FPGA Introduction Lab

Bret Comnes and Andres La Rosa

Abstract

This lab explores Field Programmable Gate Arrays (FPGA). A short background discussion of FPGAs is followed by a tutorial section covering a basic workflow used to configure an FPGA development board. It includes FPGA design using schematic layouts, as well as an example of how to use VHDL (programming language) to implement a very basic digital circuit. Students will be exposed to the processes used to design and simulate an FPGA configuration as well as compile their design and see it run on an actual FPGA. The lab is designed around the Digilent Nexys 3 prototyping board, which provides relatively easy access to a Xilinx Spartan6 XC6LX16-CS324 FPGA chip with about 2 million logic gates, a programmer, program memory, and array of switches, buttons, LEDs and I/O ports.

Glossary:

IDE	Integrated Development Environment
IC	Integrated Circuit
IP	Intellectual Property (Used in reference to premade FPGA design modules).
IDE	Integrated Development Environment
FPGA	Field Programmable Gate Array
VHSIC	Very High Speed Integrated Circuits
HDL	Hardware Description Languages
VHDL	VHSIC Hardware Description Language
Verilog	Another HDL which is an alternative to VHDL
ISE	Integrated software environment. ISE is a GUI accessible tool provided by XILINX to create designs for its FPGA chips.
GUI	Graphical User Interface

Table of Contents

1 FPGA Introduction

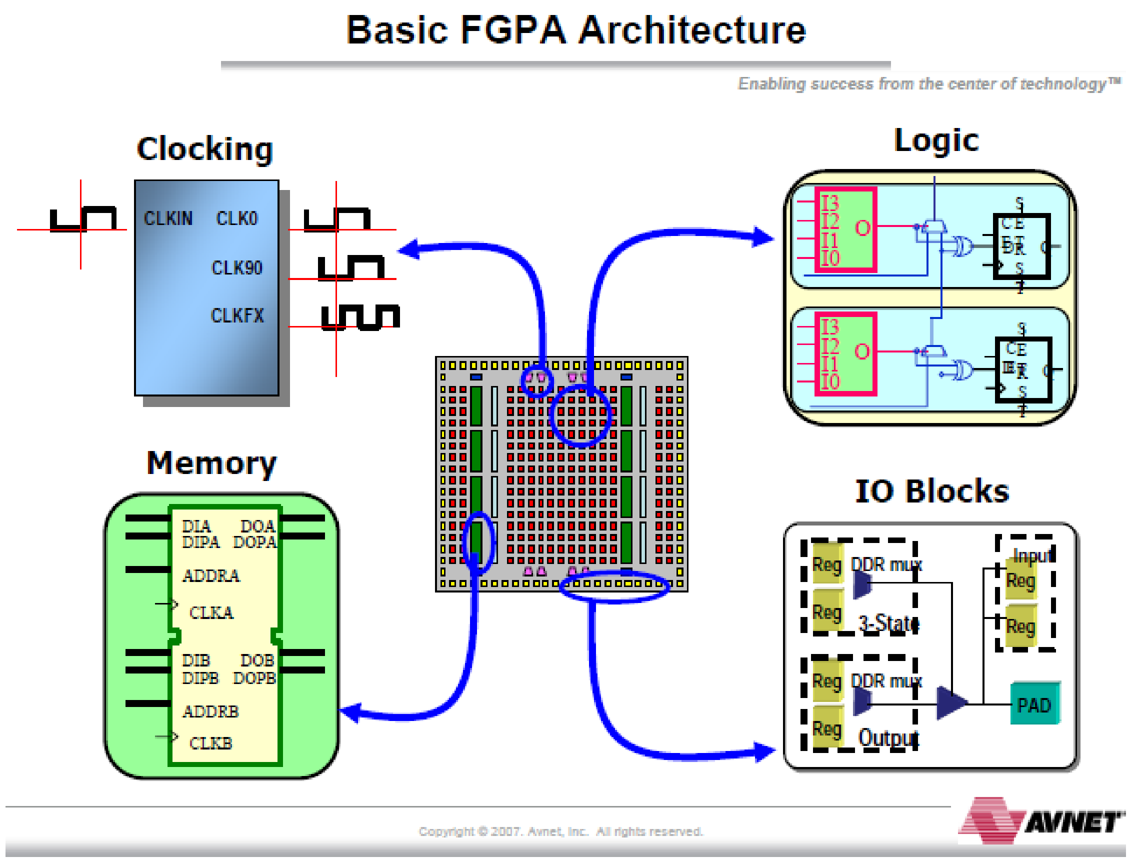
1.1 What is an FPGA?

