



EXPERIMENT 4

XOR and XNOR Gates with Applications

OBJECTIVES:

- Examining the characteristics of XOR and XNOR gates.
- Demonstrate applications of XOR and XNOR gates
- Learn to use the VHDL approach to combinational logic design.

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.

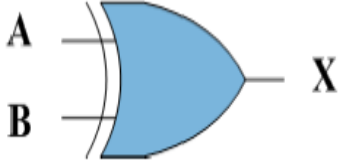
DISCUSSION:

So far we have studied five basic types of gates: AND, OR, NAND, NOR and NOT. In some applications, it is convenient to use two other types of gates: XOR and XNOR. The XOR and XNOR gates have their own symbols and unique characteristics. Common applications for XOR and XNOR gates are: comparators, switchable inverter/buffers, parity generator/checkers and adder/subtractor. They can also be used to simplify Boolean equations.

We will first discuss the properties of XOR and XNOR having two inputs.

Gate Characteristics:

1. The XOR Gate

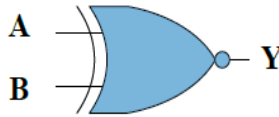
Symbol	Boolean Equation	Truth Table		
	$X = A'B + AB'$ $X = A \oplus B$	Input		Output
		A	B	X
		0	0	0
		0	1	1
		1	0	1
		1	1	0

For a 2-input XOR gate, the output is High when the inputs are unequal. The output is Low when the inputs are equal. The Boolean equation for a 2-input XOR gate can be abbreviated as:

$$X = A \oplus B$$

However, the function definition remains the same.

2. The XNOR Gate

Symbol	Boolean Equation	Truth Table																		
	$Y = A'B' + AB$	<table border="1"> <thead> <tr> <th colspan="2">Input</th> <th>Output</th> </tr> <tr> <th>A</th> <th>B</th> <th>Y</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	Input		Output	A	B	Y	0	0	1	0	1	0	1	0	0	1	1	1
Input		Output																		
A	B	Y																		
0	0	1																		
0	1	0																		
1	0	0																		
1	1	1																		

The output of an XNOR gate is the complement of that of a XOR. For a 2-input XNOR gate, the output is Low when the inputs are unequal but High when the inputs are equal. The Boolean equation for a 2-input XNOR gate can be written as:

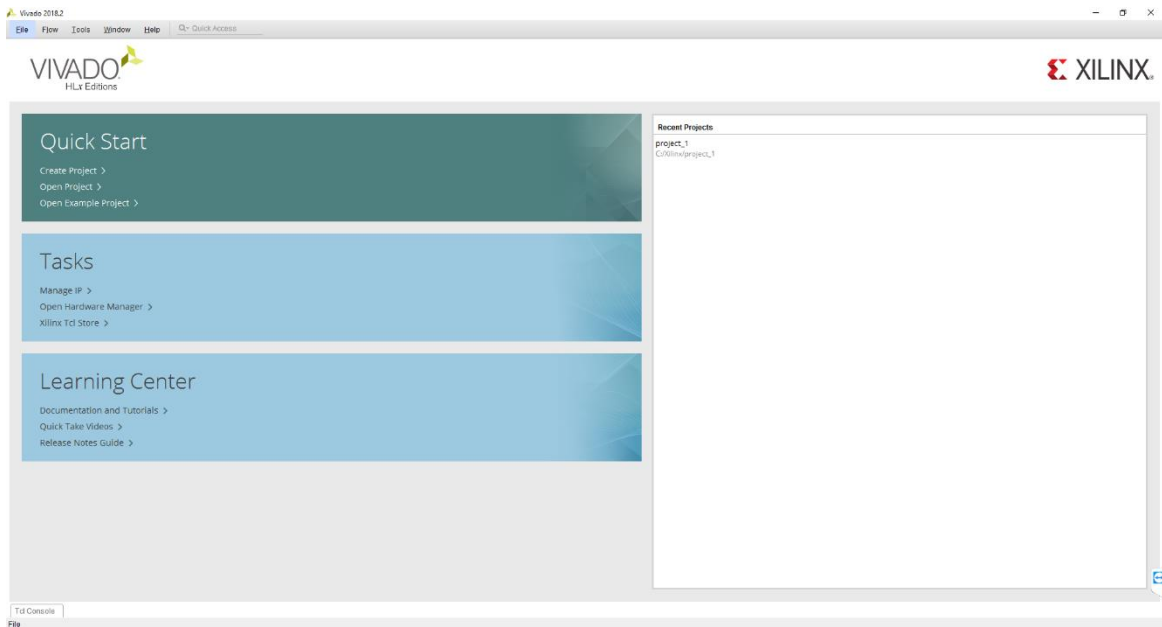
$$Y = \overline{A \oplus B}$$

The number of inputs for the XOR and XNOR gates can be two or more. The characteristics of XOR and XNOR gates can be extended to three or more inputs. We will examine the characteristics of 3-input XOR and XNOR gates.

PROCEDURE:

Section 1 XOR and XNOR characteristics:

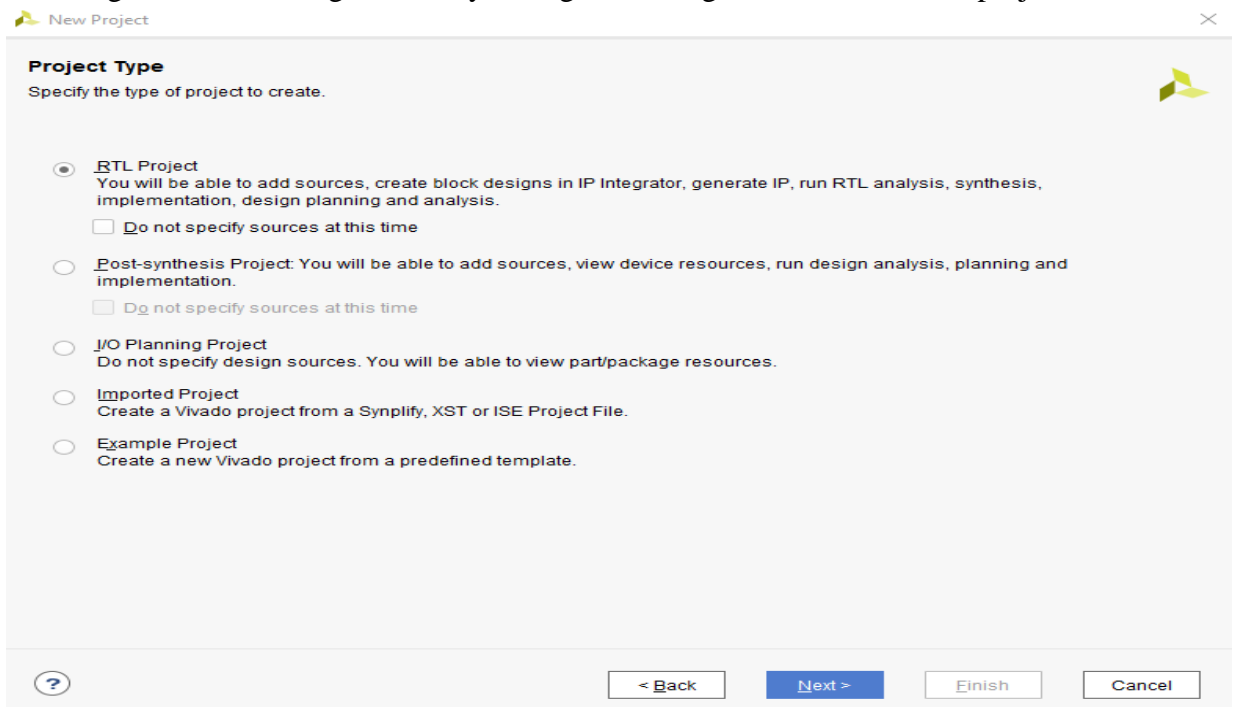
1. Open Xilinx Vivado.



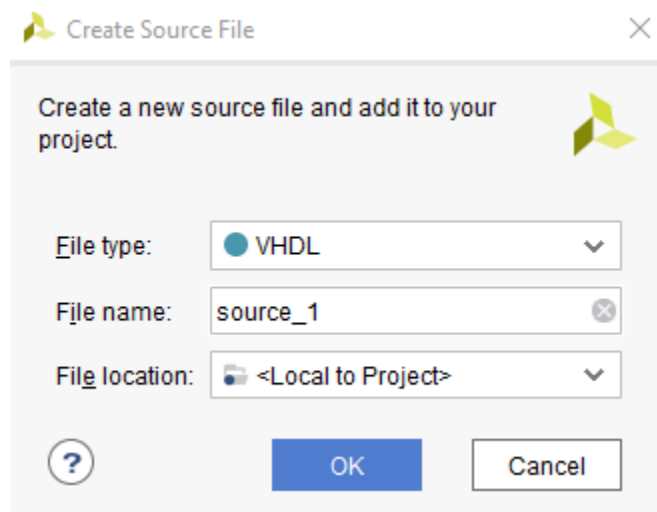
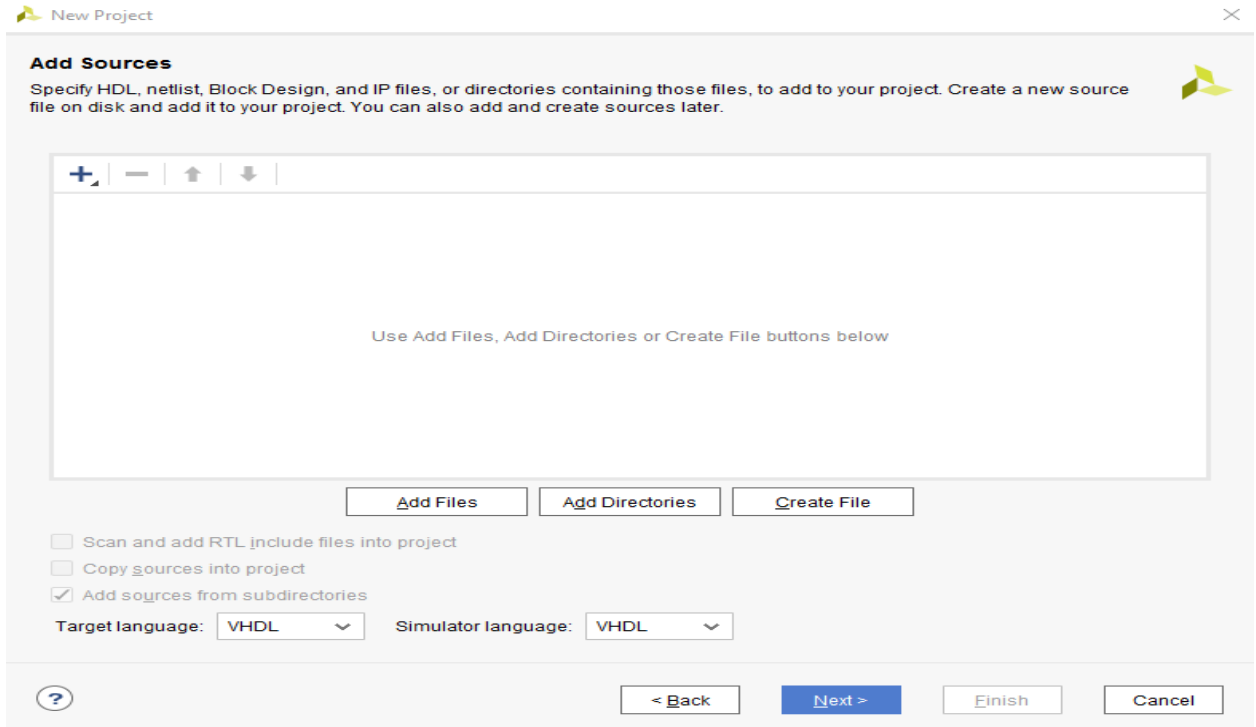
2. In the **Xilinx-Project Navigator** window, Quick start, **New Project**.

