12-2015

Logic Gates Using the Digilent Basys3

Austin H. Duncan
East Tennessee State University

Logic Gates Using the Digilent Basys3

Thesis submitted in partial fulfillment of Honors

By

Austin Duncan
The Honors College
East Tennessee State University

December 2, 2015

Austin Duncan, Author

Hugh Blanton, Faculty Mentor

Paul Sims, Faculty Reader

Karen Kornweibel, Faculty Reader

Table of Contents

Introduction…………………………………………………………………………………1
Methods……………………………………………………………………………………...2
Results………………………………………………………………………………………….3
Discussion…………………………………………………………………………………...4
References…………………………………………………………………………………...5
Acknowledgements………………………………………………………………………5
Appendix A…………………………………………………………………………………….6
Appendix B…………………………………………………………………………………...14
Appendix C…………………………………………………………………………………...22
Appendix D…………………………………………………………………………………...30

Introduction

In the field of electronics few things are as important as logic gates. Logic gates form the basis of various circuits, devices, and techniques that are important to understanding electrical principles. Currently, logic gates are taught at ETSU with an analog board and physical chips. These devices can be difficult to understand. One solution is replacing the analog devices with a digital device. With the acquisition of the Digilent Basys3 this became a reality.

The Digilent Basys3 was designed to be an entry level Field Programmable Gate Array (FPGA) board. The board can be programmed to model many different things. This is accomplished by utilizing the various switches, LEDS, pushbuttons, and 7-segment displays that are built onto the board. The Basys3 was specifically built to interface with the Vivado Design Suite, which is regarded as one of the highest quality design tools used by modern engineers (Basys™3 Artix-7 FPGA Board, n.d.).

I was tasked with designing three basic lab activities utilizing the Basys3 board. The activities were to model three basic logic gates: AND, OR, and NAND. These three gates are the basis of digital circuitry. It is important to start with the basics when teaching something as complicated as digital circuits. However, these labs barely scratch the surface as to what the Basys3 is capable of. The hope is that labs will continue to be developed using this board and that more complicated designs will be implemented.

In recent years medical technology has moved away from analog devices in favor of smaller, simpler, digital designs. This is due to the small size and programmability that is inherent with digital devices. According to RTC Magazine, digital devices have greatly improved healthcare. Advances in digital microcontrollers allow medical devices to be smaller, require less power, have fewer parts, and are just as powerful as their analog counterparts

(Sankman, 2010). Because digital devices have become so prominent within healthcare, it is important that Biomedical Engineering Technology are exposed to them. The Basys3 is a perfect introduction to digital devices.

Methods

I began by researching how to program the Basys3. This was difficult at first because the Basys3 is a relatively new product. Over time more information was published by people on the Digilent forums (Digilent Forum, n.d.). This is where I started learning the basics of programming the Basys3. The forums provided little help to me because they were discussing more complex programs than the ones I was attempting to create. The most relevant information I was able to find was in YouTube videos. Digilent produced an introductory tutorial video which I watched. This showed me the basic steps needed to program the Basys3 with a preexisting file (Diglent, Inc, 2014). However, the video failed to show the process of how to write an original program. Another useful YouTube video was one by a man named Andrew Danowitz, his video showed me how to correctly indicate the inputs in Vivado as well as how to correctly write the program (Danowitz, 2015). After viewing these videos I was able to develop my own programs for the Basys3. The programs I developed modeled the basic AND, OR, and NAND logic gates on the Basys3.

After the programs were developed I wrote lab activities to be used in the ENTC 3370 Digital Circuits class. These lab activities will more than likely be the students' first exposure to digital circuits. The labs were designed to be easy to follow and reproduce. In order to better illustrate the steps that had to be followed screen shots were used. If the lab activities are followed exactly the student should be able to model the gates with ease. The step by step nature of the labs make it simple to return to a previous step if needed.

I based the lab activities on a lab sample provided by my thesis mentor, Dr. Hugh Blanton. This sample can be viewed in Appendix D. I used the basic step by step structure of the lab activity and applied the steps used to program the Basys3. These labs use an experimental learning approach to education. By performing the lab the student is more inclined to learn the material than if that student simply read material (Teaching Strategies, n.d.). I believe that experimentation is a very effective learning tool in the field of Engineering Technology. In my personal experience I did not truly grasp the concepts I was taught in class until I performed a lab activity such as the labs developed. This is why the lab activities were designed in this way.

Results

After the programs were developed I wrote lab activities to be used in the ENTC 3370 Digital Circuits class. The goal of this class is to introduce students to the concepts of digital circuits. Because of this, these lab activities will more than likely be the students' first exposure to digital circuits. The labs were designed to be easy to follow and reproduce. In order to better illustrate the steps that had to be followed pictures were used. If the lab activities are followed exactly the student should be able to model the gates with ease. The step by step nature of the labs make it simple to return to a previous step if needed.

I asked one of my classmates to test the lab activities to see if they would be successful in a classroom setting. He was able to reproduce the desired results after following the procedures. However, he did suggest that I rephrase some of the definitions of the gates. The terms I had used previously led to confusion. After this test I updated the definitions of the gates to be more easily understood. The lab activities can be viewed in the appendices.

Discussion

Various problems were encountered during this process. A lack of information was one of the hardest things to overcome. Because the Basys3 had been recently released when I started my research it was difficult to find basic information about programming it. This led to a lot of trial and error in order to find a solution. I also had trouble determining what computer programming language the program needed to be written in. Vivado can produce files with either the Verilog or VHDL languages. For this project I used VHDL. The commands AND, OR, and NAND are preprogrammed keywords within the language so writing the code for the logic gates was a simple process. I discovered near the end of this process that I had been selecting the wrong part number. In order to program the device the specific part number of the FPGA has to be entered into Vivado. This made it impossible for the program to be loaded onto the Basys3. After this was corrected the project was able to be finalized.

In a classroom setting the labs require very little previous knowledge of logic gates to perform. However, Vivado can be difficult to operate for beginners. The labs are designed to be a good introduction to programming within Vivado. Previous labs used Protoboards to model the logic gates. Although the Protoboard is faster and more user friendly than the Basys3, the Basys3 is far less limited in its possible applications. Because of this, moving from Protoboards to the Basys3 would allow more complicated lab activities to be designed.

References

Basys™3 Artix-7 FPGA Board. (n.d.). Retrieved from Digilent: https://www.digilentinc.com/Products/Detail.cfm?NavPath=2,400,1288&Prod=BASYS3

Danowitz, A. (2015, July 9). *vivado and basys3 getting started*. Retrieved from Youtube: https://www.youtube.com/watch?v=nV-Dn9cw1R8

Digilent Forum. (n.d.). Retrieved from https://forum.digilentinc.com/

Diglent, Inc. (2014, September 18). *Getting started with Vivado and Basys3*. Retrieved from Youtube: https://www.youtube.com/watch?v=6_GxkslqbcU

Sankman, J. (2010, September). *Transitioning From Analog to Digital in Medical Designs*. Retrieved from RTC Magazine: http://www.rtcmagazine.com/articles/view/101784

Teaching Strategies. (n.d.). Retrieved from Merlot Pedagogy: http://pedagogy.merlot.org/TeachingStrategies.html

Acknowledgements

Dr. Hugh Blanton

Dr. Karen Kornweibel

Dr. Paul Sims

Caleb Taber

Appendix A

Lab 1

ENTC 3370

AND Gate

Objective:

To familiarize the student with the Basys3 board and to introduce him or her to basic logic gates.

Parts and Equipment:

- Basys3 board
- USB cable
- Computer
- Vivado Design Suite

Introduction:

In this lab the Basys3 FPGA board will be used to model a basic AND gate. For an AND gate the output should be 0 if any input is 0 and 1 only if both inputs are 1. The inputs in this lab will be represented by switches 0 and 1 on the Basys3. The output will be represented by LED 0.

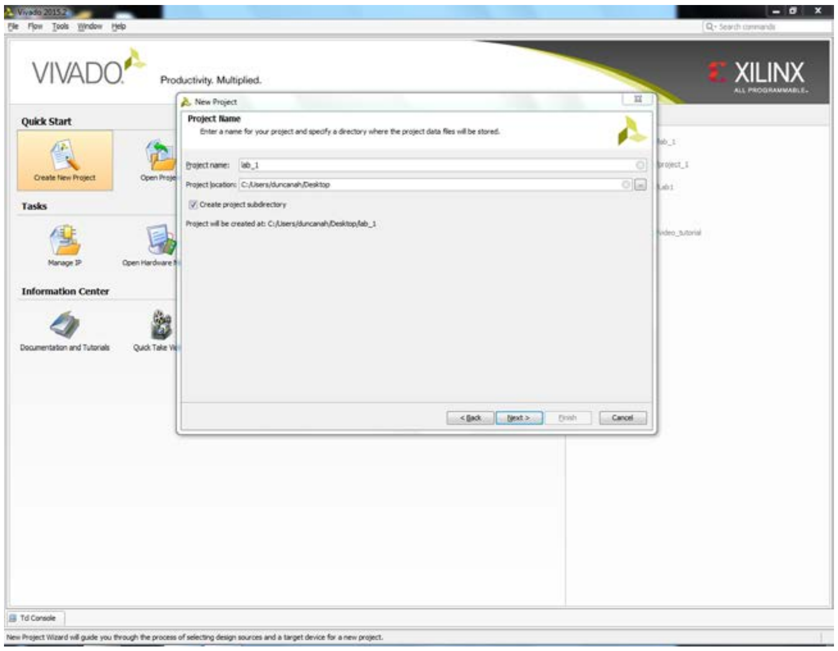
Procedure:

First the Basys3 must be configured in the correct way. The jumper in the top left corner has to be set to the USB setting. This allows the Basys3 to be powered using USB. The top right jumper must be set to the JTAG setting. This is the way the program will be loaded onto the board. Make sure the included USB cable is connected to the board and also to the computer.

Then, load Vivado and click new project.

A pop up will appear on the screen, click next. Name the file lab_1 and click next. Make sure to save the file in a place where it can be accessed later, such as the Z drive.

Figure 1: Naming the Project



Project Name
Enter a name for your project and specify a directory where the project data files will be stored.

Project name: lab_1

Project location: C:/Users/duncanah/Desktop

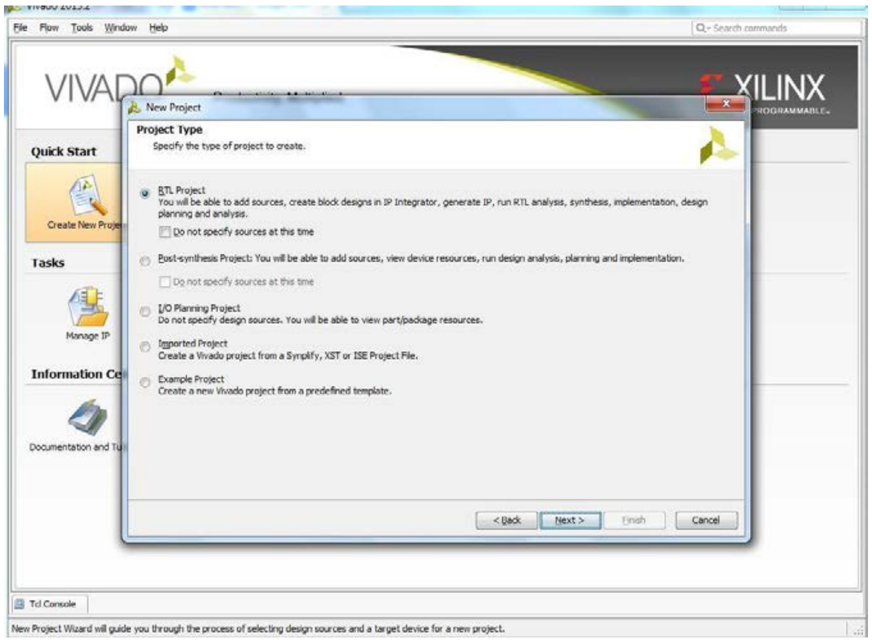
[✓] Create project subdirectory

Project will be created at: C:/Users/duncanah/Desktop/lab_1

New Project Wizard will guide you through the process of selecting design sources and a target device for a new project.

Next, specify the type of project that will be made. Select the RTL Project option as illustrated in Figure 2. RTL stands for register-transfer level.

Figure 2: Project Type



Project Type
Specify the type of project to create.

● RTL Project
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.
☐ Do not specify sources at this time

○ Post-synthesis Project: You will be able to add sources, view device resources, run design analysis, planning and implementation.
☐ Do not specify sources at this time

○ I/O Planning Project
Do not specify design sources. You will be able to view part/package resources.

○ Imported Project
Create a Vivado project from a Synplify, XST or ISE Project File.

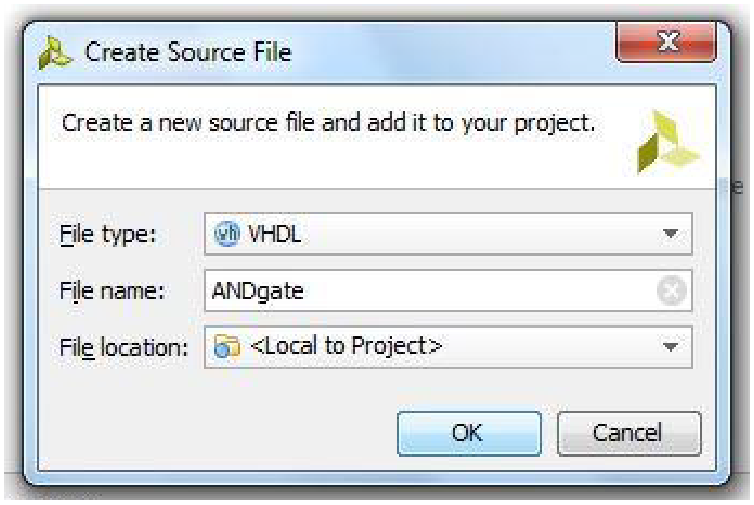
○ Example Project
Create a new Vivado project from a predefined template.

New Project Wizard will guide you through the process of selecting design sources and a target device for a new project.

Because the sources will be specified at this time, leave the dialog box “Do not specify sources at this time” unchecked.

Then, create a new source. Click the green + and select create new source. Name the source ORgate. Make sure the File type VHDL is selected.

Figure 3: Adding Sources


Create Source File

Create a new source file and add it to your project.