Field Programmable Gate Arrays (FPGAs) are semiconductor devices constituted by a matrix of configurable logic blocks (CLBs) connected via programmable interconnects. The FPGAs large arrays of generic logic gates, all inside of a single integrated circuit (IC) that can be reprogrammable arbitrarily (!) to the desired application or functionality requirement after manufacturing (!). Typically, the number of logic gates in an FPGA is on the order of magnitude of millions.

The FPGA is typically configured using a Hardware Description Language (HDL), making it possible for FPGA designs to be portable between hardware, so long as the correct constraints are used to map the design to the specific details of the FPGA that is being used. There are also other methods to configure the FPGA such as schematic layout interpreters, as well as proprietary solutions such as LabVIEW.

It is also common to use design modules, which are premade FPGA designs in addition to HDL designs. These modules often referred to as cores, come as open design cores, or IP modules that prevent you from looking inside their code to see how they work. Many of these logic cores are specific to particular FPGAs, or brands of FPGAs.

Not only can FPGAs be configured into any arbitrary digital circuit, they can also be reconfigured at a later time if, for example, you need to fix a problem in the circuit after you have integrated it into a project. It can even reconfigure itself to act as a different circuit depending on external conditions.



http://www.avnet-israel.co.il/download/downloadPresentations/Presentations/xfest07_GD.pdf

1.2 How FPGAs differ from other embedded systems

There is a large array of programmable logic (PL) technologies available, and at this point it might even seem like FPGAs are a glorified microcontroller. This is not so. When you configure an FPGA, the logic gates inside the IC are configured correctly such that you are actually physically creating the circuit that you have specified, all inside the FPGA chip.

Also, unlike some PL, the FPGA can be rewritten as many times as needed. The flash memory, which stores the program to configure the FPGA on power up, will be the limiting factor, with a re-write limit of about 100,000.

Many FPGAs have an array of IP (Intellectual Property) modules which are premade FPGA configurations that can perform a complicated task. For example there are IP modules to implement a soft CPU in your FPGA so that it can be used as a more general purpose computer. These modules typically cost money and include some form of DRM or obfuscation to keep you from seeing how they work. The whole purpose of not just simply sharing the HDL code is to prevent others from seeing how the developer implemented a particular design for whatever reason.

1.3 FPGA Advantages

When implementing complex digital circuits, FPGAs provide numerous advantages. The design can be written, tested and simulated on the computer. The designs can be done in a way so that they are portable to other FPGA devices, for repeatable and rapid deployment in many devices. Multiple people can work on the same HDL files and increase the speed of circuit development.

Once the design is ready for implementation, the entire production process is uploading the design to the board, as the FPGA will configure the circuit for you, rather than having to build it with a machine or by hand. If there is ever an issue with the circuit that is discovered after deployment, it is only a matter up updating the HDL code and re-uploading it to the FPGA. The FPGA could even check online for schematic updates potentially.

Running the circuit in a single chip also allows for high speed circuitry that would simply be unattainable on a breadboard or other circuit design methods. When used correctly, FPGAs can be extremely fast, which is typically why they are used over other options. FPGAs fall into the category of Real Time devices.

1.4 FPGA Disadvantages

FPGAs have been historically quite expensive. Additionally, they can be fairly difficult to work with, requiring a wide range of knowledge of HDLs and digital circuit design to use effectively, requiring large amounts of training to effectively use.

