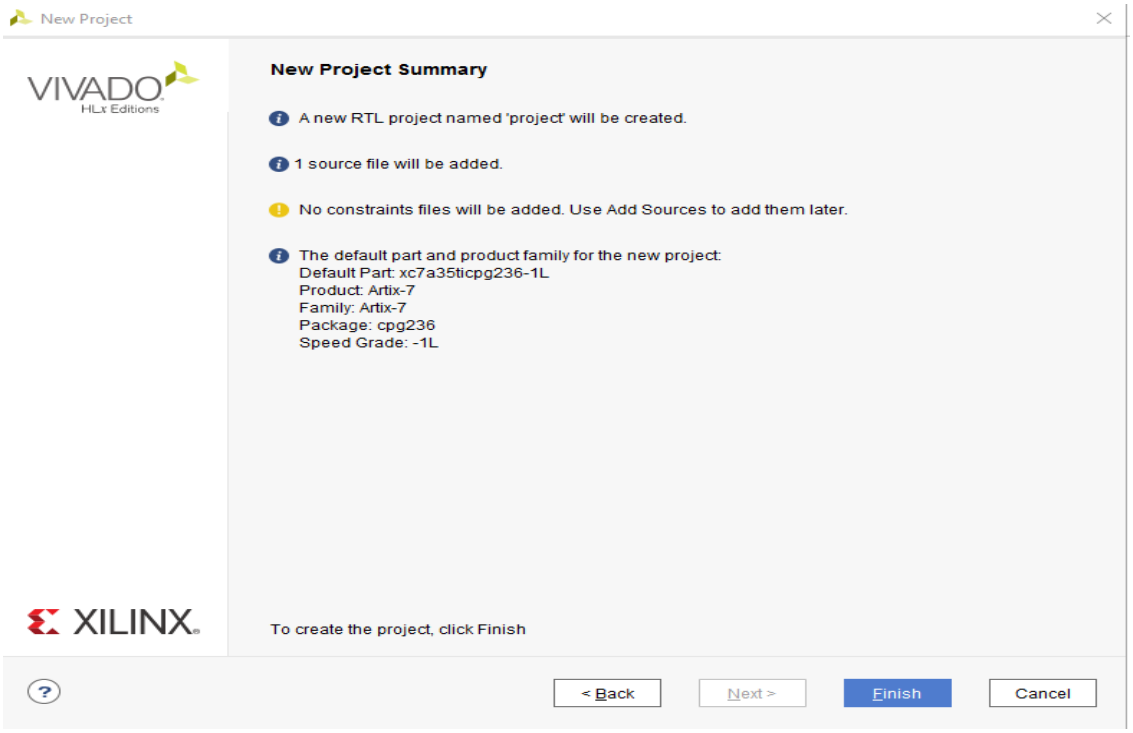




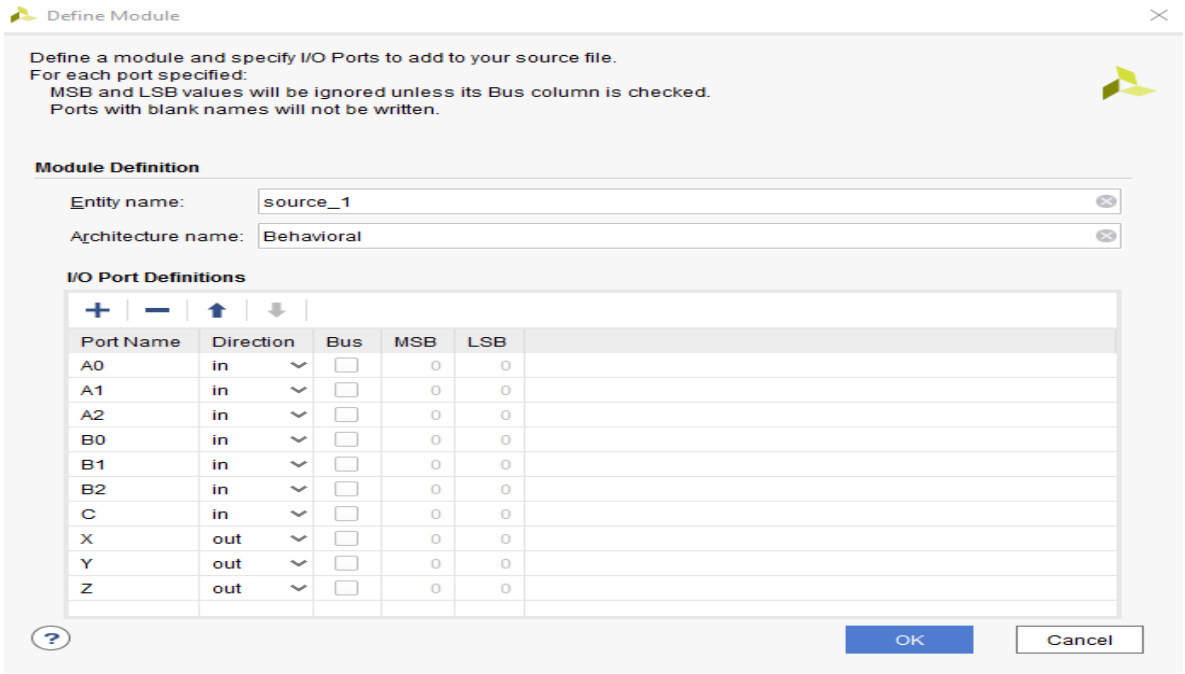
6. In the **Xilinx - Project Navigator** window, select the following
   - Category: "General Purpose"
   - Family: "Artix-7"
   - Package: "cpg236"
   - Speed: "-1"
   - Choose "xc7a35tcpg236-1" that corresponds to the board we are using.