File type: VHDL

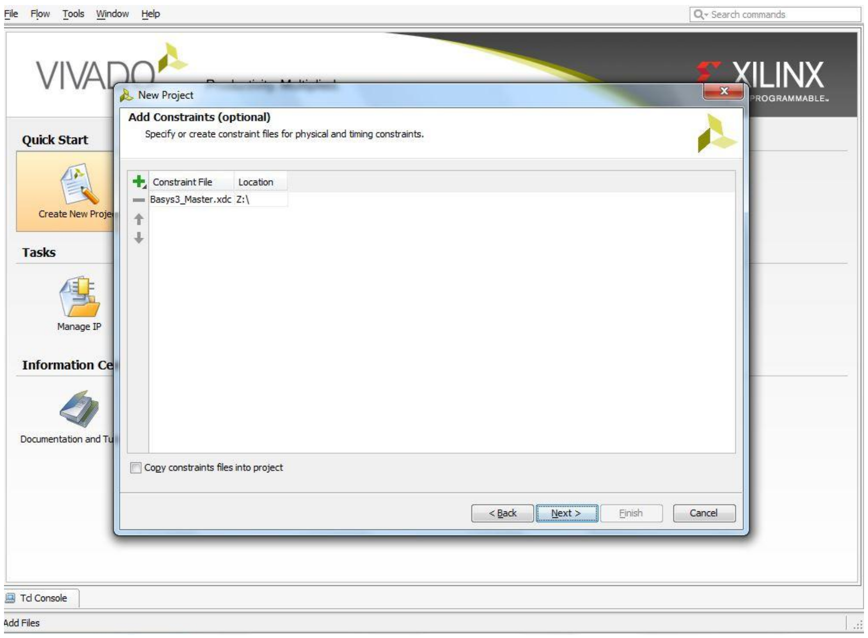
File name: ANDgate

File location: <Local to Project>

IP does not need to be added. Click the next button

Next, the constraints will be added. The constraint file is the master XDC file provided by Digilent. This will be provided by the professor or on the Basys3 homepage: https://www.digilentinc.com/Products/Detail.cfm?NavPath=2,400,1288&Prod=BASYS3

Figure 4: Adding Constraints



New Project

Add Constraints (optional)

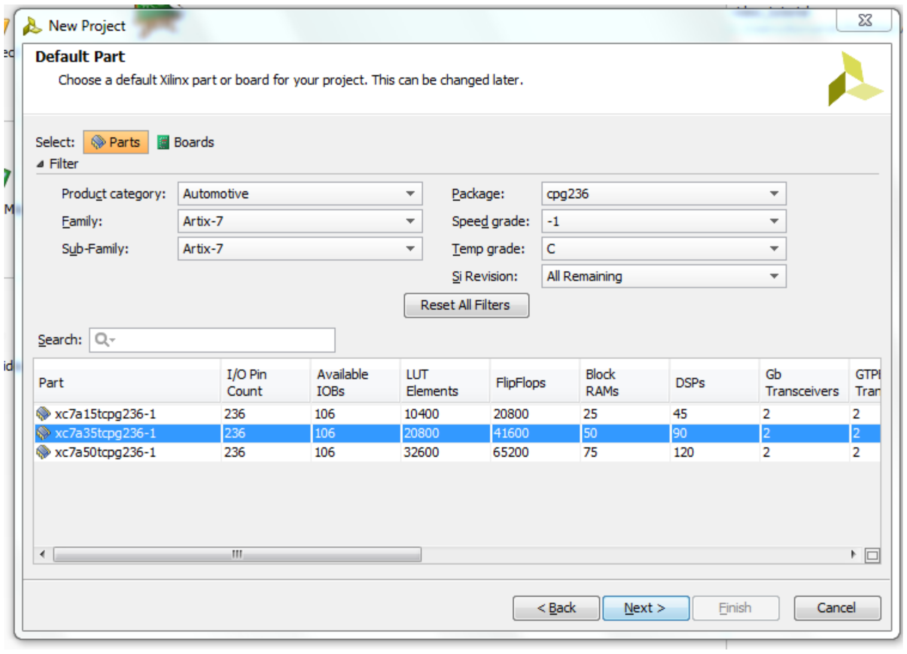
Specify or create constraint files for physical and timing constraints.

Constraint File	Location
Basys3_Master.xdc	Z:\

[ ] Copy constraints files into project

Next, the type of chip that is used must be selected. The Basys3 uses an Artix-7 FPGA chip. Enter the information as it is displayed in Figure 5 (**this is important**) and then click next:

Figure 5: Default Part


New Project

Default Part

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

Select: Parts Boards

Filter

Product category:	Automotive	Package:	cpg236
Family:	Artix-7	Speed grade:	-1
Sub-Family:	Artix-7	Temp grade:	C
		Si Revision:	All Remaining

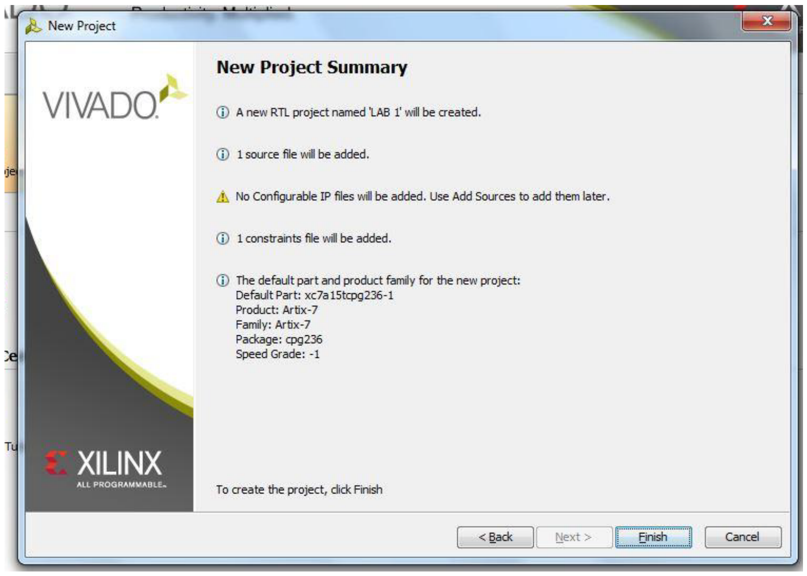
Reset All Filters

Search:

Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	DSPs	Gb Transceivers	GTPI Tran
xc7a15tcpg236-1	236	106	10400	20800	25	45	2	2
xc7a35tcpg236-1	236	106	20800	41600	50	90	2	2
xc7a50tcpg236-1	236	106	32600	65200	75	120	2	2

After all of these steps are followed a screen resembling figure 6 will be displayed. This is the project summary. It provides a preview of what is being created. Click finish when ready to move on.

Figure 6: Project Summary



New Project Summary

A new RTL project named 'LAB 1' will be created.

1 source file will be added.

No Configurable IP files will be added. Use Add Sources to add them later.

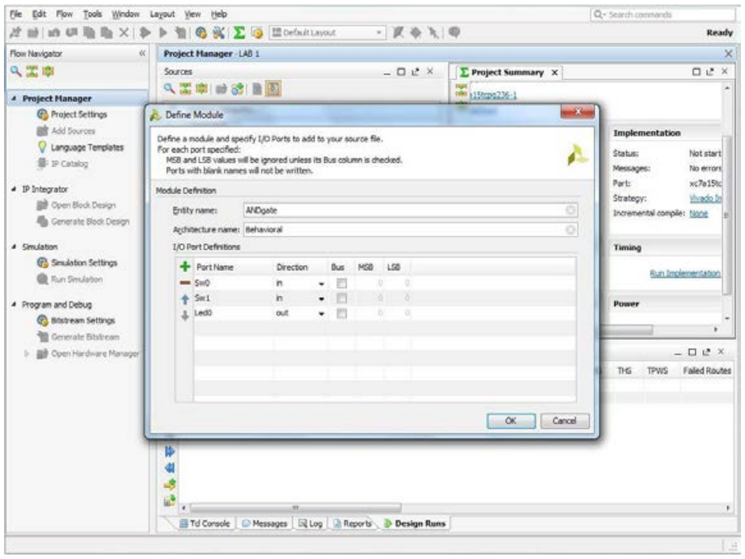
1 constraints file will be added.

The default part and product family for the new project:
Default Part: xc7a15tcpg236-1
Product: Artix-7
Family: Artix-7
Package: cpg236
Speed Grade: -1

To create the project, click Finish

The next step is to indicate inputs and outputs. This is how the Basys3 knows which pins are the inputs and outputs for the project. Enter the information located in Figure 7.

Figure 7: Inputs and Outputs


Define Module

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: ANDgate

Architecture name: Behavioral

I/O Port Definitions

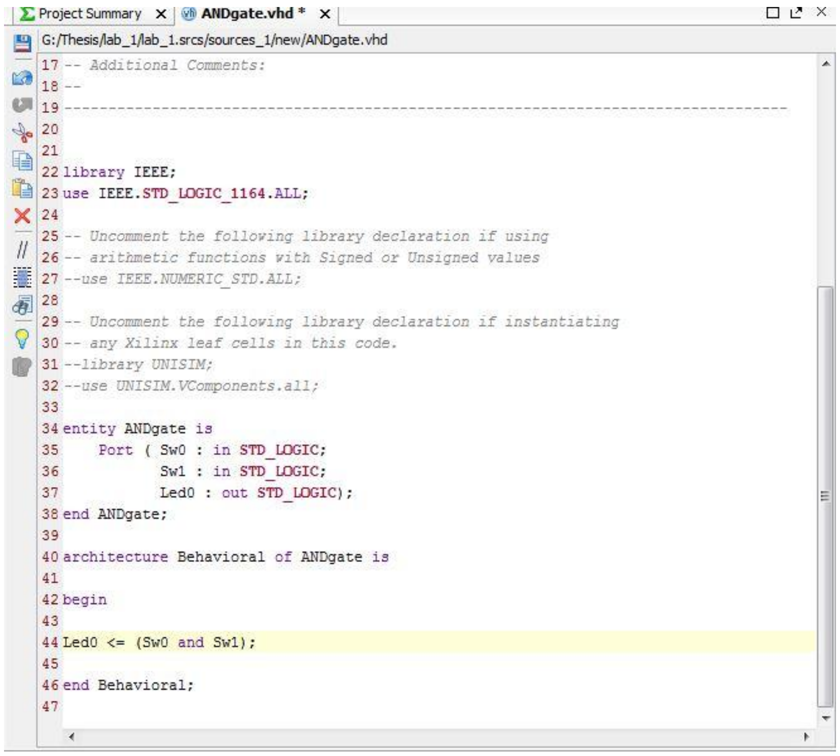
Port Name	Direction	Bus	MSB	LSB
Sw0	in	☐	0	0
Sw1	in	☐	0	0
Led0	out	☐	0	0

After these steps are accomplished the program must be written.

Double click the file name ANDgate.

Within this file insert the information in Figure 8 that is below the phrase “begin”:

Figure 8: AND gate file


G:/Thesis/lab_1/lab_1.srcs/sources_1/new/ANDgate.vhd

```
-- Additional Comments:
--
----------------------------------------------------------------------------------


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 ANDgate is
    Port ( Sw0 : in STD_LOGIC;
           Sw1 : in STD_LOGIC;
           Led0 : out STD_LOGIC);
end ANDgate;

architecture Behavioral of ANDgate is

begin

Led0 <= (Sw0 and Sw1);

end Behavioral;

```

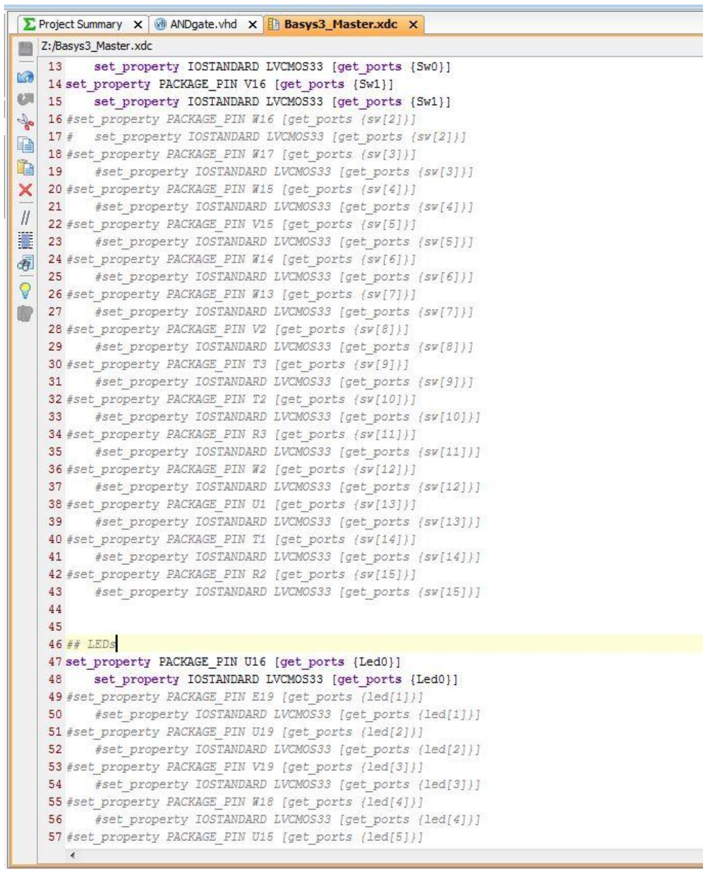
This is the code that the program reads to implement the AND gate. The inputs and outputs should be automatically inputted into the file so all that needs to be written is the code that will program the gate. Led0 is the output. The “<=” means ‘gets’ in VHDL. This basically means that the output will be 1 (led on) when the condition within the parentheses is met. Programming the actual gate is very simple. The “and” function is a preprogrammed command within the language. All that needs to be done is to indicate that the inputs Sw0 and Sw1 will be involved with the “and” function.

Next go to the master XDC file. Rows of text that are greyed out with “#” at the beginning of each row can be seen. The “#” indicates that the row is a comment and not a piece of code. Comments are nonessential to the program and can be used to give information or indicate a heading.

The rows that say sw[0], sw[1], and led[0] must all be uncommented to allow them to be recognized as ports for the Basys3. This can be done by removing the “#” in front of each row.

Next, make sure that the pins indicated in the master XDC match the names given to the inputs and outputs. Change sw[0] to Sw0, sw[1] to Sw1, and led[0] to Led0. If done correctly the file will look like Figure 9.