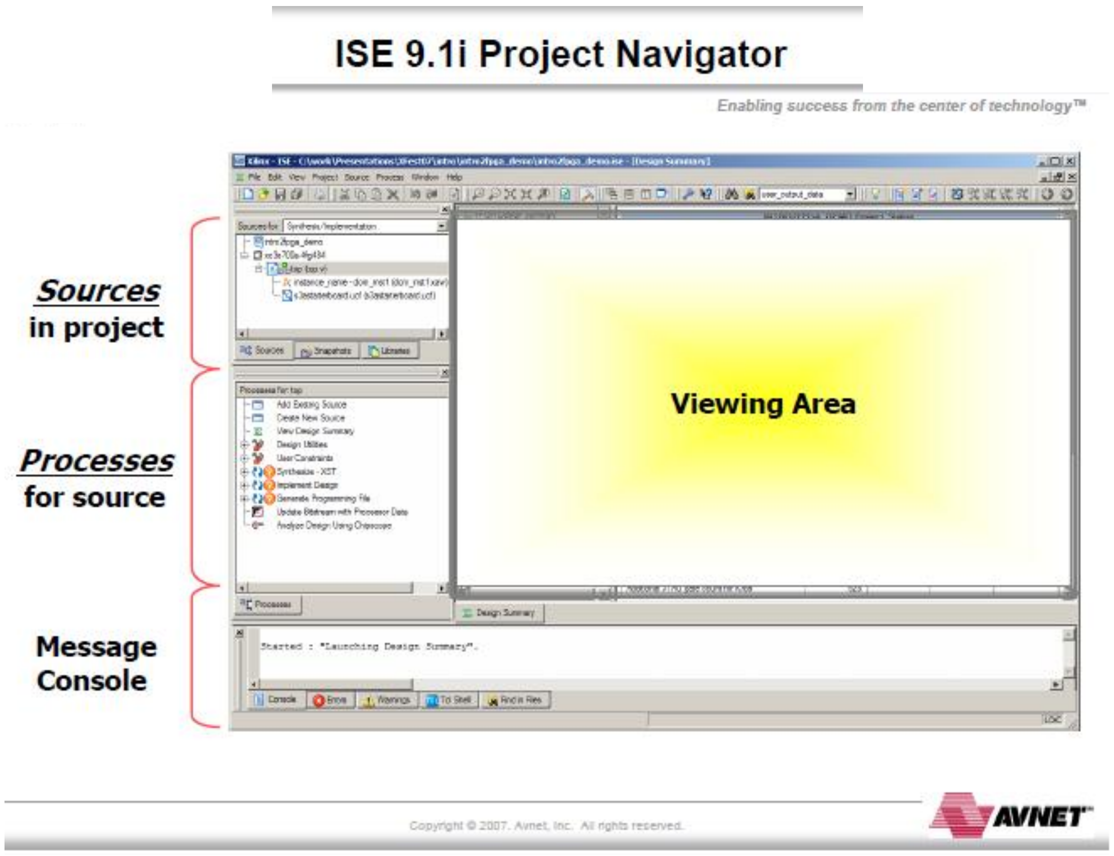
The software used to work with many FPGAs can be quite expensive, and as well as bulky in size. The IDE we are using is more than 8Gb in download size, and as you will learn, has a user interface that will take some time getting used too.

2 Working with the FPGA

2.1 Workflow overview

For working with our Nexys 3 FPGA, we will be using the **ISE Project Navigator**. (ISE is a GUI accessible tool provided by XILINX to configure its FPGA The GUI is called Project Navigator).
This integrated development environment (IDE) allows setting up a project file, which contains our designs, test and constraint files. It also acts as a debugger and handles the compiling of our designs.

Once we have design, tested, and compiled our FPGA configuration, we will need to upload it to the board. This is accomplished using a tool called **Adept**, which is made by Digilent, the company who made our FPGA development board. This program is used to upload .bit files to many different Digilent FPGA boards.


ISE 9.1i Project Navigator
Enabling success from the center of technology™
Sources in project
Processes for source
Message Console
Viewing Area
Copyright © 2007. Avnet, Inc. All rights reserved.
AVNET

```
Started : "Launching Design Summary".
```

http://www.avnet-israel.co.il/download/downloadPresentations/Presentations/xfest07_GD.pdf

2.2 Development options

There are a number of different options for configuring an FPGA, such as HDLs, schematics or proprietary solutions. We will focus on schematics and an HDL called

VHDL. First we will turn our FPGA into an AND gate using a schematic, and then we will do the same thing using VHDL.