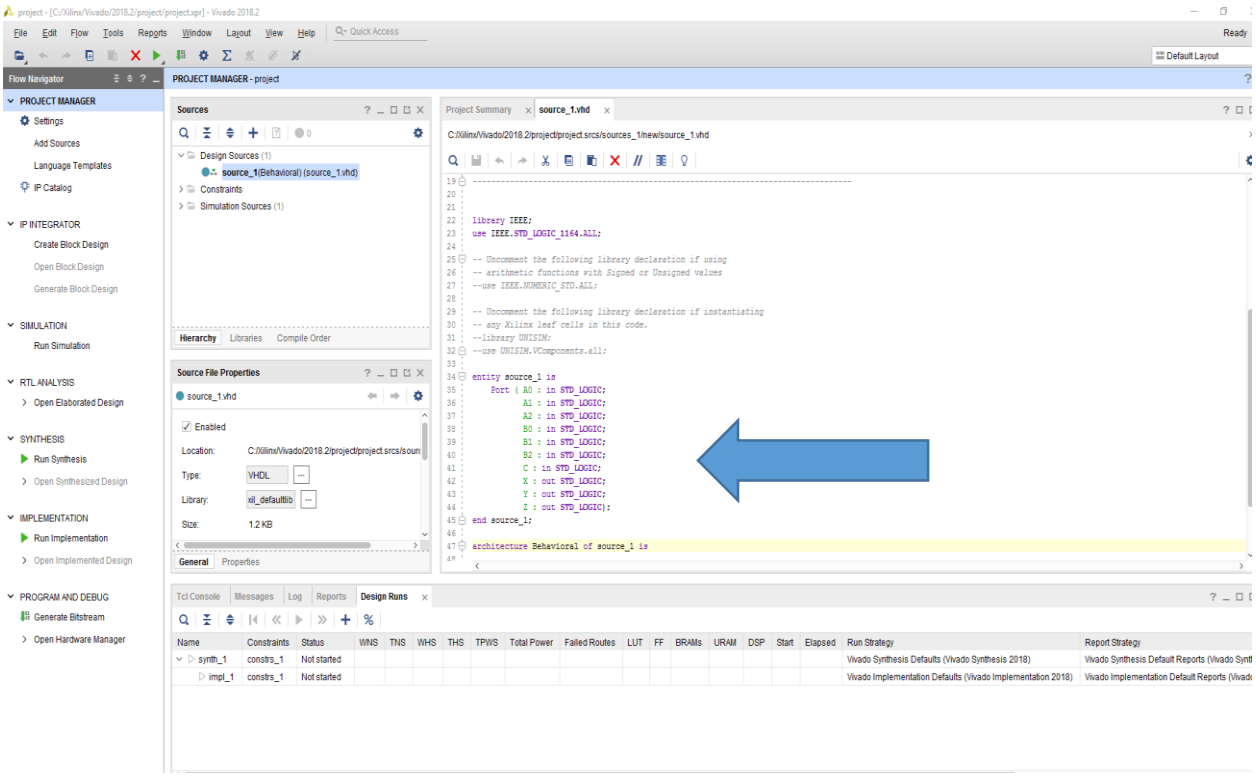
Then Choose Finish.