Figure 9: Master XDC



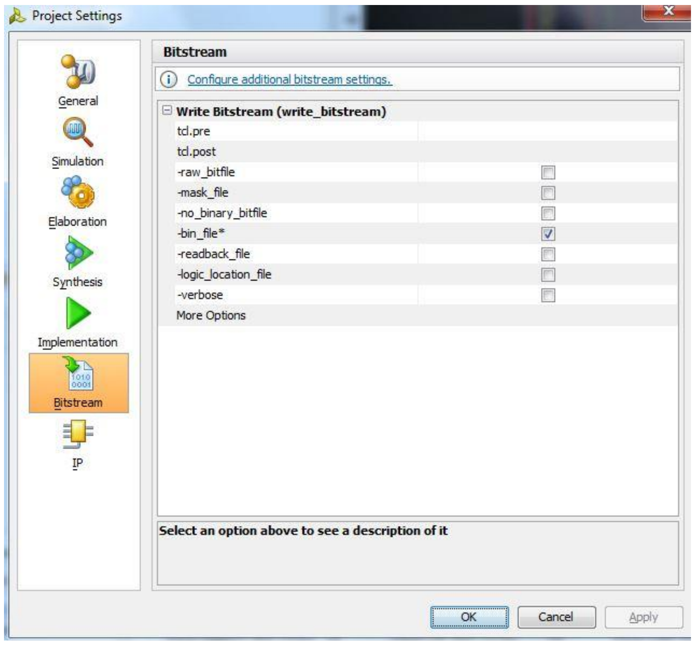
```
    set_property IOSTANDARD LVCMOS33 [get_ports {Sw0}]
set_property PACKAGE_PIN V16 [get_ports {Sw1}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Sw1}]
#set_property PACKAGE_PIN W16 [get_ports {sw[2]}]
#   set_property IOSTANDARD LVCMOS33 [get_ports {sw[2]}]
#set_property PACKAGE_PIN W17 [get_ports {sw[3]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[3]}]
#set_property PACKAGE_PIN W15 [get_ports {sw[4]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[4]}]
#set_property PACKAGE_PIN V15 [get_ports {sw[5]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[5]}]
#set_property PACKAGE_PIN W14 [get_ports {sw[6]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[6]}]
#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]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[12]}]
#set_property PACKAGE_PIN U1 [get_ports {sw[13]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[13]}]
#set_property PACKAGE_PIN T1 [get_ports {sw[14]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[14]}]
#set_property PACKAGE_PIN R2 [get_ports {sw[15]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[15]}]


## LEDs
set_property PACKAGE_PIN U16 [get_ports {Led0}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Led0}]
#set_property PACKAGE_PIN E19 [get_ports {led[1]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[1]}]
#set_property PACKAGE_PIN U19 [get_ports {led[2]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[2]}]
#set_property PACKAGE_PIN V19 [get_ports {led[3]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[3]}]
#set_property PACKAGE_PIN W18 [get_ports {led[4]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[4]}]
#set_property PACKAGE_PIN U15 [get_ports {led[5]}]
```

After the correct rows have been uncommented, the program is ready to be loaded onto the Basys3.

First, make sure that the correct file is generated. Click the Bitstream settings button in the lower left hand corner. The option -bin_file should be selected. This ensures that the file generated will be compatible with the Basys3. This is shown in Figure 10.

Figure 10: .bin File settings



Bitstream

Configure additional bitstream settings.

Write Bitstream (write_bitstream)

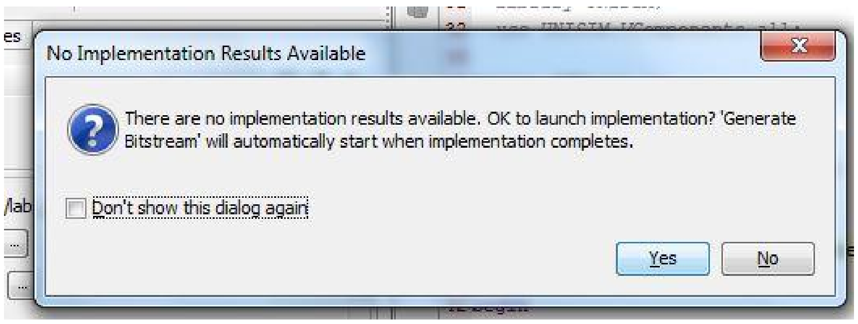
tcl.pre	
tcl.post	
-raw_bitfile	☐
-mask_file	☐
-no_binary_bitfile	☐
-bin_file*	☑
-readback_file	☐
-logic_location_file	☐
-verbose	☐
More Options	

Select an option above to see a description of it

Then, click the run Bitstream button in the lower left hand corner.

There will be a dialog box that says that there are no implementation results available (Figure 11).

Figure 11: Implementation



No Implementation Results Available

There are no implementation results available. OK to launch implementation? 'Generate Bitstream' will automatically start when implementation completes.

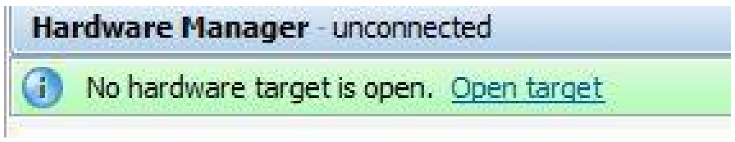
☐ Don't show this dialog again

All this means is that the design has to be implemented before the bitstream can be produced. This process could take a few minutes. After the bitstream is generated, the program must be loaded onto the Basys3.

Make sure that the power is turned on and the green LED is illuminated.

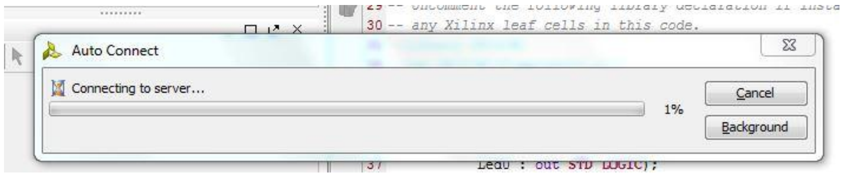
Next, open the Hardware Manager. There will be an option to “Open target’ (Figure 12) at the top of the screen. Click that and then click auto connect (Figure 13). This should automatically connect the Basys3 to the computer.

Figure 12: Hardware Manager


Hardware Manager - unconnected

No hardware target is open. Open target

Figure 13: Auto Connect.

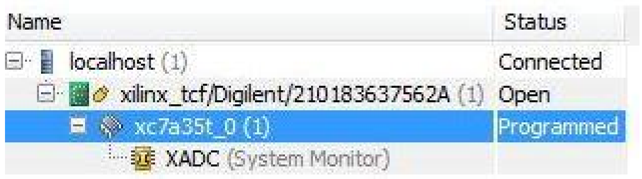

Auto Connect

Connecting to server...

1%

After these steps are completed the screen displayed in Figure 14 will be displayed on the screen. This confirms that the program has been loaded onto the Basys3.

Figure 14: Basys3 is programmed.



Name	Status
localhost (1)	Connected
xilinx_tcf/Digilent/210183637562A (1)	Open
xc7a35t_0 (1)	Programmed
XADC (System Monitor)	

Conclusion:

If everything is done correctly an AND gate should be modeled on the Basys3. When either switch 0 or switch 1 is turned to the off position (0) LED 0 should be turned off. When both of the switches are turned on (1) the LED should light up to indicate the output is 1.

Appendix B

Lab 2

ENTC 3370

OR Gate

Objective:

To familiarize the student with the Basys3 board and to introduce him or her to basic logic gates.

Parts and Equipment:

- Basys3 board
- USB cable
- Computer
- Vivado Design Suite

Introduction:

In this lab the Basys3 FPGA board will be used to model an OR gate. For an OR gate the output should be 1 if any input is 1 and 0 only if both inputs are 0. The inputs in this lab will be represented by switches 0 and 1 on the Basys3. The output will be represented by LED 0.

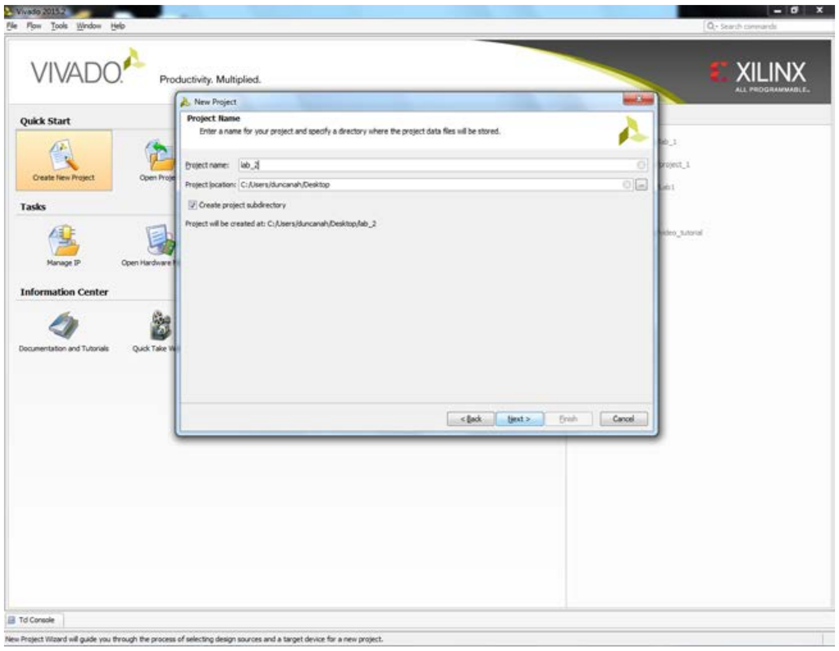
Procedure:

First the Basys3 must be configured in the correct way. The jumper in the top left corner has to be set to the USB setting. This allows the Basys3 to be powered using USB. The top right jumper must be set to the JTAG setting. This is the way the program will be loaded onto the board. Make sure the included USB cable is connected to the board and also to the computer.

Then, load Vivado and click new project.

A pop up will appear on the screen, click next. Name the file lab_2 and click next. Make sure to save the file in a place where it can be accessed later, such as the Z drive.

Figure 1: Naming the Project


New Project

Project Name

Enter a name for your project and specify a directory where the project data files will be stored.

Project name: lab_2

Project location: C:/Users/duncanah/Desktop

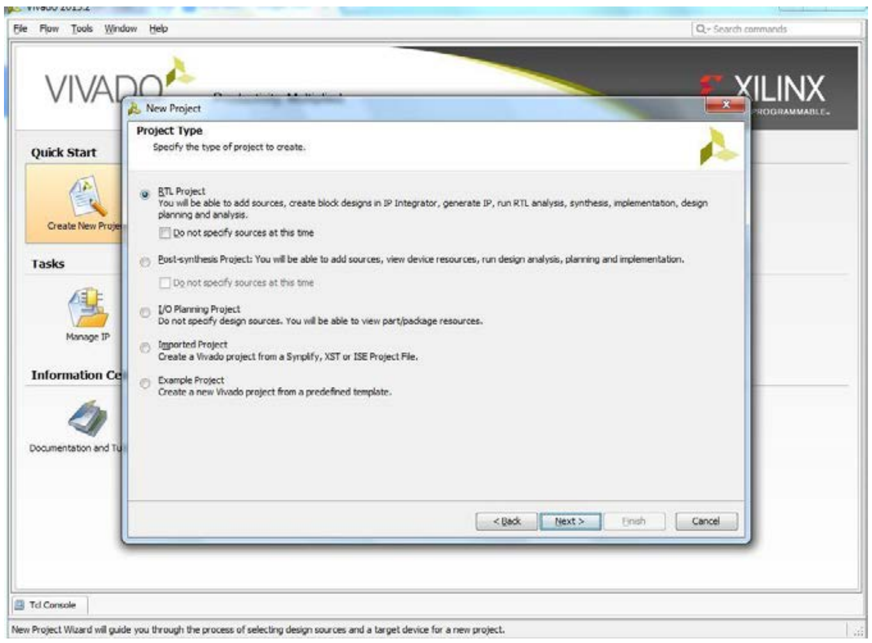
☑ Create project subdirectory

Project will be created at: C:/Users/duncanah/Desktop/lab_2

New Project Wizard will guide you through the process of selecting design sources and a target device for a new project.

Next, specify the type of project that is to be made. Select the RTL Project option as illustrated in Figure 2. RTL stands for register-transfer level.

Figure 2: Project Type


New Project

Project Type

Specify the type of project to create.

● RTL Project
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.

☐ Do not specify sources at this time

○ Post-synthesis Project: You will be able to add sources, view device resources, run design analysis, planning and implementation.

☐ Do not specify sources at this time

○ I/O Planning Project
Do not specify design sources. You will be able to view part/package resources.

○ Imported Project
Create a Vivado project from a Synplify, XST or ISE Project File.

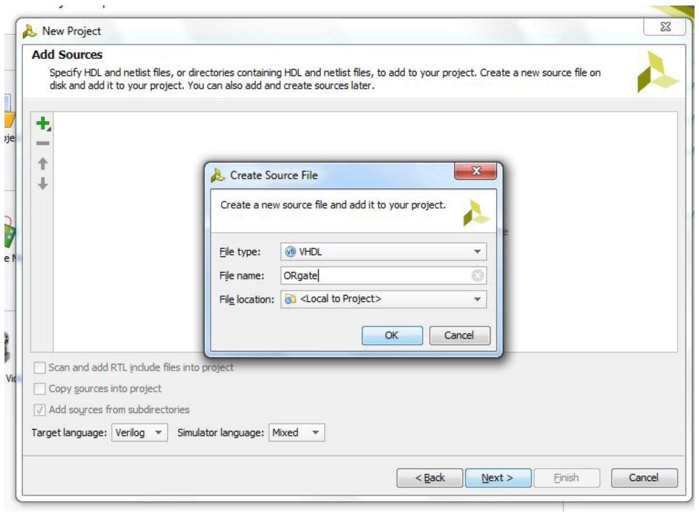
○ Example Project
Create a new Vivado project from a predefined template.

New Project Wizard will guide you through the process of selecting design sources and a target device for a new project.

Because the sources will be specified at this time leave the dialog box “Do not specify sources at this time” unchecked.

Then, create a new source. Click the green + and select create new source. Name the source ORgate. Make sure the File type VHDL is selected.

Figure 3: Adding Sources


New Project

Add Sources

Specify HDL and netlist files, or directories containing HDL and netlist files, to add to your project. Create a new source file on disk and add it to your project. You can also add and create sources later.

Create Source File

Create a new source file and add it to your project.