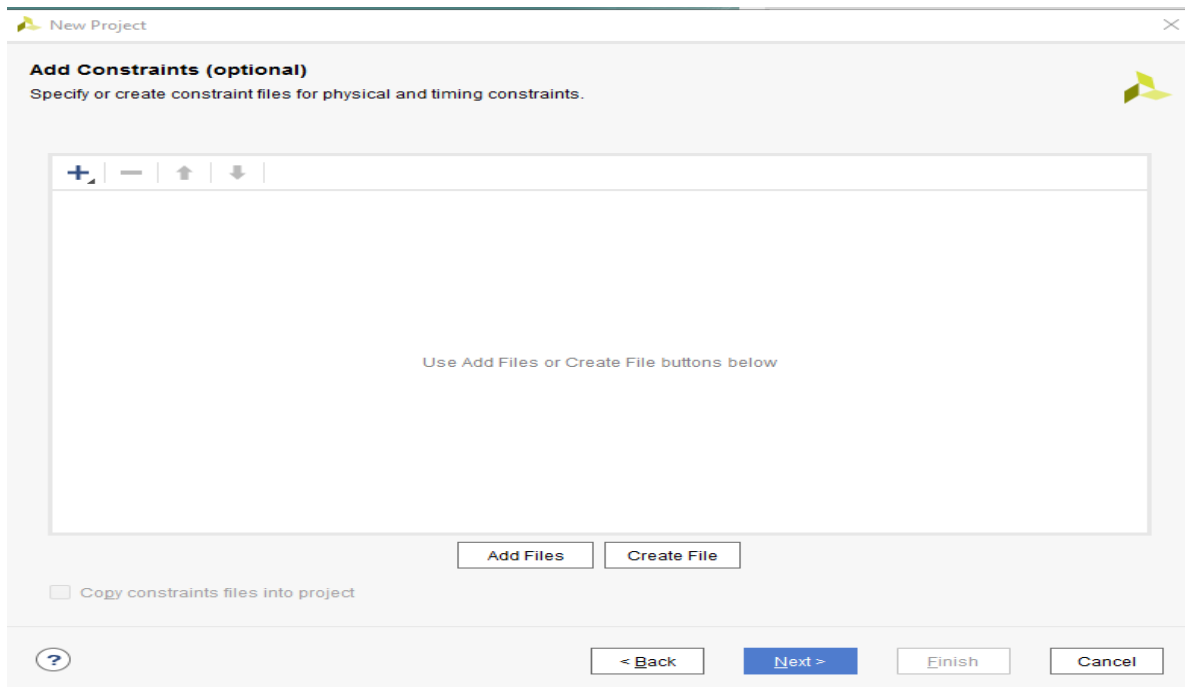
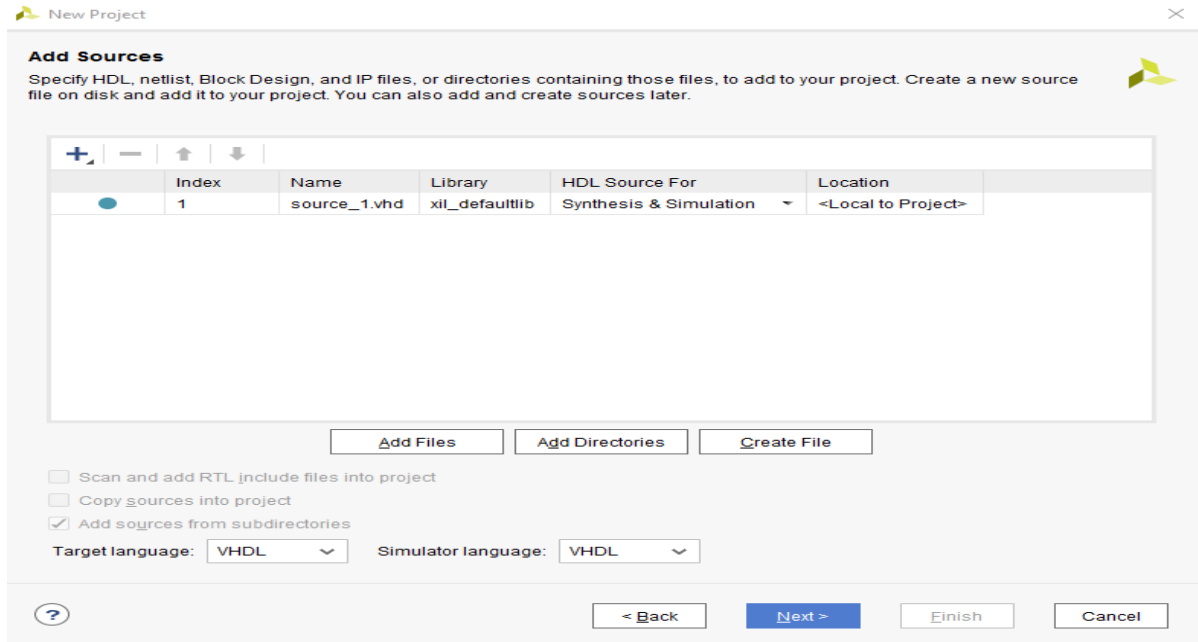
3. Name the project.

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.

New Project

Default Part

Choose a default Xilinx part or board for your project. This can be changed later.