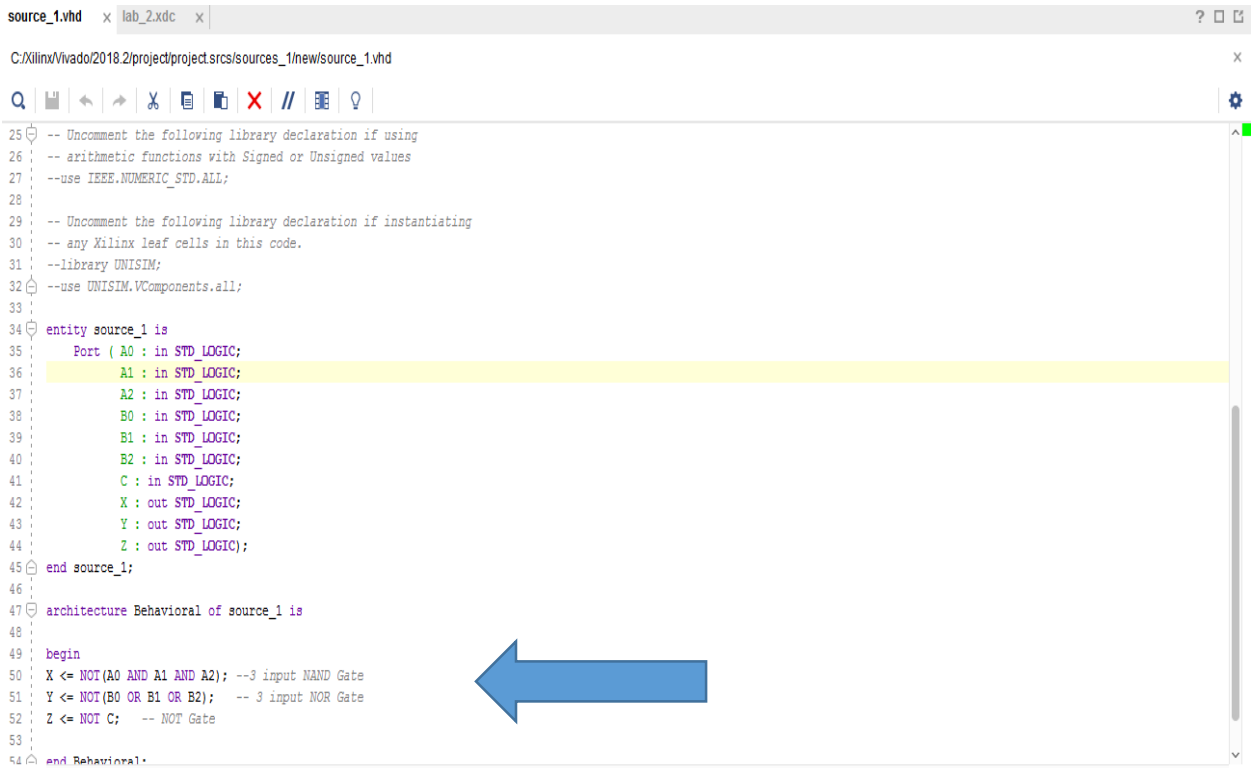


7. The Define Module Window that will appear, we will choose the input and output labels for the gates under investigation in this experiment. In this experiment, we are investigating a 3-input NAND gate and 3-input NOR and a NOT (Inverse) gate. Then Under "Port Name", add "A0", "A1" , "A2" as inputs for NAND gate, add "B0", "B1" , "B2" as inputs for NOR gate and add "C", as inputs for the NOT gate. Then add "X", "Y" , "Z" as outputs for the mentioned gates and select OK.