File type: VHDL

File name: ORgate

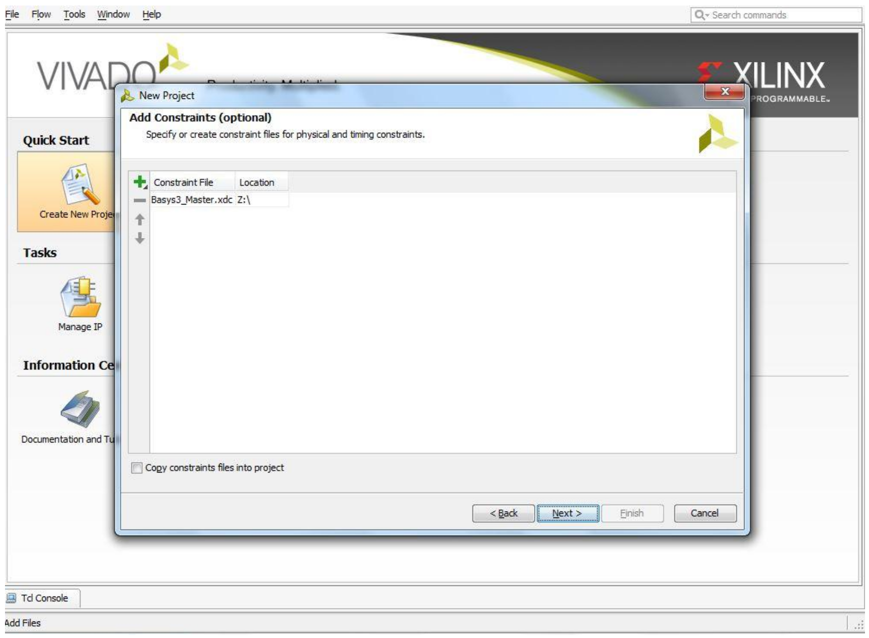
File location: <Local to Project>

OK
Cancel
Scan and add RTL include files into project
Copy sources into project
Add sources from subdirectories
Target language: Verilog
Simulator language: Mixed
< Back
Next >
Finish
Cancel

IP does not need to be added. Click the next button

Next, constraints must be added. The constraint file is the master XDC file provided by Digilent. This will be provided by the professor or on the Basys3 homepage: https://www.digilentinc.com/Products/Detail.cfm?NavPath=2,400,1288&Prod=BASYS3

Figure 4: Adding Constraints


New Project

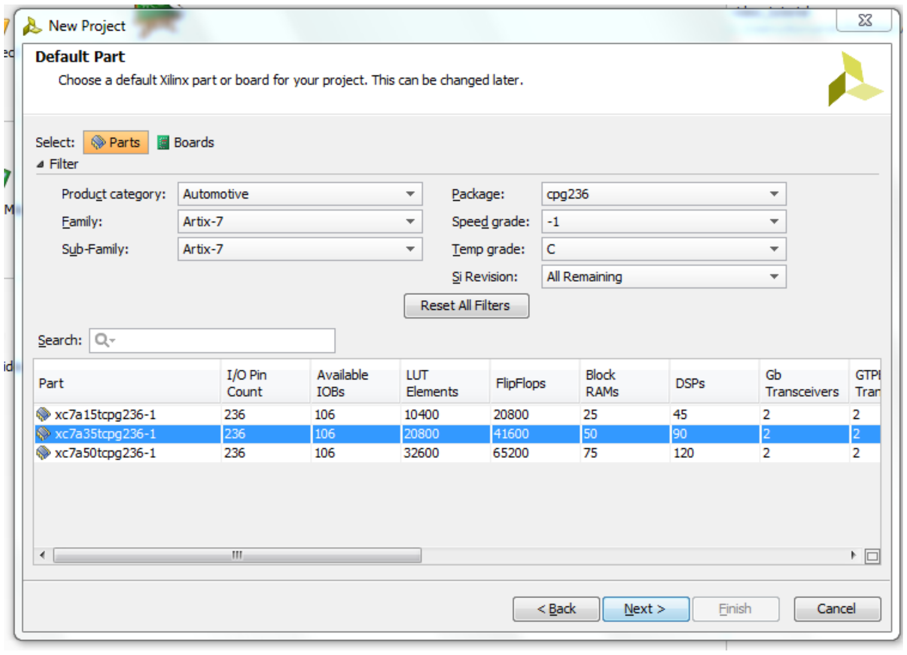
Add Constraints (optional)

Specify or create constraint files for physical and timing constraints.

Constraint File	Location
Basys3_Master.xdc	Z:\

Next, designate which type of chip that is being used. The Basys3 uses an Artix-7 FPGA chip. Enter the information as it is displayed in Figure 5 (**this is important**) and then click next:

Figure 5: Default Part


New Project

Default Part

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

Select: Parts Boards

Filter

Product category: Automotive
Family: Artix-7
Sub-Family: Artix-7
Package: cpg236
Speed grade: -1
Temp grade: C
Si Revision: All Remaining

Reset All Filters

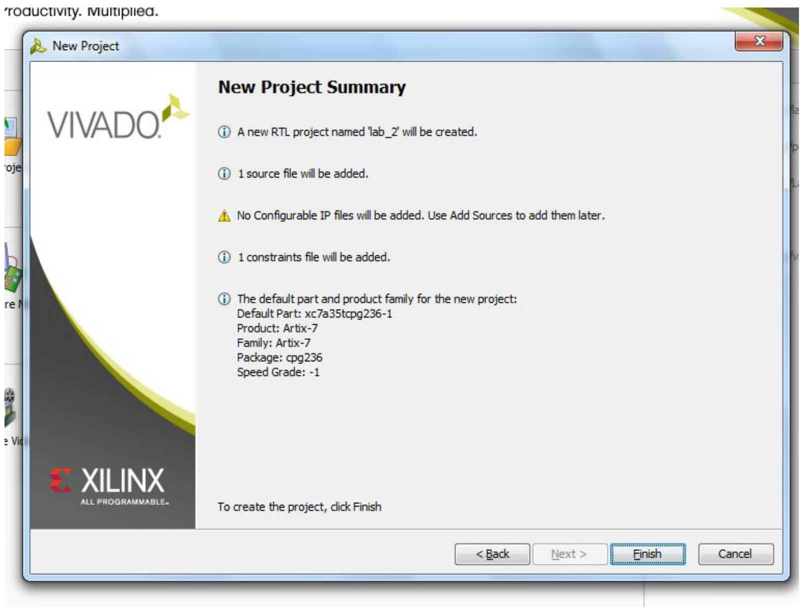
Search:

Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	DSPs	Gb Transceivers	GTPI Tran
xc7a15tcpg236-1	236	106	10400	20800	25	45	2	2
xc7a35tcpg236-1	236	106	20800	41600	50	90	2	2
xc7a50tcpg236-1	236	106	32600	65200	75	120	2	2

< Back | Next > | Finish | Cancel

After all of these steps are followed a screen resembling figure 6 will be displayed. This is the project summary. It provides a preview of what is being created. Click finish when ready to move on.

Figure 6: Project Summary


New Project

New Project Summary

A new RTL project named 'lab_2' will be created.

1 source file will be added.

No Configurable IP files will be added. Use Add Sources to add them later.

1 constraints file will be added.

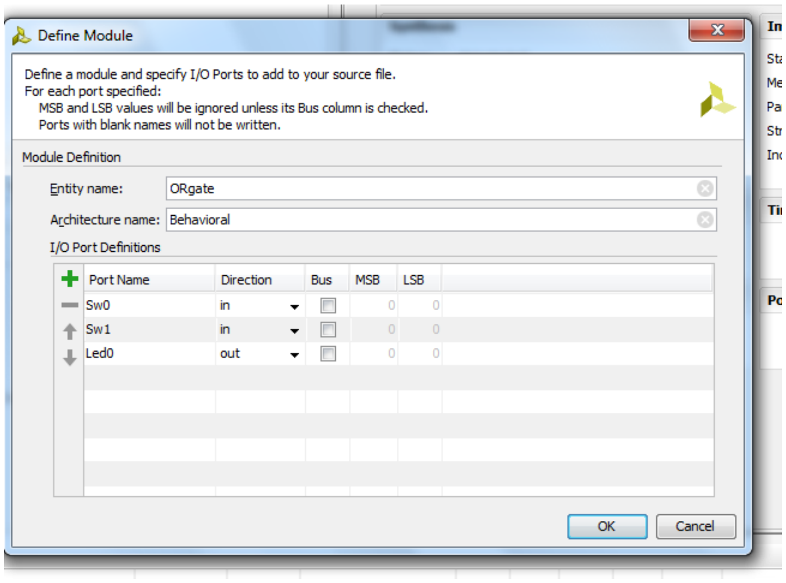
The default part and product family for the new project:
Default Part: xc7a35tcpg236-1
Product: Artix-7
Family: Artix-7
Package: cpg236
Speed Grade: -1

To create the project, click Finish

< Back | Next > | Finish | Cancel

The next step is to specify the inputs and outputs. This is how the Basys3 knows which pins are the inputs and outputs for the project. Enter the information located in Figure 7.

Figure 7: Inputs and Outputs



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: ORgate

Architecture name: Behavioral

I/O Port Definitions

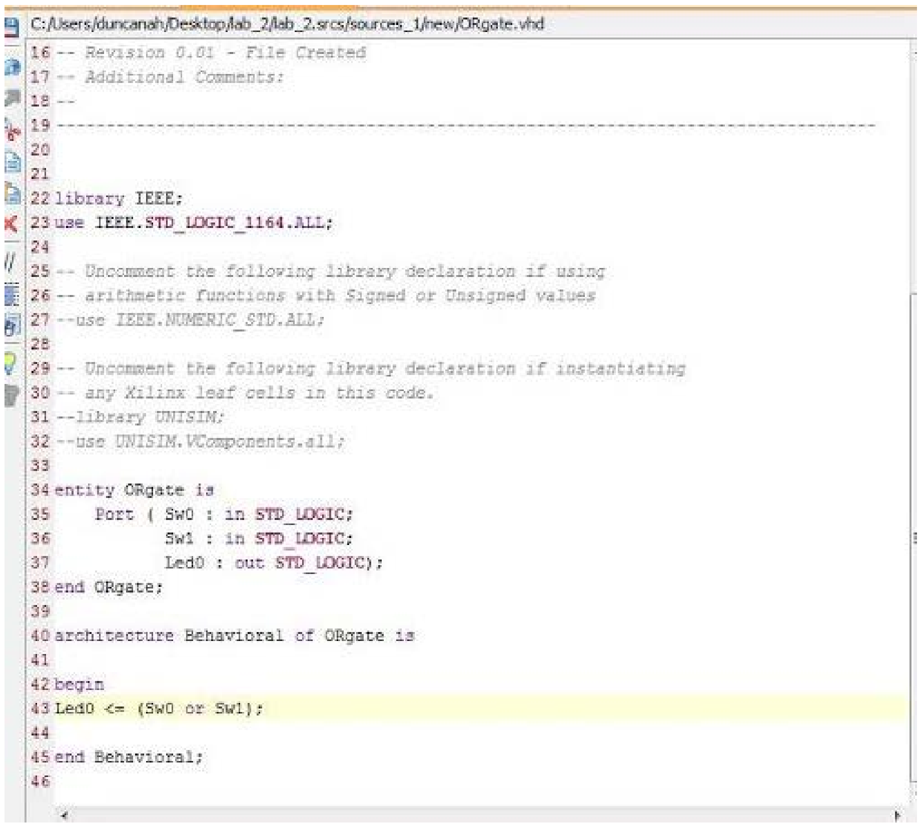
Port Name	Direction	Bus	MSB	LSB
Sw0	in	☐	0	0
Sw1	in	☐	0	0
Led0	out	☐	0	0

After these steps are accomplished the program is ready to be written.

Double click the file name ORgate.

Within this file insert the information in Figure 8 that is below “begin”:

Figure 8: OR gate file



C:/Users/duncanah/Desktop/lab_2/lab_2.srcs/sources_1/new/ORgate.vhd

```
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


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 ORgate is
    Port ( Sw0 : in STD_LOGIC;
           Sw1 : in STD_LOGIC;
           Led0 : out STD_LOGIC);
end ORgate;

architecture Behavioral of ORgate is

begin
Led0 <= (Sw0 or Sw1);

end Behavioral;

```

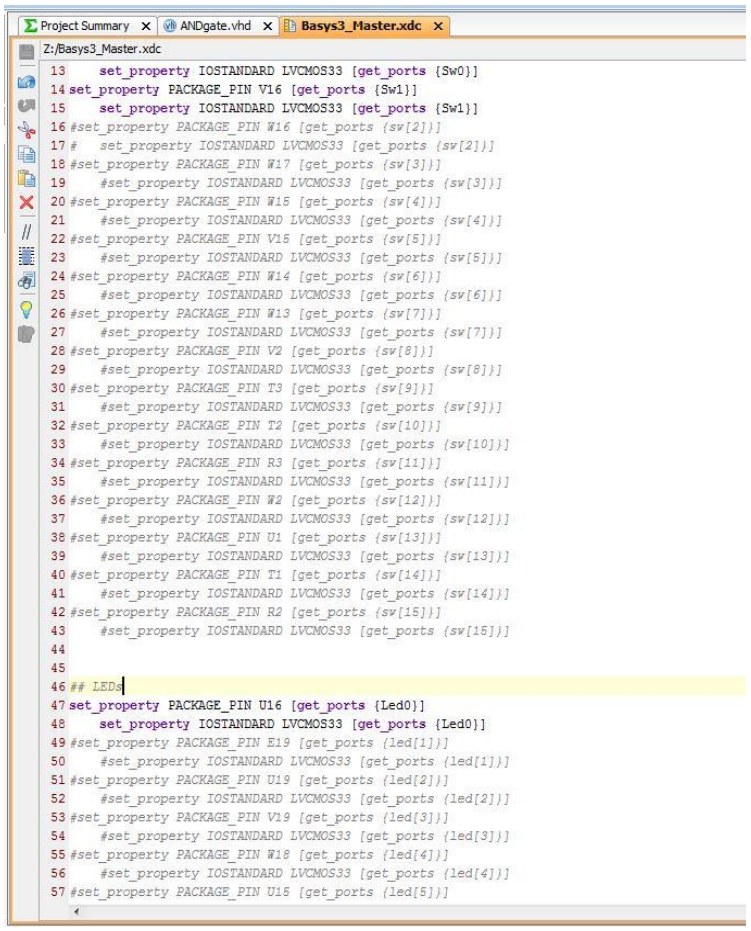
This is the code that the program reads to implement the OR gate. The inputs and outputs should be automatically inputted into the file so all that needs to be written is the code that will program the gate. Led0 is the output. The “<=” means ‘gets’ in VHDL. This basically means that the output will be 1 (led on) when the condition within the parentheses is met. Programming the actual gate is very simple. The “or” function is a preprogrammed command within the language. All that needs to be done is to indicate that the inputs Sw0 and Sw1 will be involved with the “or” function.