2.3 Hardware Description Languages

FPGA designs are usually made with an HDL such as **VHDL** or **Verilog**. It is easier to maintain code rather than a 2-D schematic, which is why it HDLs are used more often than schematic designs in FPGAs.

These languages are similar to a programming language, but are rather geared towards digital circuit design, so they have the same limitations as digital circuits. For this lab, these differences will not be of focus. Understanding these differences would be important if you were actually to use an FPGA in a more complex project.

We are going to configure the FPGA first using schematics (Section 3) and then using VHDL code (Section 4). [For the latter you may complement the information checking the following site http://www.youtube.com/watch?v=Ob7B6x5g6tw How to program in VHDL: Making a counter.]

3 Configuring the FPGA using schematics

In this section we will implement a simple AND gate, test that it works using a simulation, add constraints to hook up two onboard switches and an LED to the schematic file and then deploy our design to the actual FPGA using the Digilent Adept software.

Find a computer

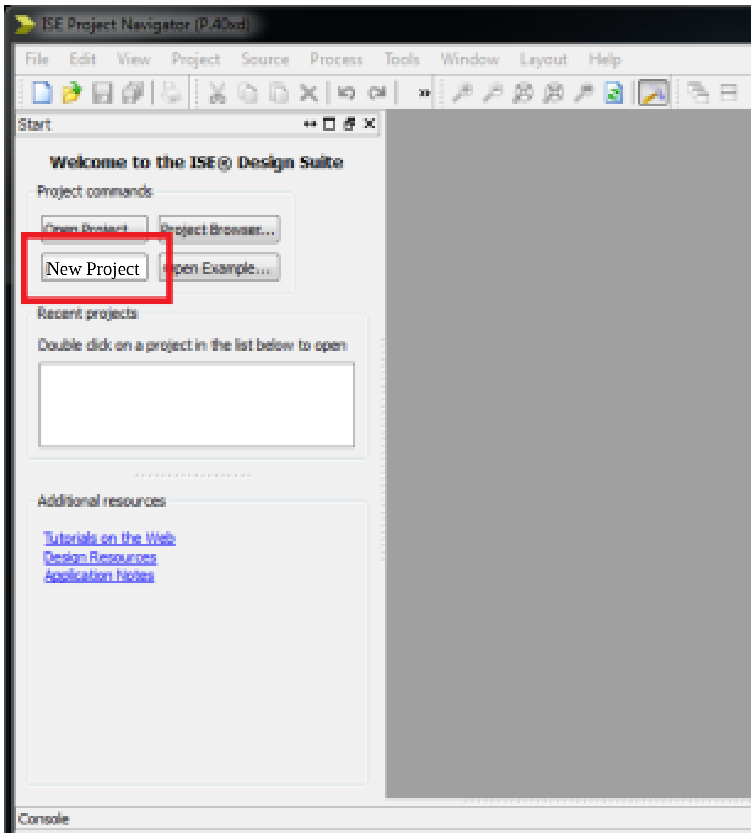
You must use one of the computers in SB1-201, since they are the only ones with the licensed software already installed.

Open ISE Project Navigator

There should be a desktop icon, otherwise search through the Start menu to find it.