Parts | Boards

[Reset All Filters](#)


Category: Package: Temperature:

Family: Speed:

Search: (1 match)


Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DSPs	Gate
xc7a35tcpg236-1L	236	106	20800	41600	50	0	90	2

New Project



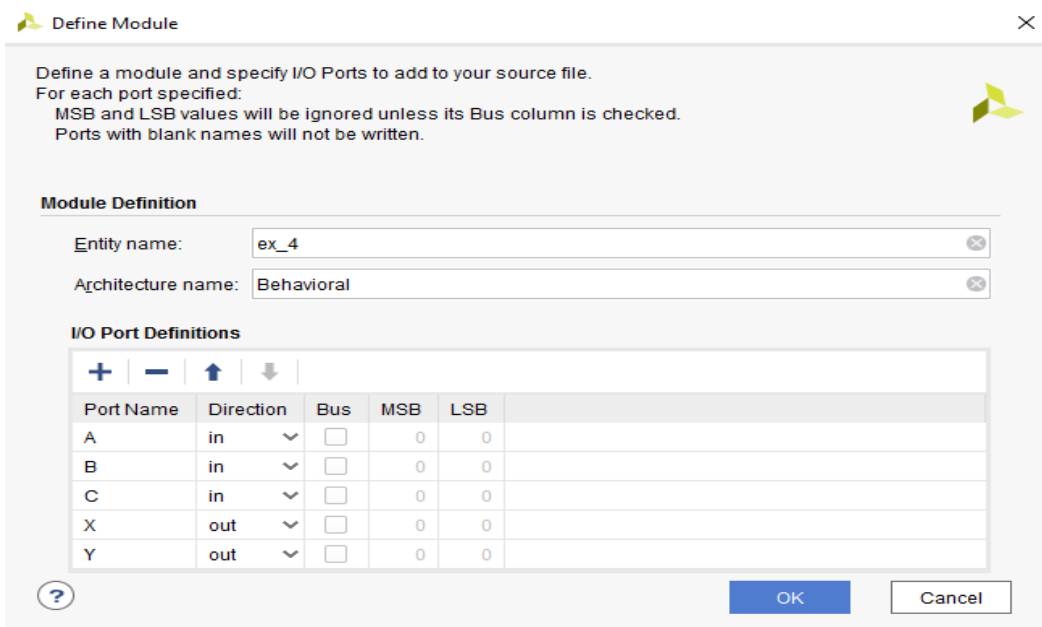
New Project Summary

- A new RTL project named 'project' will be created.
- 1 source file will be added.
- No constraints files will be added. Use Add Sources to add them later.
- The default part and product family for the new project:
Default Part: xc7a35tcpg236-1L
Product: Artix-7
Family: Artix-7
Package: cpg236
Speed Grade: -1L



To create the project, click Finish

- The Define Module Window that will appear, we will choose the input and output labels for the gates under investigation in this experiment.

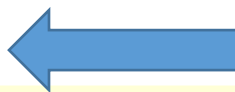


- In the “source_1.vhd” created file, type the gates equivalent VHDL code for the XOR and XNOR gates between the “begin” and “end Behavioral” as follows and then save the file.

```

22  library IEEE;
23  use IEEE.STD_LOGIC_1164.ALL;
24
25  -- Uncomment the following library declaration if using
26  -- arithmetic functions with Signed or Unsigned values
27  --use IEEE.NUMERIC_STD.ALL;
28
29  -- Uncomment the following library declaration if instantiating
30  -- any Xilinx leaf cells in this code.
31  --library UNISIM;
32  --use UNISIM.VComponents.all;
33
34  entity ex_4 is
35      Port ( A : in STD_LOGIC;
36            B : in STD_LOGIC;
37            C : in STD_LOGIC;
38            X : out STD_LOGIC;
39            Y : out STD_LOGIC);
40  end ex_4;
41
42  architecture Behavioral of ex_4 is
43
44  begin
45  X<= A xor B xor C;
46  Y<= not(A XOR B XOR C);
47
48  end Behavioral;
49

```




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 "xilinx" 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.

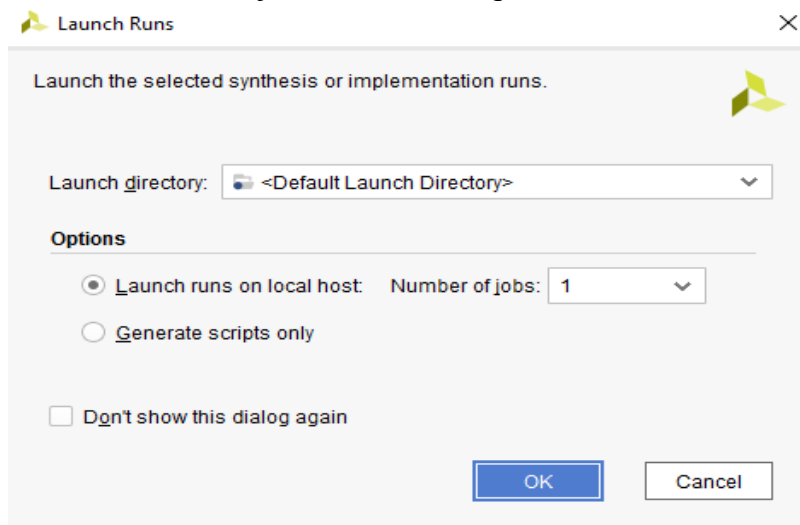
```

11 | ## Switches
12 | set_property PACKAGE_PIN V17 [get_ports {A}]
13 |     set_property IOSTANDARD LVCMOS33 [get_ports {A}]
14 | set_property PACKAGE_PIN V16 [get_ports {B}]
15 |     set_property IOSTANDARD LVCMOS33 [get_ports {B}]
16 | set_property PACKAGE_PIN W16 [get_ports {C}]
17 |     set_property IOSTANDARD LVCMOS33 [get_ports {C}]
...
46 | ## LEDs
47 | set_property PACKAGE_PIN U16 [get_ports {X}]
48 |     set_property IOSTANDARD LVCMOS33 [get_ports {X}]
49 | set_property PACKAGE_PIN E19 [get_ports {Y}]
50 |     set_property IOSTANDARD LVCMOS33 [get_ports {Y}]

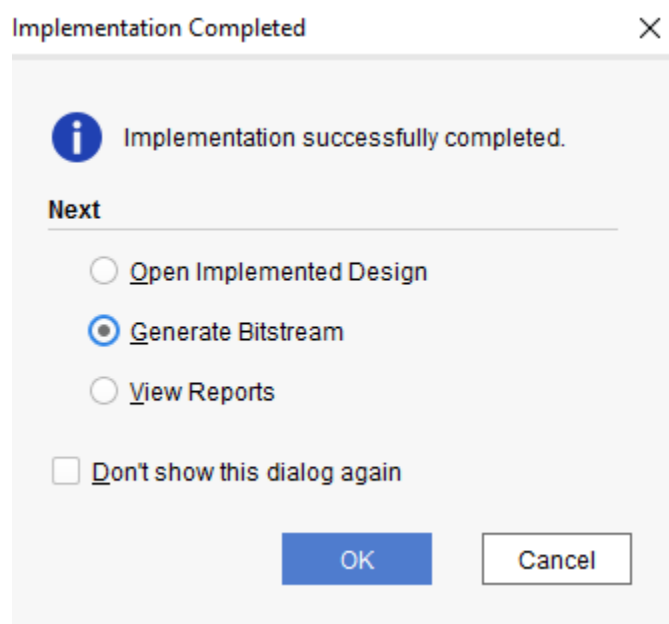
```