Next go to the master XDC file. Rows of text that are greyed out with “#” at the beginning of each row can be seen. The “#” indicates that the row is a comment and not a piece of code. Comments are nonessential to the program and can be used to give information or indicate a heading.

The rows that say sw[0], sw[1], and led[0] must all be uncommented to allow them to be recognized as ports for the Basys3. This can be done by removing the “#” in front of each row.

Next, make sure that the pins indicated in the master XDC match the names given to the inputs and outputs. Change sw[0] to Sw0, sw[1] to Sw1, and led[0] to Led0. If done correctly the file will look like Figure 9.

Figure 9: Master XDC



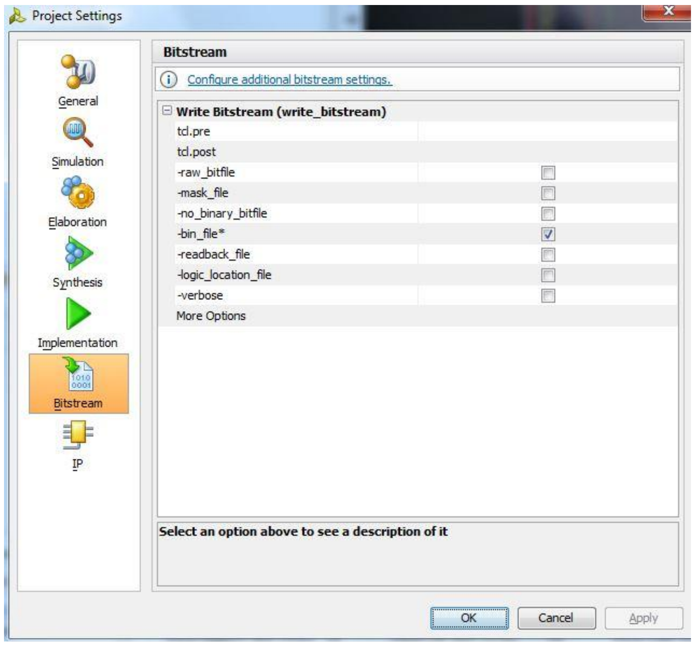
```
	set_property IOSTANDARD LVCMOS33 [get_ports {Sw0}]
set_property PACKAGE_PIN V16 [get_ports {Sw1}]
	set_property IOSTANDARD LVCMOS33 [get_ports {Sw1}]
#set_property PACKAGE_PIN W16 [get_ports {sw[2]}]
#	set_property IOSTANDARD LVCMOS33 [get_ports {sw[2]}]
#set_property PACKAGE_PIN W17 [get_ports {sw[3]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[3]}]
#set_property PACKAGE_PIN W15 [get_ports {sw[4]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[4]}]
#set_property PACKAGE_PIN V15 [get_ports {sw[5]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[5]}]
#set_property PACKAGE_PIN W14 [get_ports {sw[6]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[6]}]
#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]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[12]}]
#set_property PACKAGE_PIN U1 [get_ports {sw[13]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[13]}]
#set_property PACKAGE_PIN T1 [get_ports {sw[14]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[14]}]
#set_property PACKAGE_PIN R2 [get_ports {sw[15]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[15]}]


## LEDs
set_property PACKAGE_PIN U16 [get_ports {Led0}]
	set_property IOSTANDARD LVCMOS33 [get_ports {Led0}]
#set_property PACKAGE_PIN E19 [get_ports {led[1]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {led[1]}]
#set_property PACKAGE_PIN U19 [get_ports {led[2]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {led[2]}]
#set_property PACKAGE_PIN V19 [get_ports {led[3]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {led[3]}]
#set_property PACKAGE_PIN W18 [get_ports {led[4]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {led[4]}]
#set_property PACKAGE_PIN U15 [get_ports {led[5]}]
```

After the correct rows have been uncommented, the program is ready to be loaded onto the Basys3.

First, make sure that the correct file is generated. Click the Bitstream settings button in the lower left hand corner. The option -bin_file should be selected. This ensures that the file generated will be compatible with the Basys3. This is shown in Figure 10.

Figure 10: .bin File settings



Bitstream

Configure additional bitstream settings.

Write Bitstream (write_bitstream)

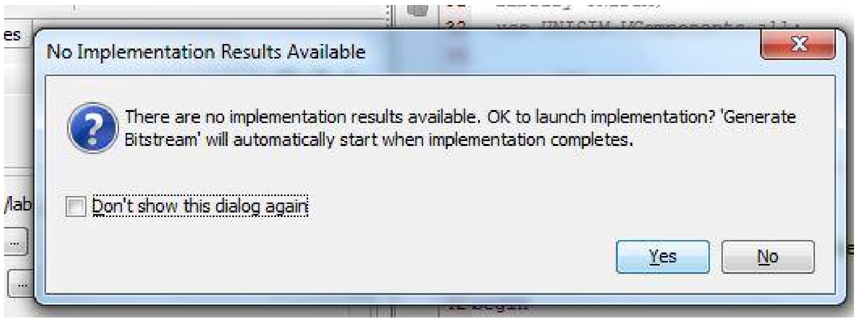
tcl.pre	
tcl.post	
-raw_bitfile	☐
-mask_file	☐
-no_binary_bitfile	☐
-bin_file*	✓
-readback_file	☐
-logic_location_file	☐
-verbose	☐
More Options	

Select an option above to see a description of it

Then, click the run Bitstream button in the lower left hand corner.

There will be a dialog box that says that there are no implementation results available (Figure 11).

Figure 11: Implementation



No Implementation Results Available

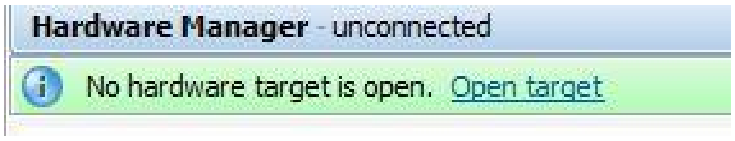
There are no implementation results available. OK to launch implementation? 'Generate Bitstream' will automatically start when implementation completes.

☐ Don't show this dialog again

All this means is that the design has to be implemented before the bitstream can be produced. This process could take a few minutes. After the bitstream is generated the program must be loaded onto the Basys3.

Next, open the Hardware Manager. There will be an option to “Open target’ (Figure 12) at the top of the screen. Click that and then click auto connect (Figure 13). This should automatically connect the Basys3 to the computer.

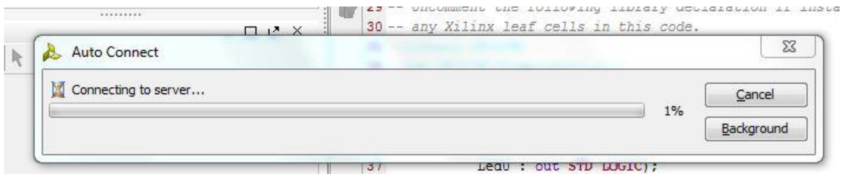
Figure 12: Hardware Manager



Hardware Manager - unconnected

No hardware target is open. Open target

Figure 13: Auto Connect.



-- any Xilinx leaf cells in this code.

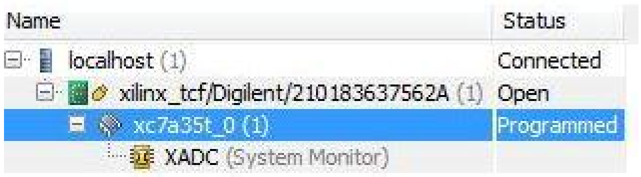
Auto Connect

Connecting to server...

1%

After these steps are completed the screen displayed in Figure 14 will be displayed on the screen. This confirms that the program has been loaded onto the Basys3.

Figure 14: Basys3 is programmed.



Name	Status
localhost (1)	Connected
xilinx_tcf/Digilent/210183637562A (1)	Open
xc7a35t_0 (1)	Programmed
XADC (System Monitor)	

Conclusion:

If everything is done correctly an OR gate should be modeled on the Basys3. When both of the switches are turned to the off position (0) LED 0 should be turned off. If either switch is switched on (1) the LED should light up to indicate the output is 1.

Appendix C

Lab 3

ENTC 3370

NAND Gate

Objective:

To familiarize the student with the Basys3 board and to introduce him or her to basic logic gates.

Parts and Equipment:

- Basys3 board
- USB cable
- Computer
- Vivado Design Suite

Introduction:

In this lab we will be using the Basys3 FPGA board to model a NAND gate. For a NAND gate the output will be 0 if both inputs are 1, otherwise the output is 1. The inputs in this lab will be represented by switches 0 and 1 on the Basys3. The output will be represented by LED 0.

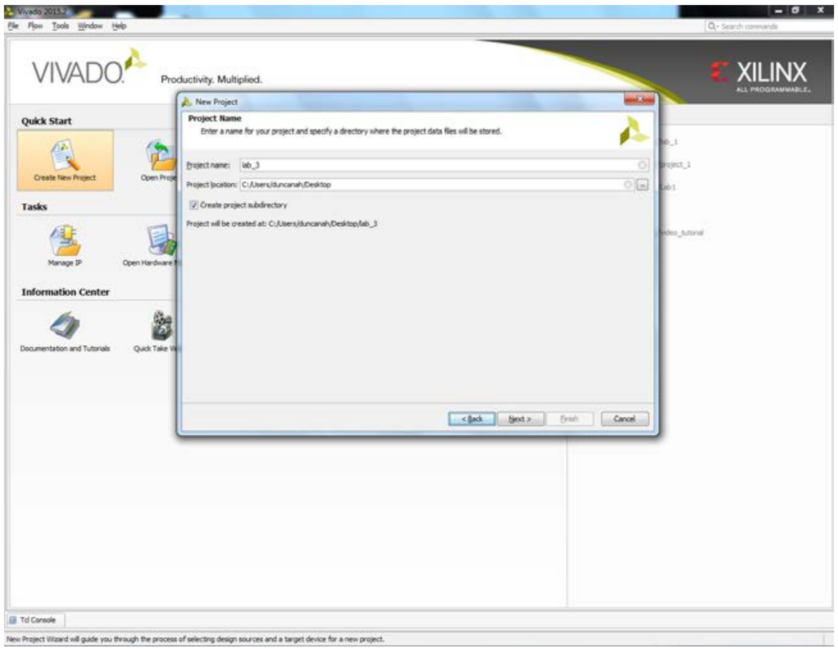
Procedure:

First the Basys3 must be configured in the correct way. The jumper in the top left corner has to be set to the USB setting. This allows the Basys3 to be powered using USB. The top right jumper must be set to the JTAG setting. This is the way the program will be loaded onto the board. Make sure the included USB cable is connected to the board and also to the computer.

Then, must load Vivado and click new project.

A pop up will appear on the screen, click next. Name the file lab_3 and click next. Make sure to save the file in a place where it can be accessed later, such as the Z drive.

Figure 1: Naming the Project



New Project

Project Name

Enter a name for your project and specify a directory where the project data files will be stored.

Project name: lab_3

Project location: C:/Users/duncanah/Desktop

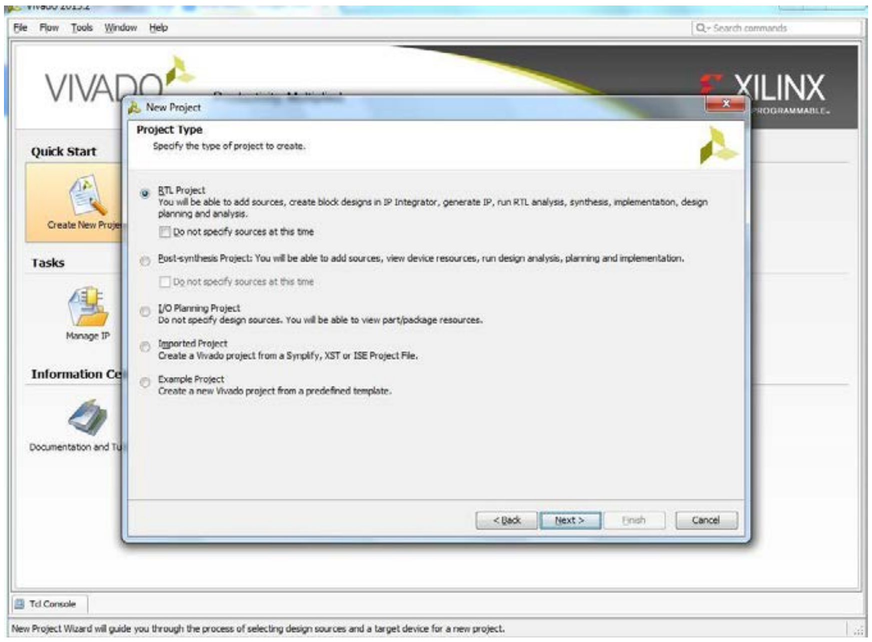
[✓] Create project subdirectory

Project will be created at: C:/Users/duncanah/Desktop/lab_3

New Project Wizard will guide you through the process of selecting design sources and a target device for a new project.

Next, specify the type of project that will be made. Select the RTL Project option as illustrated in Figure 2. RTL stands for register-transfer level.

Figure 2: Project Type



New Project

Project Type

Specify the type of project to create.

● RTL Project
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.
☐ Do not specify sources at this time

○ Post-synthesis Project: You will be able to add sources, view device resources, run design analysis, planning and implementation.
☐ Do not specify sources at this time

○ I/O Planning Project
Do not specify design sources. You will be able to view part/package resources.

○ Imported Project
Create a Vivado project from a Synplify, XST or ISE Project File.

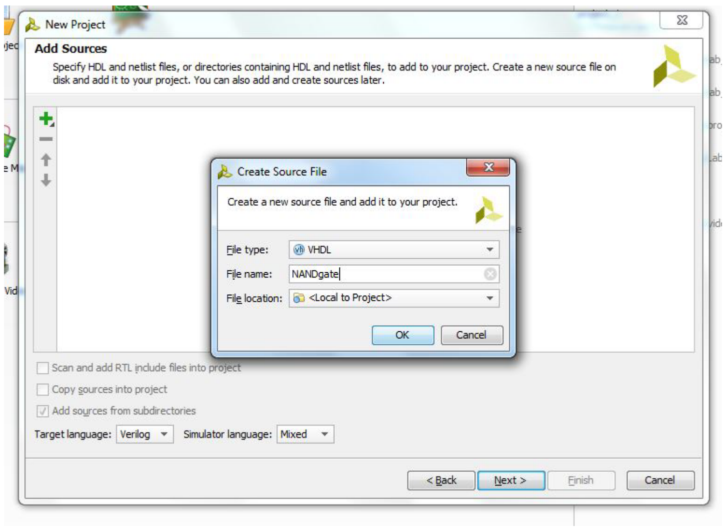
○ Example Project
Create a new Vivado project from a predefined template.