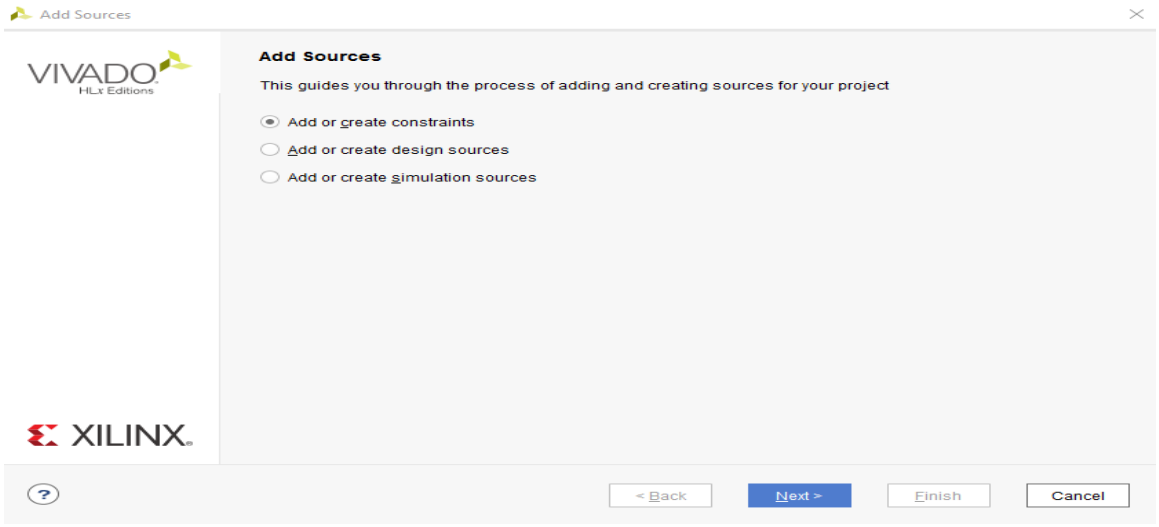


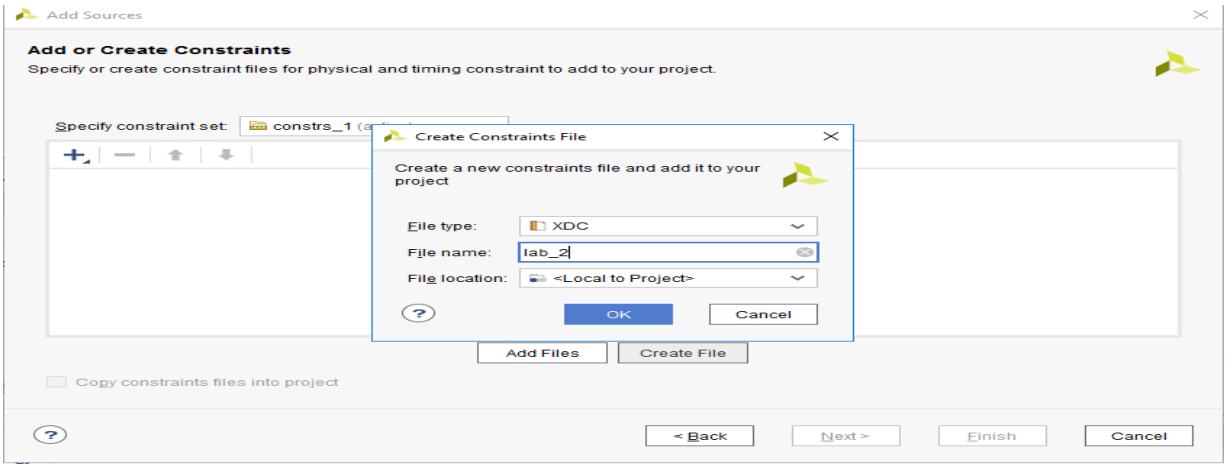


8. In the “source_1.vhd” created file, type the gates equivalent VHDL code for the NAND, NOR and NOT gates between the “begin” and “end Behavioral” as follows and then save the file.