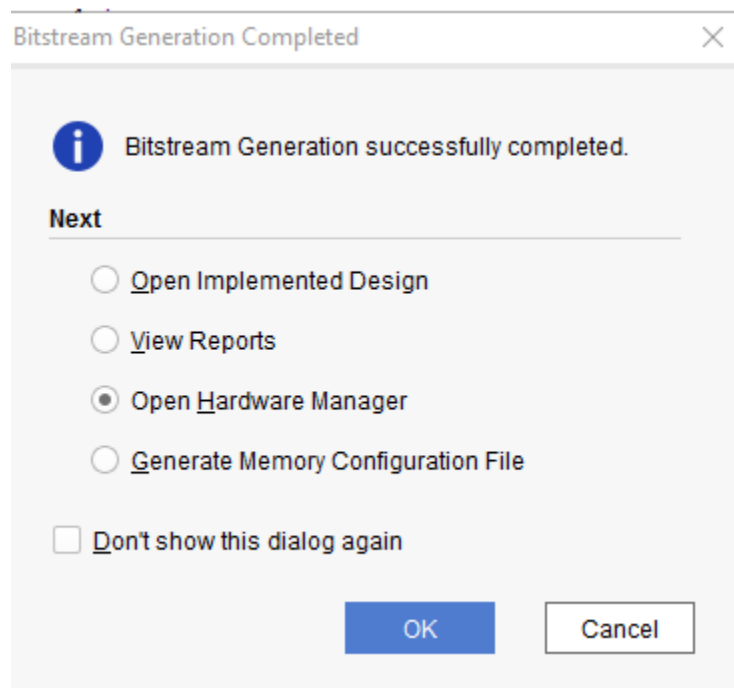
11. From the tool tab choose the play button  and then "Run Implementation". Select "Number of jobs" =1 and then press OK.



12. 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.



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



14. A green tab will appear in the top of the Vivado window, from which choose “open target” to program the hardware.
15. 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.
16. Fill in the following truth tables for all the gates by observing the inputs/outputs on the programmed board.

A. XOR Gate

Truth Table (1)

A	B	C	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. XNOR Gate

Truth Table (2)