New Project Wizard will guide you through the process of selecting design sources and a target device for a new project.

Because the sources will be specified at this time, leave the dialog box “Do not specify sources at this time” unchecked.

We will then create a new source. Click the green + and select create new source. Name the source NANDgate. Make sure the File type VHDL is selected.

Figure 3: Adding Sources

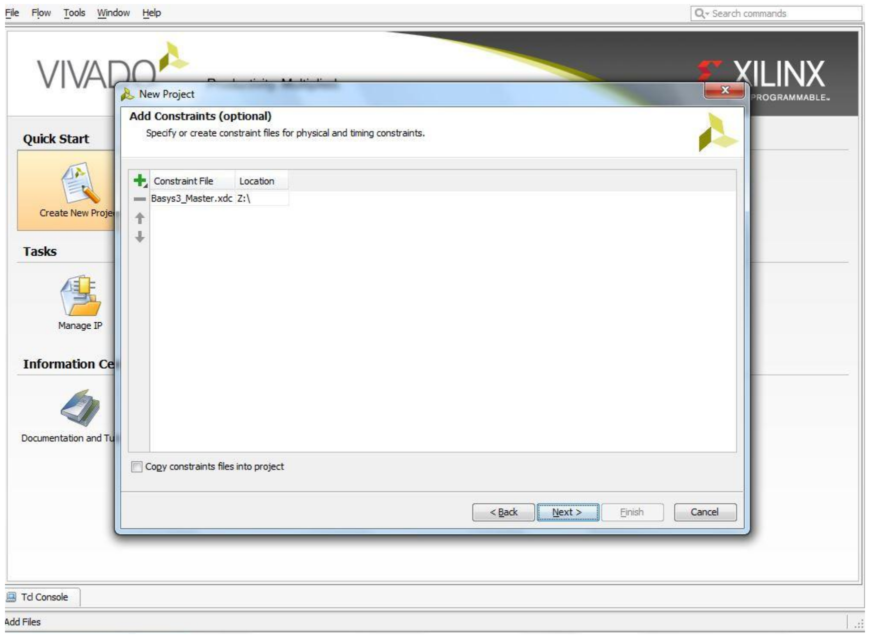

New Project
Add Sources
Specify HDL and netlist files, or directories containing HDL and netlist files, to add to your project. Create a new source file on disk and add it to your project. You can also add and create sources later.
Create Source File
Create a new source file and add it to your project.
OK
Cancel
Scan and add RTL include files into project
Copy sources into project
Add sources from subdirectories
< Back
Next >
Finish
Cancel

File type: VHDL
File name: NANDgate
File location: <Local to Project>
Target language: Verilog
Simulator language: Mixed

We do not need to add IP. Click the next button

Next we will add our constraints. Our constraint file is the master XDC file provided by Digilent. This will be provided by the professor or on the Basys3 homepage: https://www.digilentinc.com/Products/Detail.cfm?NavPath=2,400,1288&Prod=BASYS3

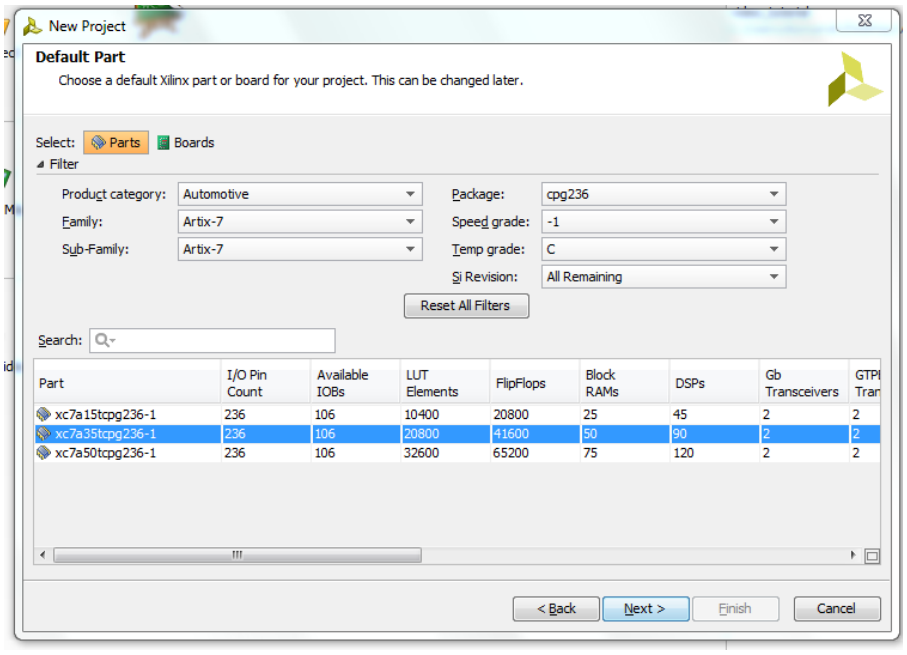
Figure 4: Adding Constraints



Constraint File	Location
Basys3_Master.xdc	Z:\

Next we have to designate which type of chip we are using. The Basys3 uses an Artix-7 FPGA chip. Enter the information as it is displayed in Figure 5 (**this is important**) and then click next:

Figure 5: Default Part

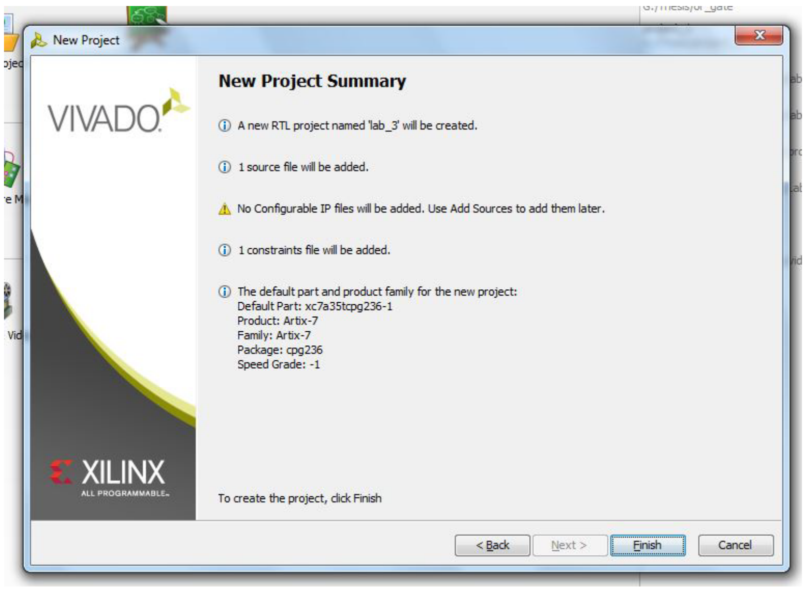


Product category: Automotive
Family: Artix-7
Sub-Family: Artix-7
Package: cpg236
Speed grade: -1
Temp grade: C
Si Revision: All Remaining

Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	DSPs	Gb Transceivers	GTPI Tran
xc7a15tcpg236-1	236	106	10400	20800	25	45	2	2
xc7a35tcpg236-1	236	106	20800	41600	50	90	2	2
xc7a50tcpg236-1	236	106	32600	65200	75	120	2	2

After all of these steps are followed a screen resembling figure 6 will be displayed. This is the project summary. It provides a preview of what is being created. Click finish when ready to move on.

Figure 6: Project Summary



New Project Summary

A new RTL project named 'lab_3' will be created.

1 source file will be added.

No Configurable IP files will be added. Use Add Sources to add them later.

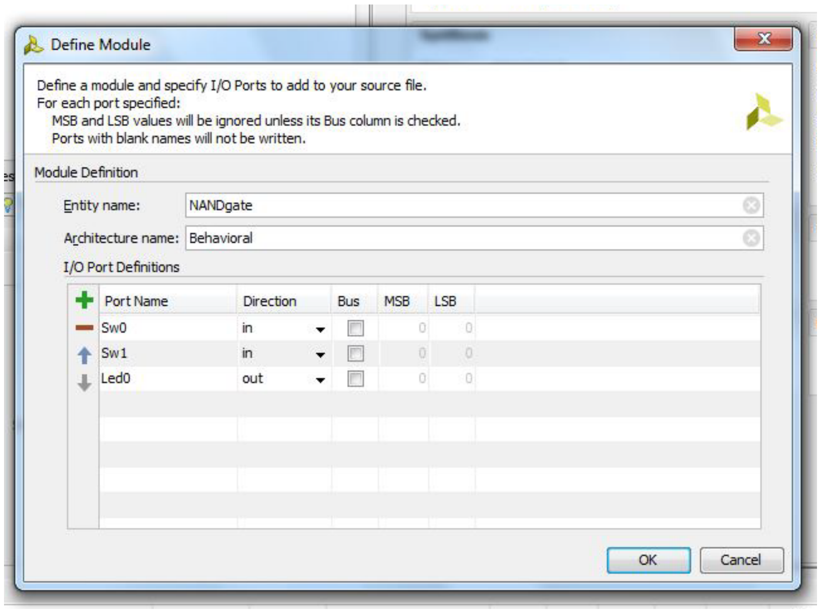
1 constraints file will be added.

The default part and product family for the new project:
Default Part: xc7a35tcpg236-1
Product: Artix-7
Family: Artix-7
Package: cpg236
Speed Grade: -1

To create the project, click Finish

The next step is to specify the inputs and outputs. This is how the Basys3 knows which pins are the inputs and outputs for the project. Enter the information located in Figure 7.

Figure 7: Inputs and Outputs


Define Module

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: NANDgate

Architecture name: Behavioral

I/O Port Definitions

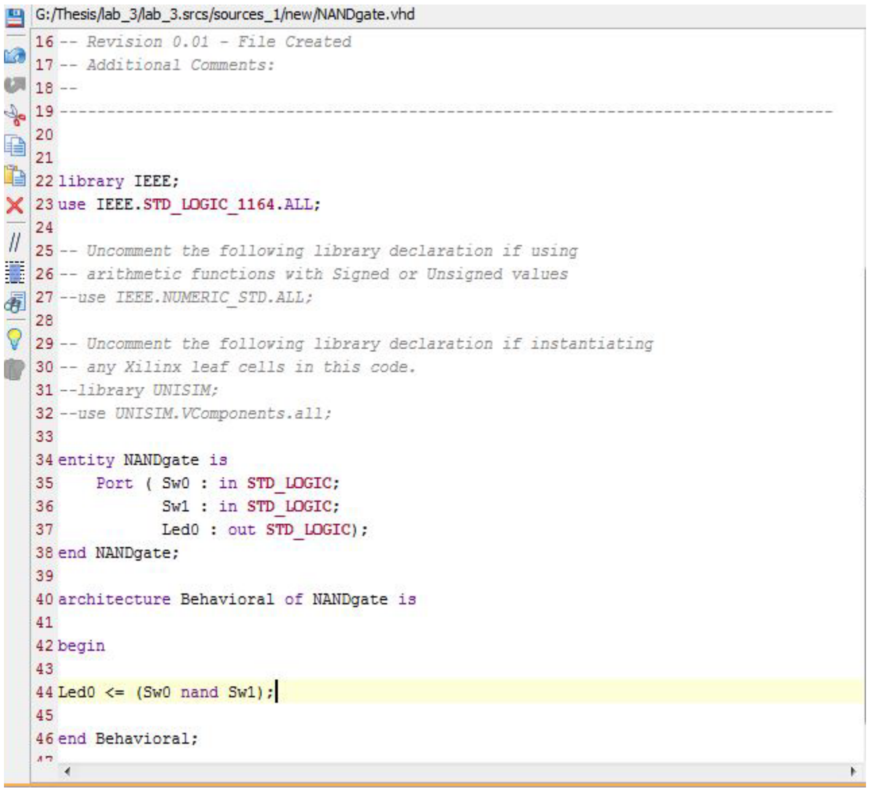
Port Name	Direction	Bus	MSB	LSB
Sw0	in	☐	0	0
Sw1	in	☐	0	0
Led0	out	☐	0	0

After these steps are accomplished the program is ready to be written.

Double click the file name NANDgate.

Within this file insert the information in Figure 8 that is below "begin":

Figure 8: NAND gate file


G:/Thesis/lab_3/lab_3.srcs/sources_1/new/NANDgate.vhd

```
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


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 NANDgate is
    Port ( Sw0 : in STD_LOGIC;
           Sw1 : in STD_LOGIC;
           Led0 : out STD_LOGIC);
end NANDgate;

architecture Behavioral of NANDgate is

begin

Led0 <= (Sw0 nand Sw1);

end Behavioral;
```

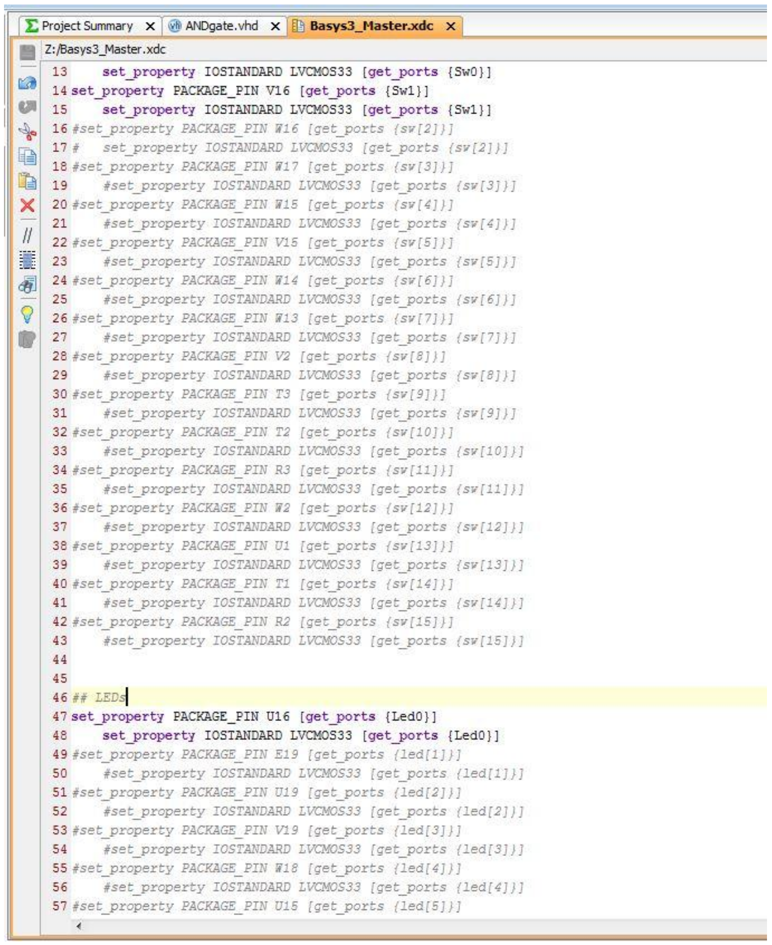
This is the code that the program reads to implement the OR gate. The inputs and outputs should be automatically inputted into the file so all that needs to be written is the code that will program the gate. Led0 is our output. The “<=” means ‘gets’ in VHDL. This basically means that the output will be 1 (led on) when the condition within the parentheses is met. Programming the actual gate is very simple. The “nand” function is a preprogrammed command within the language. All that needs to be done is to indicate that the inputs Sw0 and Sw1 will be involved with the “nand” function.