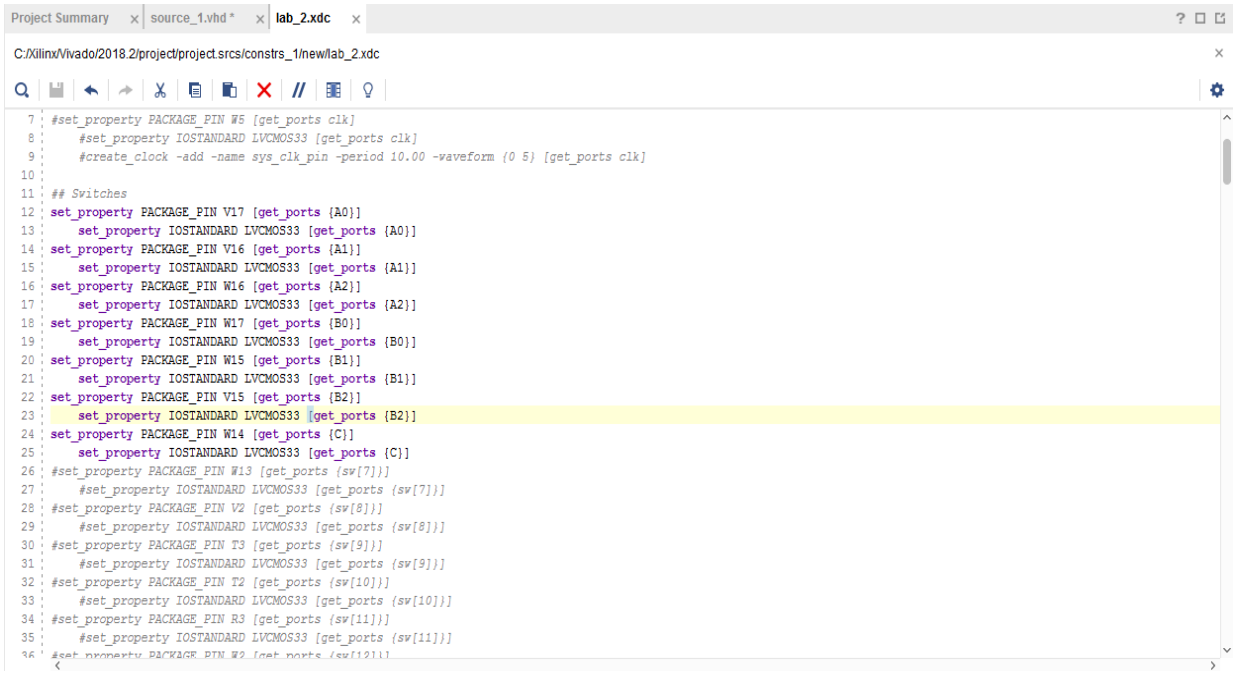
9. Next, we need to add To add a constraint file with the”.xdc” extension, as following: Go to “Flow Navigator” and from “Project Manager” select “Add Sources” then “Add or create constraints”. Next, choose “Create File” and enter the file name “lab_2” then “OK” followed by “Finish”.





10. Then, we need to get a template xdc file that is going to be edited according to the different experiments. Google "basys 3 xdc file" and choose the "xilinix" link that appears (https://www.xilinx.com/support/documentation/university/Vivado-Teaching/HDL-Design/2015x/Basys3/Supporting%20Material/Basys3_Master.xdc). Copy the whole file and paste it into the "lab_2.xdc" that you have just created in the last step. Then uncomment and edit the input Switches and the output LEDs as in the next step.

11. Uncomment (by deleting the # sign) sw[0], sw[1], sw[3],….. led[0], led[1],… lines. Note that each of them has two successive lines (Uncomment both of them). Do the following replacements: sw[0] ➔ A0, sw[1] ➔ A1,……, led[0] ➔ X, led[1] ➔ Y,…. , then Save the file



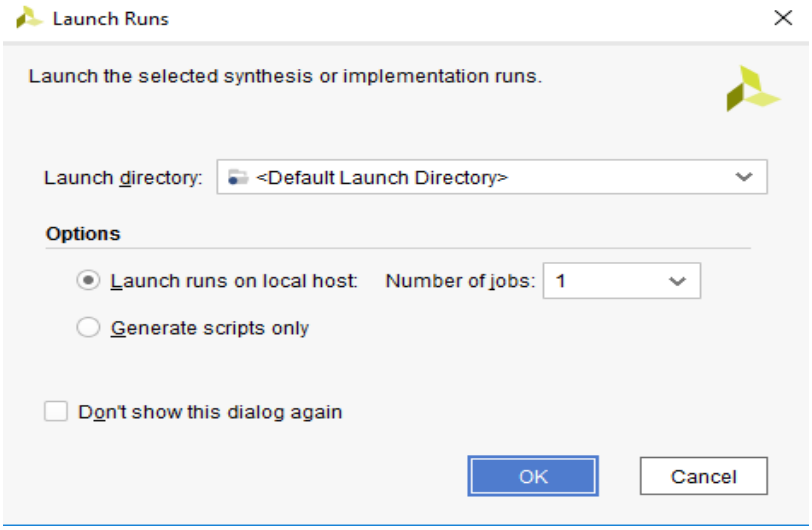
Project Summary | source_1.vhd * | lab_2.xdc