A	B	C	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

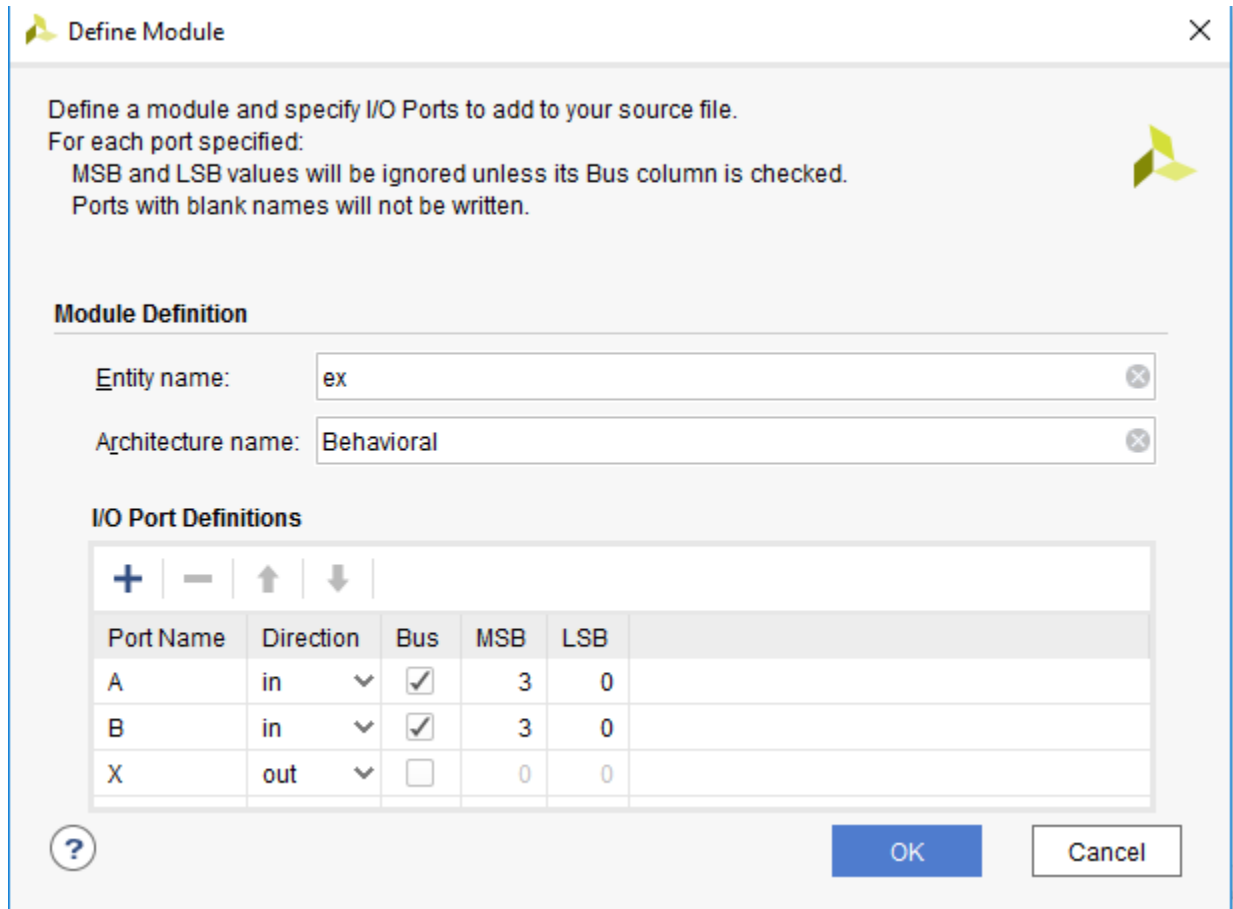
Boolean Equation

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

Checked by _____ Date _____

Section 2 XOR gates used in a comparator:

1. Repeat section 1 from step 1 to 6.
2. The Define Module Window that will appear, we will choose the input and output labels for the gates under investigation in this experiment.



Define a module and specify I/O Ports to add to your source file.
For each port specified:
MSB and LSB values will be ignored unless its Bus column is checked.
Ports with blank names will not be written.

Module Definition

Entity name:

Architecture name:

I/O Port Definitions

Port Name	Direction	Bus	MSB	LSB
A	in	<input checked="" type="checkbox"/>	3	0
B	in	<input checked="" type="checkbox"/>	3	0
X	out	<input type="checkbox"/>	0	0

Buttons: ? OK Cancel

3. In the “source_1.vhd” created file, type the gates equivalent VHDL code for the XOR and XNOR gates between the “begin” and “end Behavioral” as follows and then save the file.

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;


-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity ex is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
          B : in STD_LOGIC_VECTOR (3 downto 0);
          X : out STD_LOGIC);
end ex;

architecture Behavioral of ex is
begin
    X <= NOT ((A(0) XOR B(0)) OR (A(1) XOR B(1)) OR (A(2) XOR B(2)) OR (A(3) XOR B(3)) );
end Behavioral;

```




- 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".

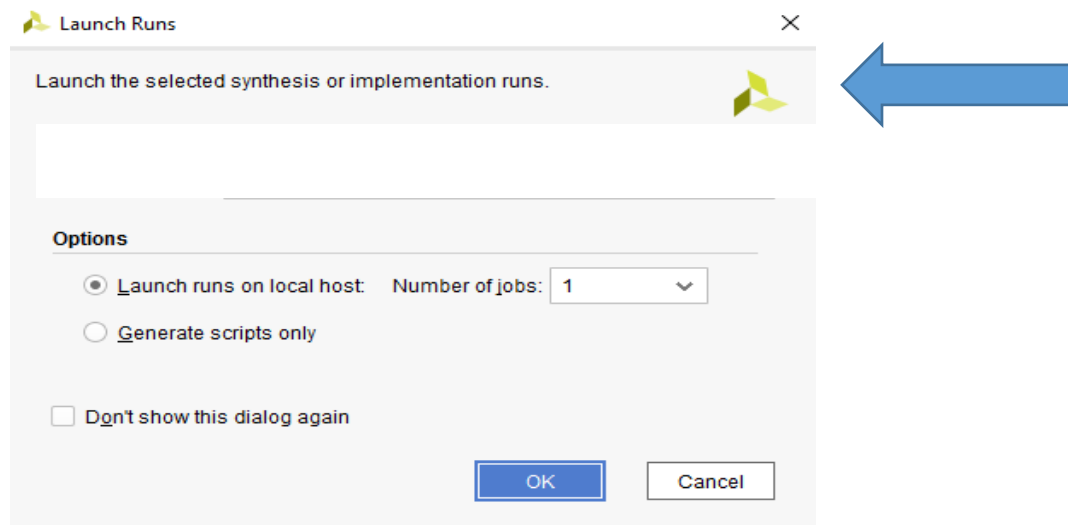
```

11 ## Switches
12 set_property PACKAGE_PIN V17 [get_ports {A[0]}]
13     set_property IOSTANDARD LVCMOS33 [get_ports {A[0]}]
14 set_property PACKAGE_PIN V16 [get_ports {A[1]}]
15     set_property IOSTANDARD LVCMOS33 [get_ports {A[1]}]
16 set_property PACKAGE_PIN W16 [get_ports {A[2]}]
17     set_property IOSTANDARD LVCMOS33 [get_ports {A[2]}]
18 set_property PACKAGE_PIN W17 [get_ports {A[3]}]
19     set_property IOSTANDARD LVCMOS33 [get_ports {A[3]}]
20 set_property PACKAGE_PIN W15 [get_ports {B[0]}]
21     set_property IOSTANDARD LVCMOS33 [get_ports {B[0]}]
22 set_property PACKAGE_PIN V15 [get_ports {B[1]}]
23     set_property IOSTANDARD LVCMOS33 [get_ports {B[1]}]
24 set_property PACKAGE_PIN W14 [get_ports {B[2]}]
25     set_property IOSTANDARD LVCMOS33 [get_ports {B[2]}]
26 set_property PACKAGE_PIN W13 [get_ports {B[3]}]
27     set_property IOSTANDARD LVCMOS33 [get_ports {B[3]}]