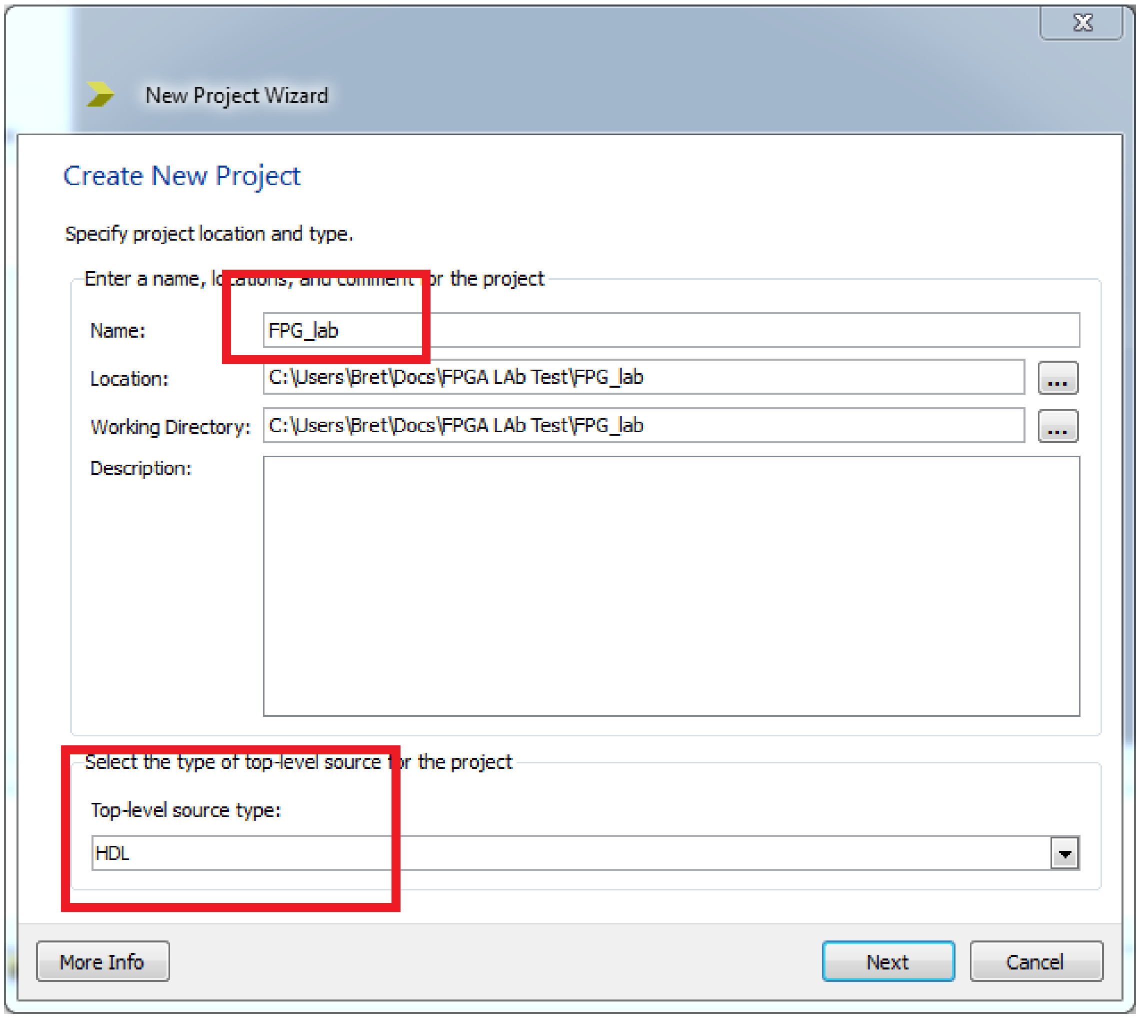
Create a new project

Go to File-> New Project


ISE Project Navigator (P.40xd)
File Edit View Project Source Process Tools Window Layout Help
Start
Welcome to the ISE® Design Suite
Project commands
Open Project...
Project Browser...
New Project
Open Example...
Recent projects
Double click on a project in the list below to open
Additional resources
Tutorials on the Web
Design Resources
Application Notes
Console

Give your project a new name. **Don't use any spaces or special characters**. Use an underscore (_) if you need to put a space in your file name. Save it to your thumb drive or user directory. The ISE Project explorer will make a royal mess inside this file, so do save it into a folder with files that already exist.