C:/Xilinx/Vivado/2018.2/project/project.srcs/constrs_1/new/lab_2.xdc

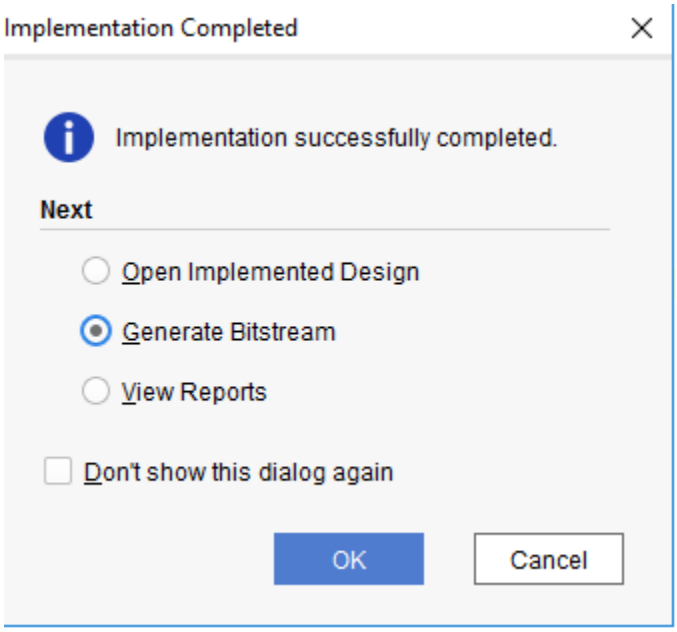
```
#set_property PACKAGE_PIN W5 [get_ports clk]
    #set_property IOSTANDARD LVCMOS33 [get_ports clk]
    #create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]

## Switches
set_property PACKAGE_PIN V17 [get_ports {A0}]
    set_property IOSTANDARD LVCMOS33 [get_ports {A0}]
set_property PACKAGE_PIN V16 [get_ports {A1}]
    set_property IOSTANDARD LVCMOS33 [get_ports {A1}]
set_property PACKAGE_PIN W16 [get_ports {A2}]
    set_property IOSTANDARD LVCMOS33 [get_ports {A2}]
set_property PACKAGE_PIN W17 [get_ports {B0}]
    set_property IOSTANDARD LVCMOS33 [get_ports {B0}]
set_property PACKAGE_PIN W15 [get_ports {B1}]
    set_property IOSTANDARD LVCMOS33 [get_ports {B1}]
set_property PACKAGE_PIN V15 [get_ports {B2}]
    set_property IOSTANDARD LVCMOS33 [get_ports {B2}]
set_property PACKAGE_PIN W14 [get_ports {C}]
    set_property IOSTANDARD LVCMOS33 [get_ports {C}]
#set_property PACKAGE_PIN W13 [get_ports {sw[7]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[7]}]
#set_property PACKAGE_PIN V2 [get_ports {sw[8]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[8]}]
#set_property PACKAGE_PIN T3 [get_ports {sw[9]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[9]}]
#set_property PACKAGE_PIN T2 [get_ports {sw[10]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[10]}]
#set_property PACKAGE_PIN R3 [get_ports {sw[11]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[11]}]
#set_property PACKAGE_PIN W2 [get_ports {sw[12]}]
```

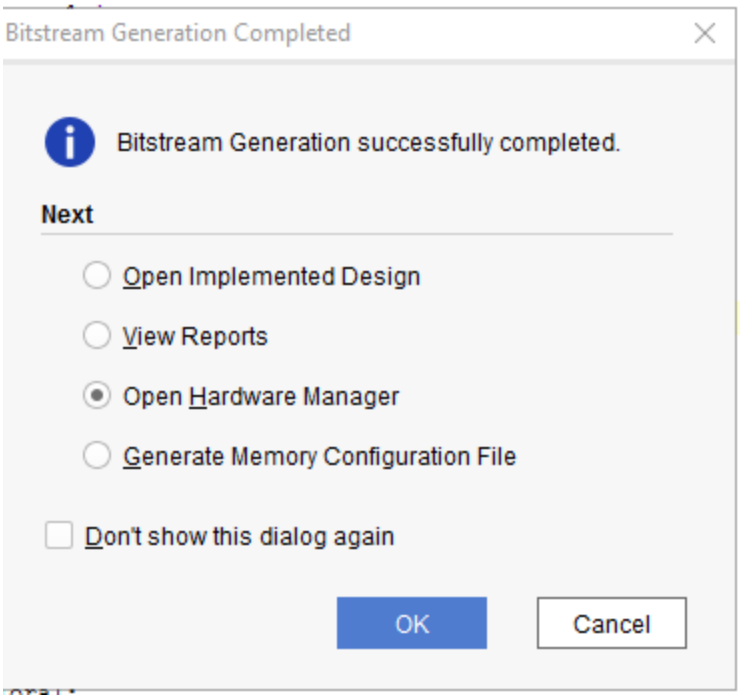
12. From the tool tab choose the play button ▶ and then “Run Implementation”. Select ”Number of jobs” =1 and then press OK.