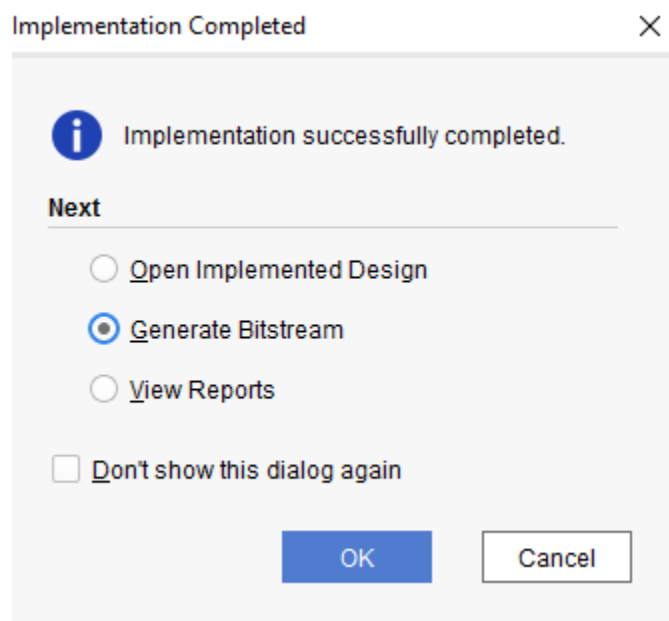
46 : ## LEDs
47 : set_property PACKAGE_PIN U16 [get_ports {X}]
48 :     set_property IOSTANDARD LVCMOS33 [get_ports {X}]

```

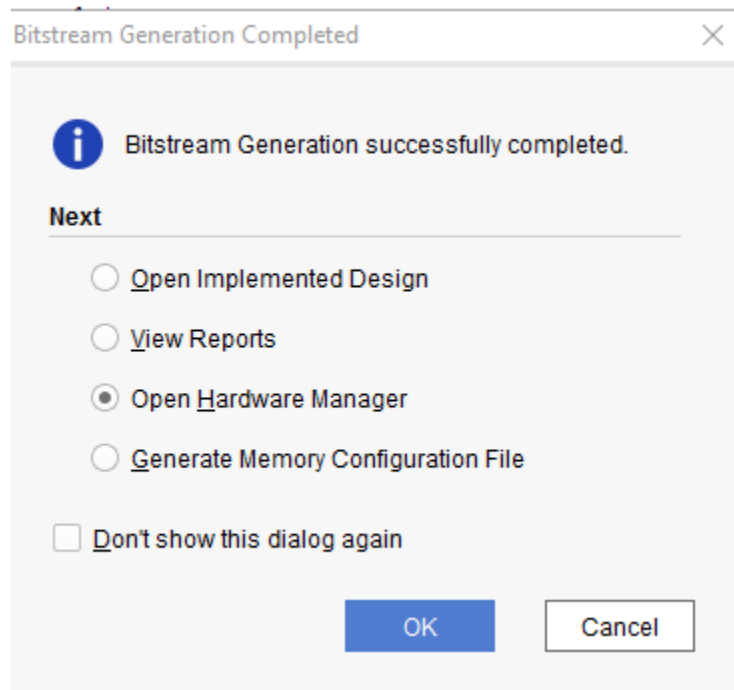
- Then, we need to get a template xdc file that is going to be edited according to the different experiments. Google “basys 3 xdc file” and choose the “xilinx” link that appears (https://www.xilinx.com/support/documentation/university/Vivado-Teaching/HDL-Design/2015x/Basys3/Supporting%20Material/Basys3_Master.xdc). 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.
- From the tool tab choose the play button  and then “Run Implementation”. Select “Number of jobs” =1 and then press OK.



- 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 programming the hardware BAYAS 3.



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



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

10. 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.
11. Fill in the following truth tables for all the gates by observing the inputs/outputs on the programmed board.

Word A				Word B				Output
A0	A1	A2	A3	B0	B1	B2	B3	X
0	1	1	0	1	0	0	1	
1	0	1	0	1	0	1	0	
0	0	1	1	1	1	0	0	
0	1	0	0	0	1	0	1	
1	1	0	1	1	1	0	1	
0	0	0	0	0	0	0	0	
0	1	1	1	0	1	1	1	
1	1	1	1	1	1	1	0	
1	0	1	1	0	0	1	1	

Truth Table

12. Summarize the results on your own words.

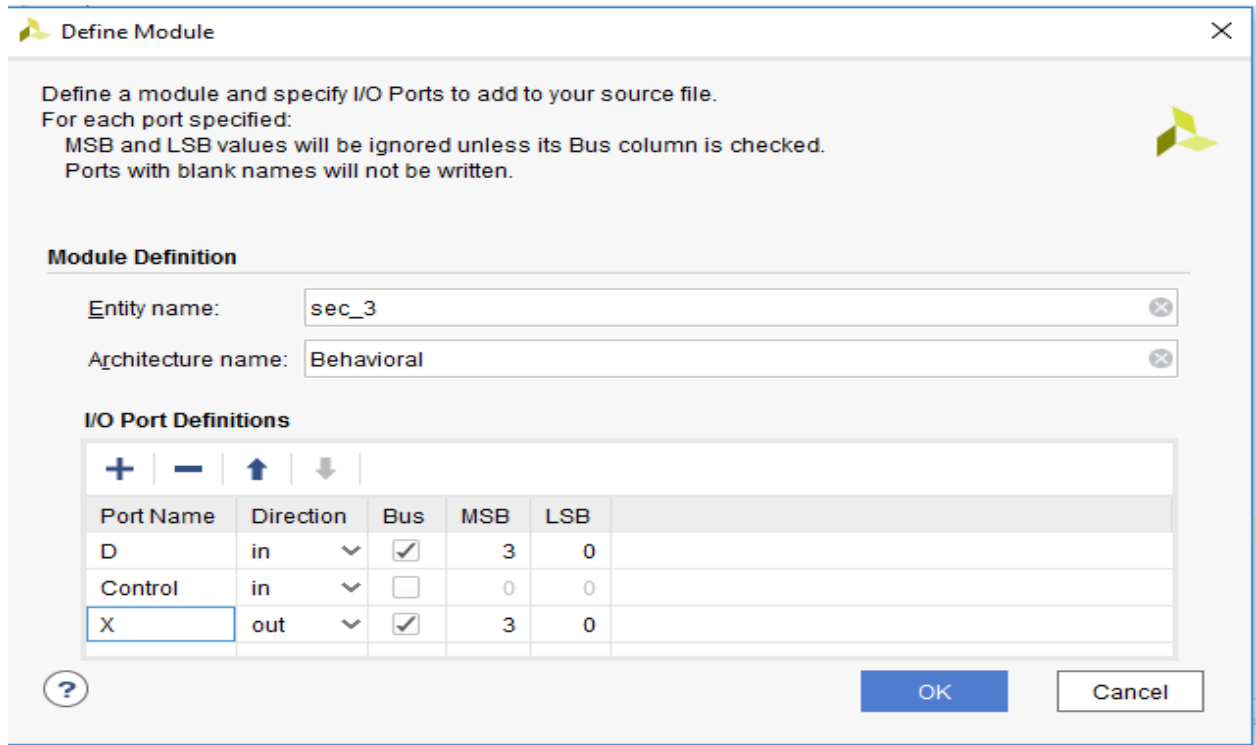
Checked by _____ Date _____

Section 3: XNOR gates used as buffers and inverters.

If you examine the truth table of an XNOR gate carefully, you will notice an interesting fact: when input A is held Low, the output is the complement of input B. When input A is kept High, the output follows input B. This effect means that the XNOR gate can be used to construct a buffer/inverter circuit. What we have to do is use one of the inputs as the control signal and the other input as the data signal. The XNOR will act like a buffer when the control signal is high, but as an inverter when the control signal is pulled Low. Here we will build a 4-bit buffer/inverter circuit and then run a simulation to verify the result.

1. Repeat section 1 from step 1 to 6.

2. The Define Module Window that will appear, we will choose the input and output labels for the gates under investigation in this experiment.



Define a module and specify I/O Ports to add to your source file.
For each port specified:
MSB and LSB values will be ignored unless its Bus column is checked.
Ports with blank names will not be written.

Module Definition

Entity name:

Architecture name:

I/O Port Definitions

Port Name	Direction	Bus	MSB	LSB
D	in	<input checked="" type="checkbox"/>	3	0
Control	in	<input type="checkbox"/>	0	0
X	out	<input checked="" type="checkbox"/>	3	0


Buttons: ? OK Cancel

3. In the “source_1.vhd” created file, type the gates equivalent VHDL code for the gates between the “begin” and “end Behavioral” as follows and then save the file.