Next go to the master XDC file. Rows of text that are greyed out with “#” at the beginning of each row can be seen. The “#” indicates that the row is a comment and not a piece of code. Comments are nonessential to the program and can be used to give information or indicate a heading.

The rows that say sw[0], sw[1], and led[0] must all be uncommented to allow them to be recognized as ports for the Basys3. This can be done by removing the “#” in front of each row.

Next, make sure that the pins indicated in the master XDC match the names given to the inputs and outputs. Change sw[0] to Sw0, sw[1] to Sw1, and led[0] to Led0. If done correctly the file will look like Figure 9.

Figure 9: Master XDC



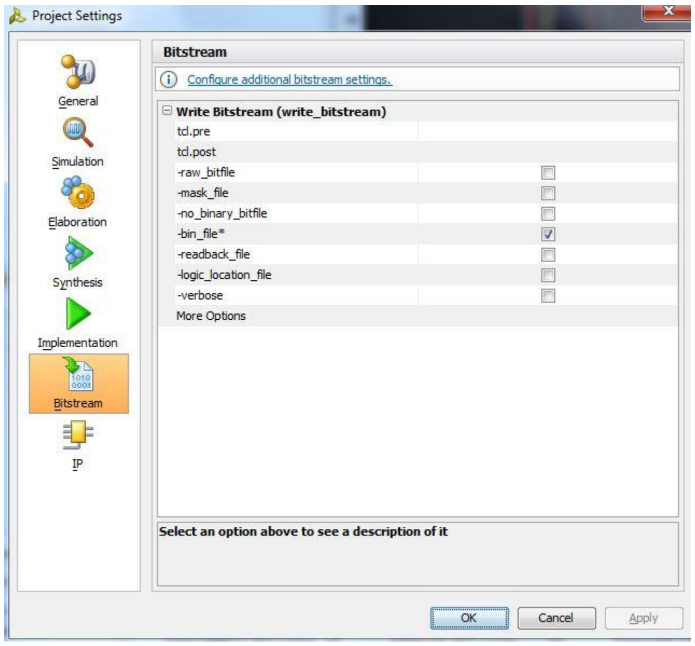
```
    set_property IOSTANDARD LVCMOS33 [get_ports {Sw0}]
set_property PACKAGE_PIN V16 [get_ports {Sw1}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Sw1}]
#set_property PACKAGE_PIN W16 [get_ports {sw[2]}]
#   set_property IOSTANDARD LVCMOS33 [get_ports {sw[2]}]
#set_property PACKAGE_PIN W17 [get_ports {sw[3]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[3]}]
#set_property PACKAGE_PIN W15 [get_ports {sw[4]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[4]}]
#set_property PACKAGE_PIN V15 [get_ports {sw[5]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[5]}]
#set_property PACKAGE_PIN W14 [get_ports {sw[6]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[6]}]
#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]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[12]}]
#set_property PACKAGE_PIN U1 [get_ports {sw[13]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[13]}]
#set_property PACKAGE_PIN T1 [get_ports {sw[14]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[14]}]
#set_property PACKAGE_PIN R2 [get_ports {sw[15]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[15]}]


## LEDs
set_property PACKAGE_PIN U16 [get_ports {Led0}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Led0}]
#set_property PACKAGE_PIN E19 [get_ports {led[1]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[1]}]
#set_property PACKAGE_PIN U19 [get_ports {led[2]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[2]}]
#set_property PACKAGE_PIN V19 [get_ports {led[3]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[3]}]
#set_property PACKAGE_PIN W18 [get_ports {led[4]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {led[4]}]
#set_property PACKAGE_PIN U15 [get_ports {led[5]}]
```

After the correct rows have been uncommented, the program is ready to be loaded onto the Basys3.

First, make sure that the correct file is generated. Click the Bitstream settings button in the lower left hand corner. The option -bin_file should be selected. This ensures that the file generated will be compatible with the Basys3. This is shown in Figure 10.

Figure 10: .bin File settings



Write Bitstream (write_bitstream)

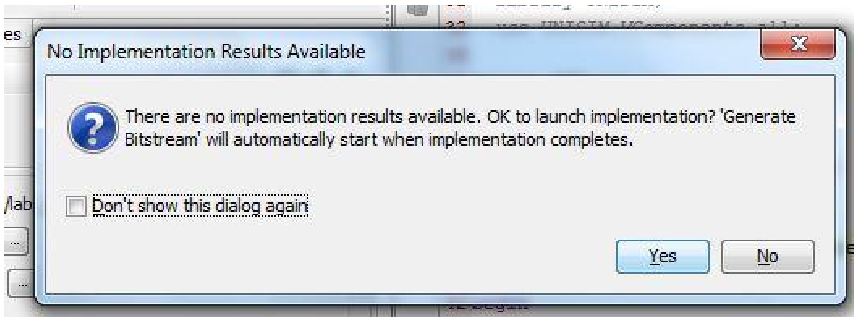
tcl.pre	
tcl.post	
-raw_bitfile	☐
-mask_file	☐
-no_binary_bitfile	☐
-bin_file*	☑
-readback_file	☐
-logic_location_file	☐
-verbose	☐
More Options	

Select an option above to see a description of it

Then, click the run Bitstream button in the lower left hand corner.

There will be a dialog box that says that there are no implementation results available (Figure 11).

Figure 11: Implementation

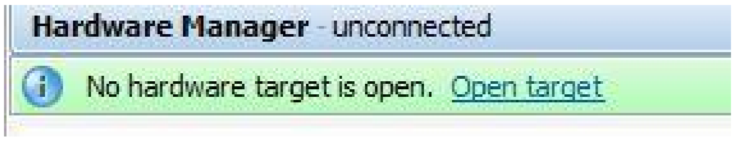

No Implementation Results Available
Don't show this dialog again
Yes
No

There are no implementation results available. OK to launch implementation? 'Generate Bitstream' will automatically start when implementation completes.

All this means is that the design has to be implemented before the bitstream can be produced. This process could take a few minutes. After the bitstream is generated the program must be loaded onto the Basys3.

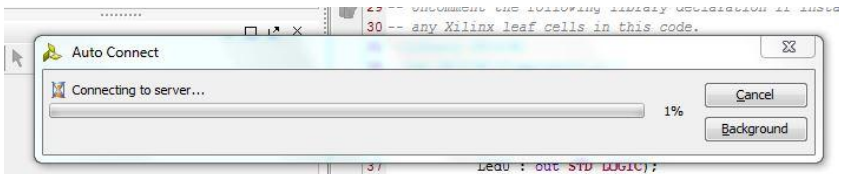
Next, open the Hardware Manager. There will be an option to “Open target’ (Figure 12) at the top of the screen. Click that and then click auto connect (Figure 13). This should automatically connect the Basys3 to the computer.

Figure 12: Hardware Manager


Hardware Manager - unconnected

No hardware target is open. Open target

Figure 13: Auto Connect.

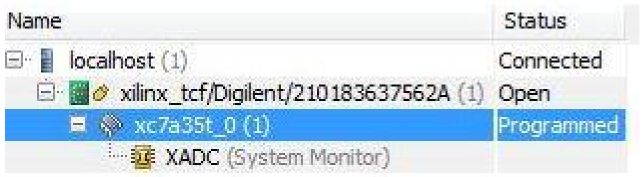

Auto Connect

Connecting to server...

1%

After these steps are completed the screen displayed in Figure 14 will be displayed on the screen. This confirms that the program has been loaded onto the Basys3.

Figure 14: Basys3 is programmed.



Name	Status
localhost (1)	Connected
xilinx_tcf/Digilent/210183637562A (1)	Open
xc7a35t_0 (1)	Programmed
XADC (System Monitor)	

Conclusion:

If everything is done correctly a NAND gate should be modeled on the Basys3. When both of the switches are turned to the on position (1) LED 0 should be turned off. Otherwise, the LED should light up to indicate the output is 1.

Appendix D
Lab 2
ENTC 3370
Logic Gates

Objective: The objective of this lab is to introduce the student to the basic logic gates.

Parts and Equipment:

1. 7408 Quad 2-input AND gate
2. 7432 Quad 2-input OR gate
3. 7404 Hex Inverter gate
4. Logic Trainer

Procedure:

Familiarize yourself with the digital trainer, Figure 1.

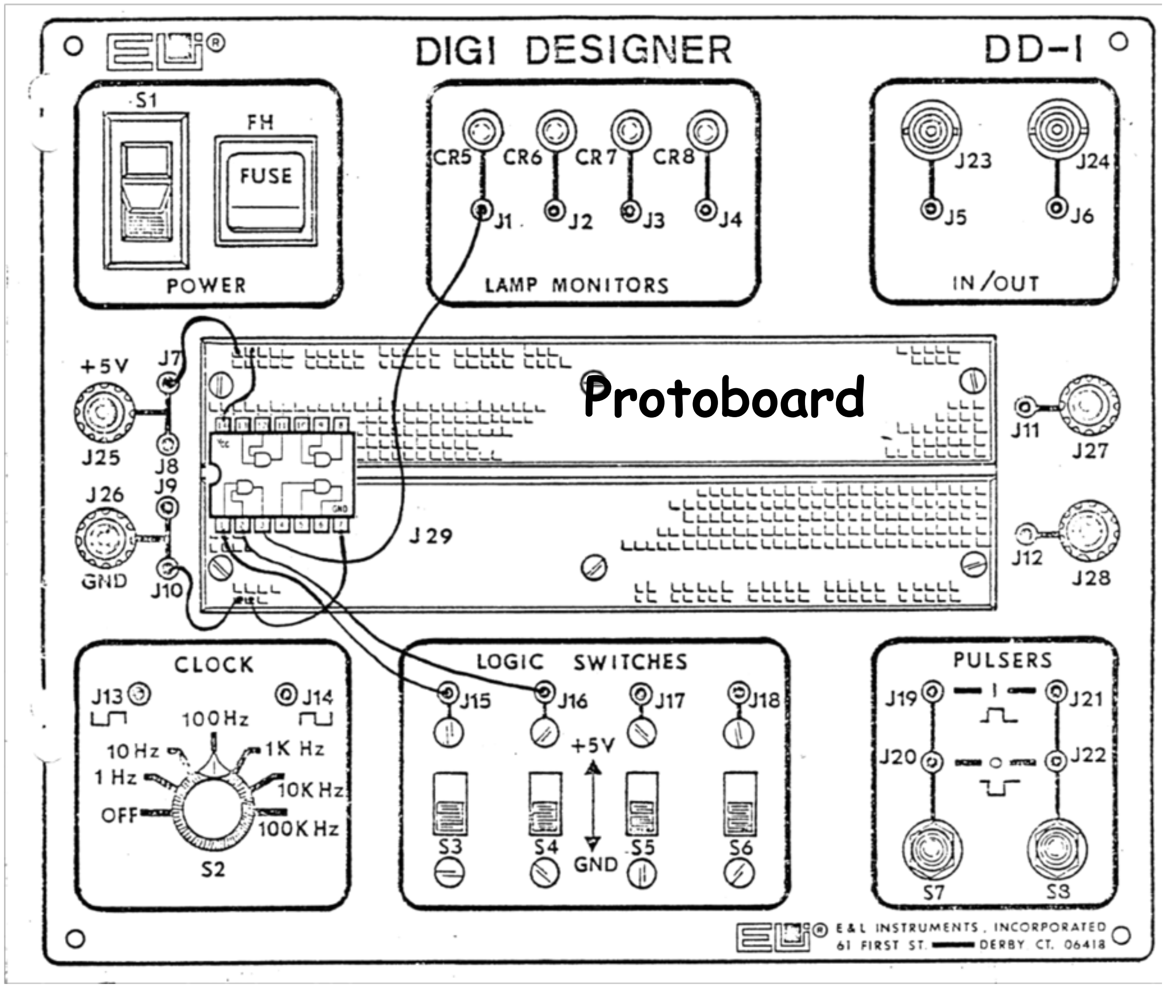


Figure 1. Logic Trainer.

Note that the protoboard is connected as follows.