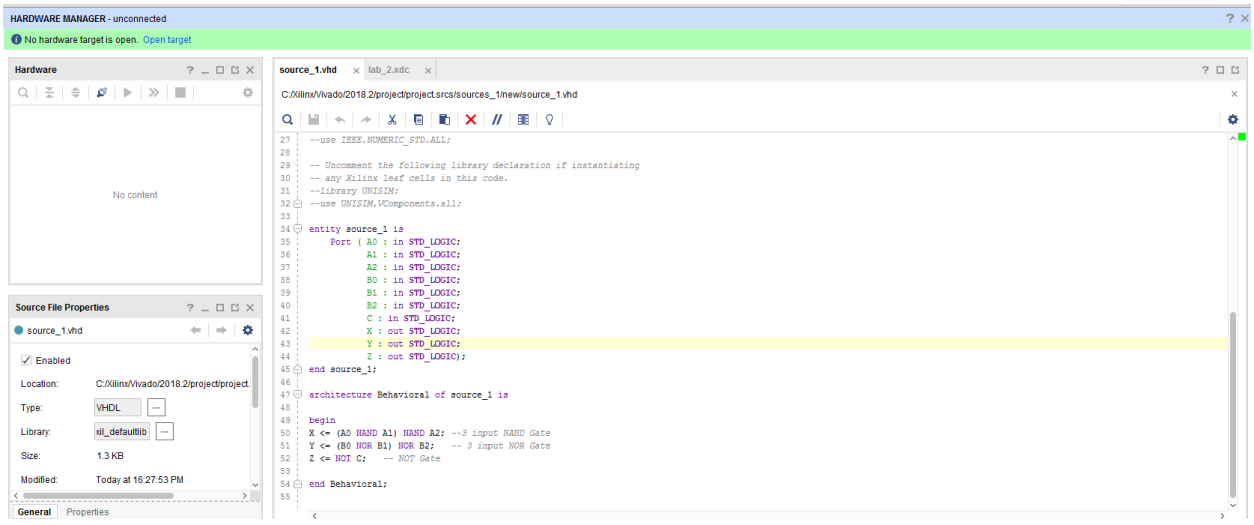
13. The implementation errors window will appear if any or the successfully completed window. From this window select “Generate Bitstream” and then OK. This will make the software generate “.bin” file to be used in programing the hardware BAYAS 3.



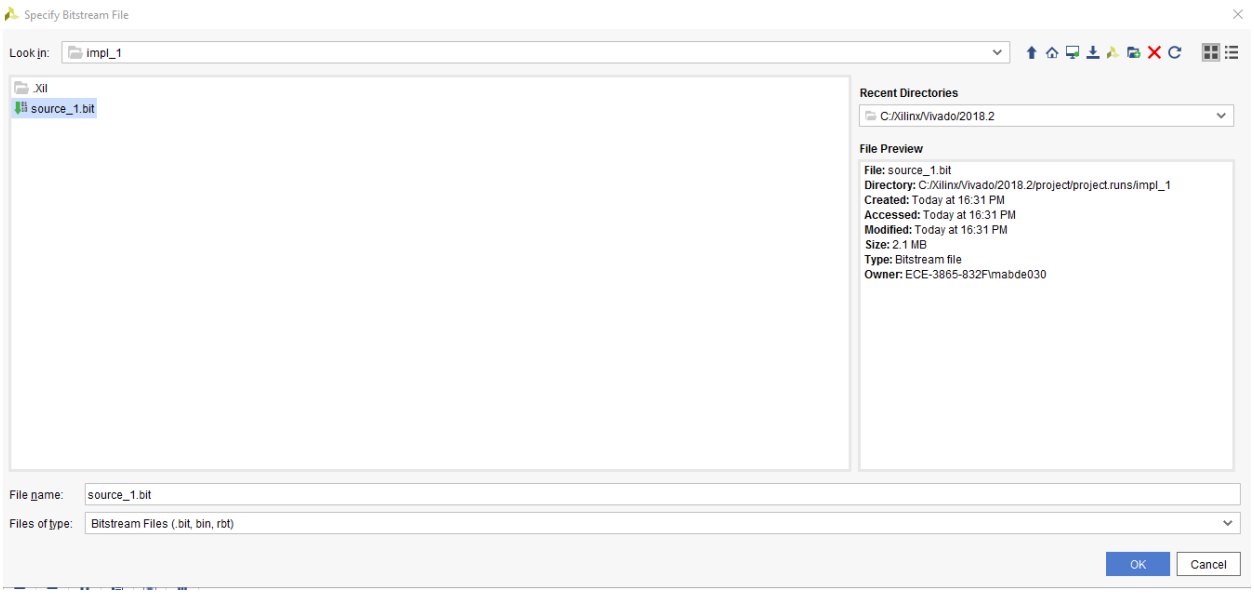
14. The next window will appear in which choose “Open Hardware Manger”, connect the Hardware Kit to the USB port and then press OK.