```

22 | library IEEE;
23 | use IEEE.STD_LOGIC_1164.ALL;
24 |
25 | -- Uncomment the following library declaration if using
26 | -- arithmetic functions with Signed or Unsigned values
27 | --use IEEE.NUMERIC_STD.ALL;
28 |
29 | -- Uncomment the following library declaration if instantiating
30 | -- any Xilinx leaf cells in this code.
31 | --library UNISIM;
32 | --use UNISIM.VComponents.all;
33 |
34 | entity sec_3 is
35 |     Port ( D : in STD_LOGIC_VECTOR (3 downto 0);
36 |           Control : in STD_LOGIC;
37 |           X : out STD_LOGIC_VECTOR (3 downto 0));
38 | end sec_3;
39 |
40 | architecture Behavioral of sec_3 is
41 |
42 |     begin
43 | X(0)<= Control XNOR D(0);
44 | X(1)<= Control XNOR D(1);
45 | X(2)<= Control XNOR D(2);
46 | X(3)<= Control XNOR D(3);
47 | end Behavioral;
48 |

```



4. 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".
5. Repeat section 1 from step 10 to 15.
6. Fill in the following truth tables for all the gates by observing the inputs/outputs on the programmed board.

Truth Table

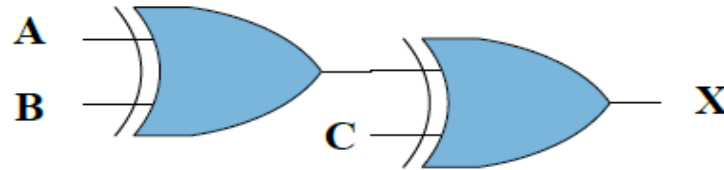
Inputs				Outputs								
				Control=0				Control=1				
D0	D1	D2	D3	X0	X1	X2	X3	X0	X1	X2	X3	

7. Summarize the results on your own words.

Checked by _____ Date _____

Questions:

1.) A 3-input XOR gate is equivalent to the circuit shown below: $ABCX$



The Boolean equation can be written as:

$$X = (A' B + AB')'C + (A' B + AB')C'$$

Or it simply denoted as:

$$X = A \oplus B \oplus C$$

Using only AND, OR and inverter gates to implement the above Boolean equation, how many gates are needed? Draw the logic diagram. Compare the savings of a single XOR gate implementation with the circuit you just drew.

2.) How can you use a 2-input XOR gate to function as a 1-bit buffer/inverter? Draw the logic diagram. Show the logic connections for the control and data input lines.