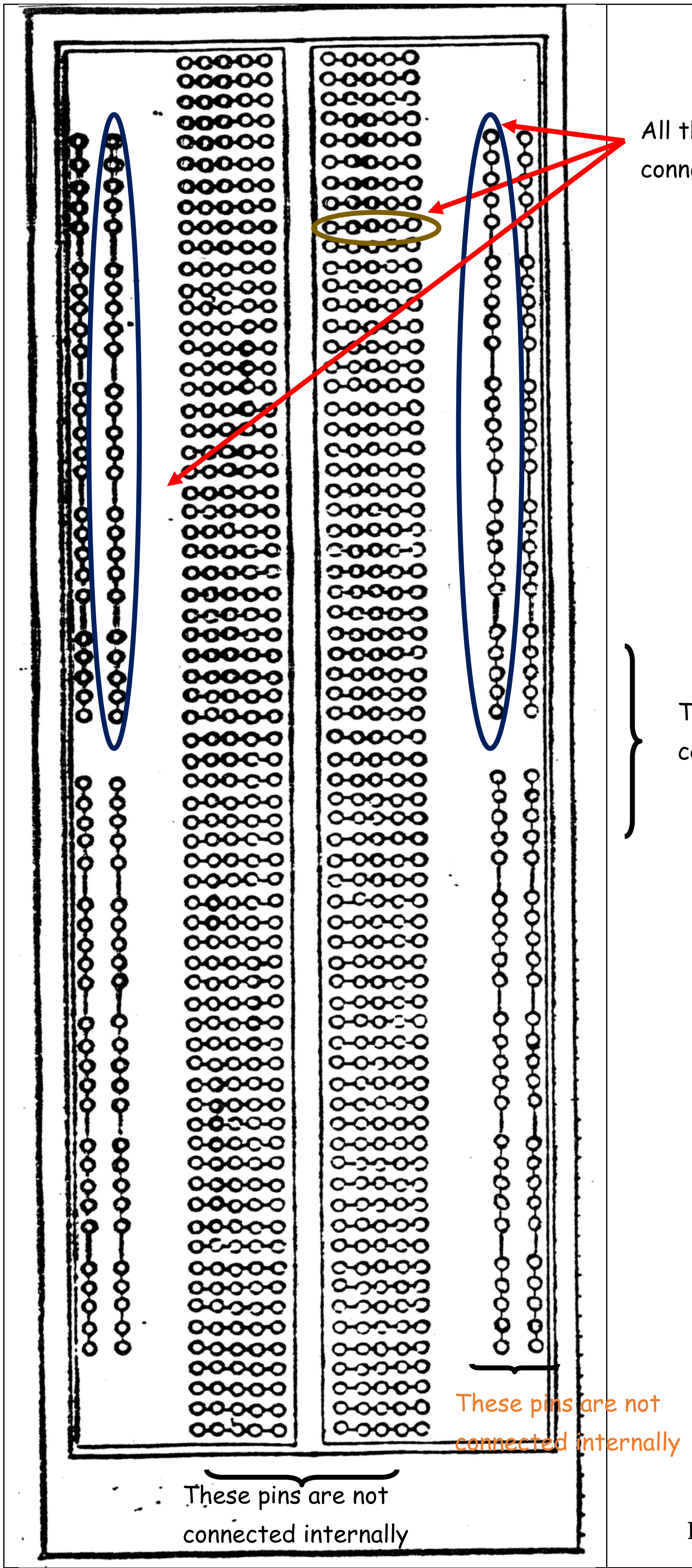
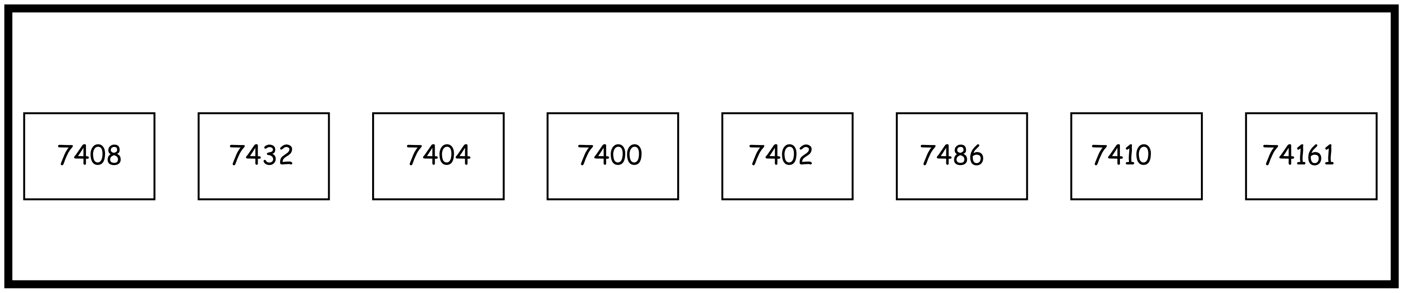

All these pins are connected internally
These pins are not connected internally
These pins are not connected internally
These pins are not connected internally

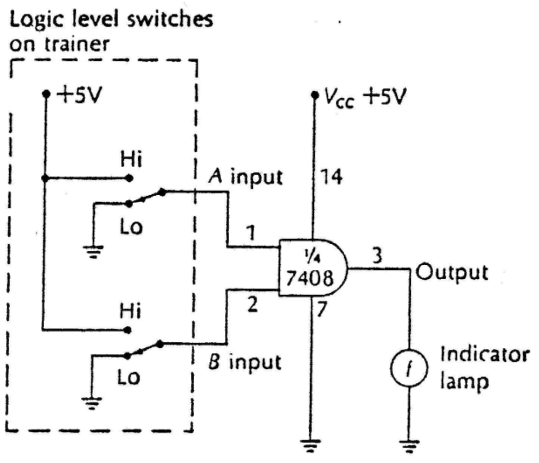
Figure 2. Protoboard.

The protoboards are prepopulated with the following gates:

Figure 3. Protoboard polpulation.



Given a quad two-input AND gate chip, the 7408, construct the circuit shown below in Figure 4 and verify the truth table for an AND gate.



Circuit of Experiment 1-1, the AND Gate (Boolean equation: $f = A \cdot B$)

Figure 4. AND Circuit.

Note that we are using gate A which has one input at pin 1 (1A) and one input at pin 2 (1B). The output (1Y) is take off of pin 3, Figure 5.

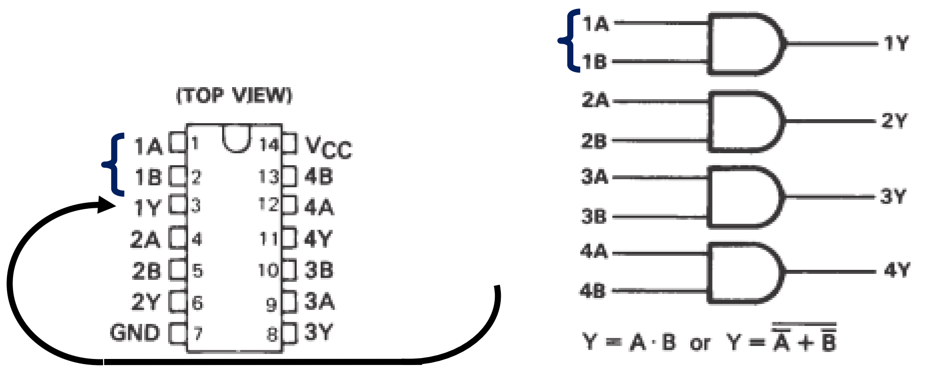


Figure 5. AND hook up.

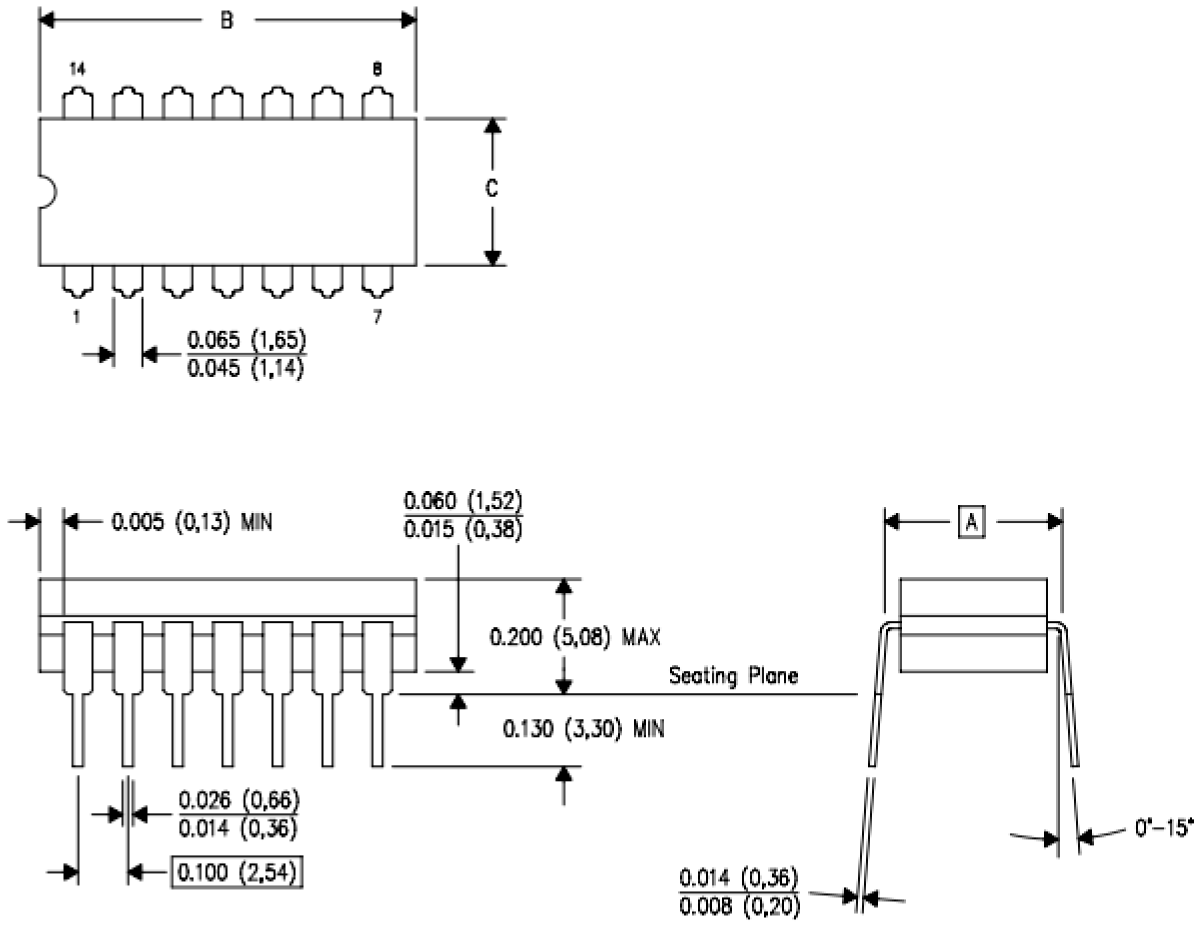


Figure 6. 7408 physical dimensions.

Given a quad two-input OR gate chip, the 7432, construct the circuit shown below in Figure 7 and verify the truth table for an OR gate.

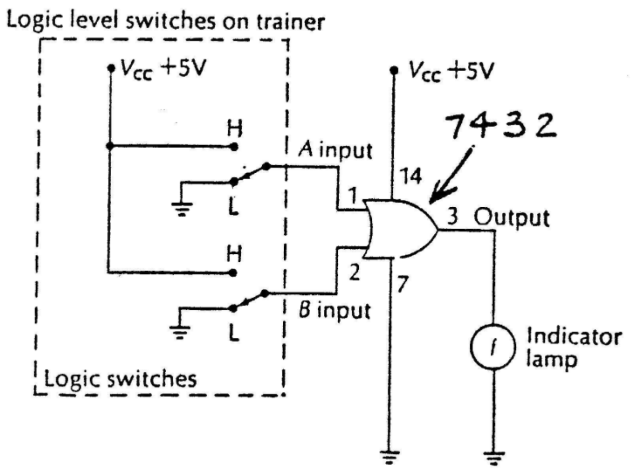


Figure 7. Protoboard.

Circuit for Evaluating the OR Gate
(Boolean equation: $f = A+B$, read $f = A$ OR B)

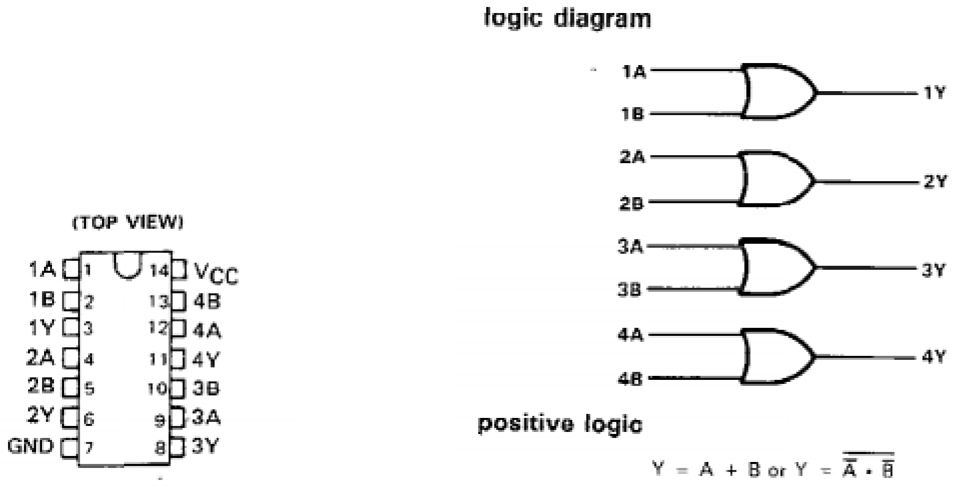
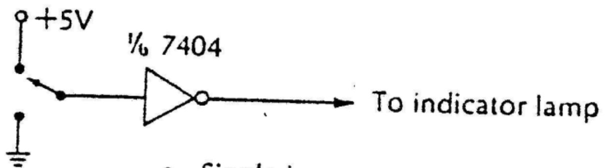
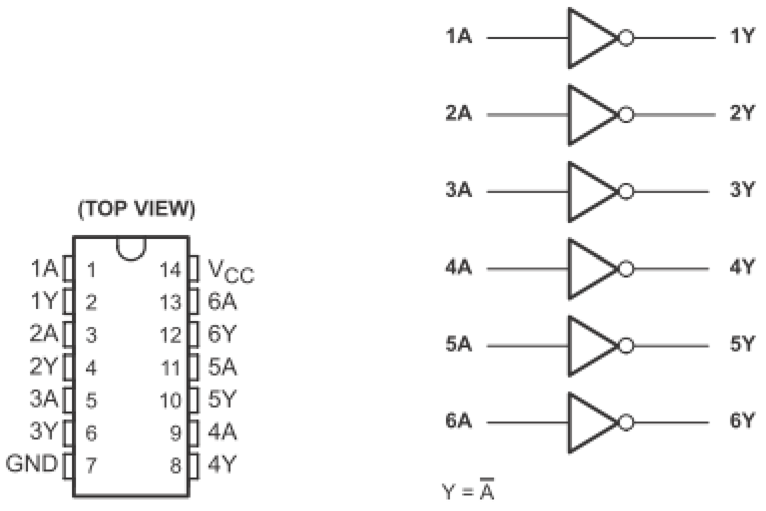


Figure 8. OR hook up.

Given a hex inverter gate chip, the 7404, construct the circuit shown below in Figure 9 and verify the truth table for an OR gate.



a. Single inverter



Explain how these gates compare to the theory and your expectations.