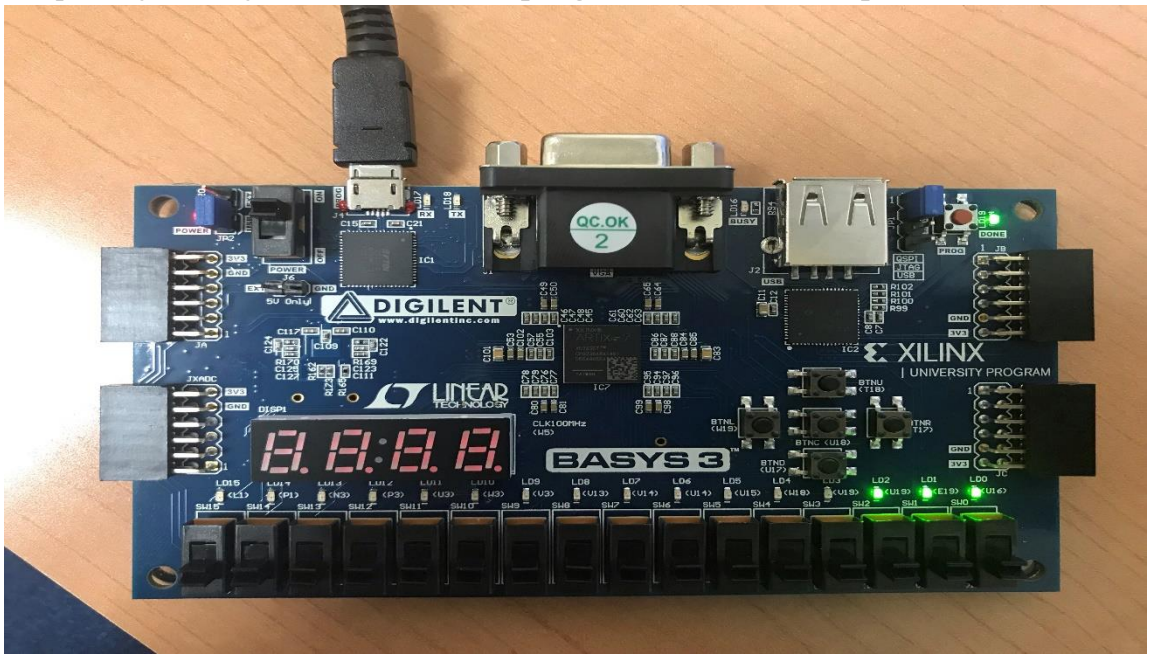
15. A green tab will appear in the top of the Vivado window, from which choose "open target" to program the hardware.



16. From the window appears, select the ".bin" file from the Project you create by browsing for the generated ".bit file" under the ".runs" folder and program the board then press OK.



17. Notice that the 7-segment on the hardware is counting up from 0 to 9 frequently until you download the program and it will stop.



18. Fill in the following truth tables for all the gates by observing the inputs/outputs on the programmed board.

A. NAND Gate

Truth Table (1)

A0	A1	A2	X
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Symbol

Boolean Equation

B. NOR Gate

Truth Table (2)

B0	B1	B2	Y
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Symbol

Boolean Equation

a. NOT Gate

Truth Table (3)

C	Z
0	
1	

Symbol

Boolean Equation

19. Verify that the experimental results are consistent with the Discussion.

Checked by_____ Date _____

Questions:

1. Create a Xilinx project called ***LAB2*** in the same way that you did the projects ***AND_OR3*** and ***INVERT*** In this new project prove the following:

a) A 2-input **NAND** gate is equivalent to a 2-input **AND** gate followed by a **NOT** gate.
b) A 2-input **NOR** gate is equivalent to a 2-input **OR** gate followed by a **NOT** gate.
c) A 2-input **NAND** gate is equivalent to an inverter when the two inputs are tied together.
d) A 2-input **NOR** gate is equivalent to an inverter when one of the inputs is connected to ground.

2. Draw the truth tables in the following to demonstrate your results obtained in the last step. Do they match what you expected?