Also, under ***top-level source type***, choose HDL


New Project Wizard
Create New Project
Specify project location and type.
Enter a name, locations, and comment for the project
Select the type of top-level source for the project
More Info
Next
Cancel

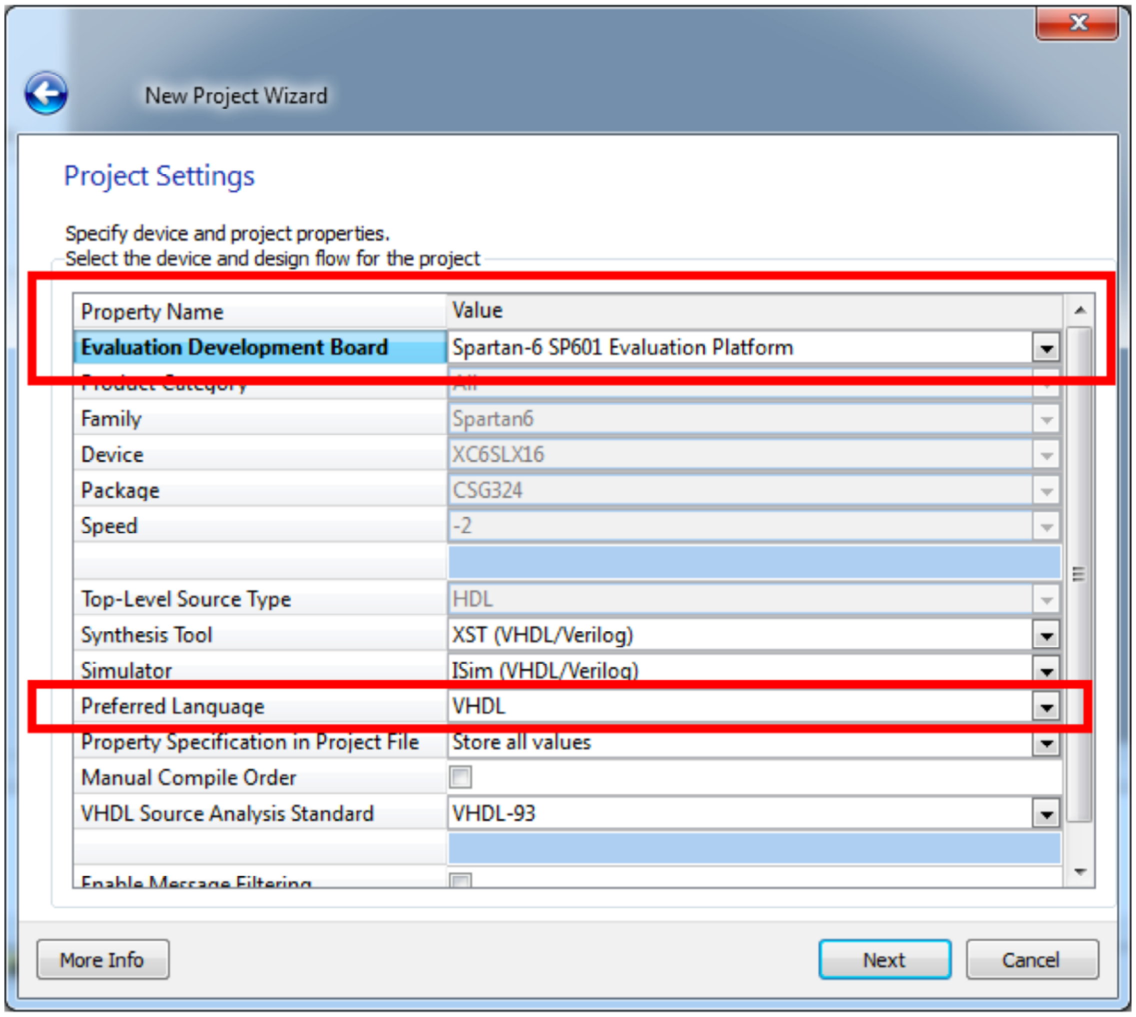
Name: FPG_lab
Location: C:\Users\Bret\Docs\FPGA LAb Test\FPG_lab
Working Directory: C:\Users\Bret\Docs\FPGA LAb Test\FPG_lab
Description:
Top-level source type: HDL

Click Next.

Selecting the board

Under *Evaluation Development Board*, choose Spartan 6 SP601 Evaluation Platform. This will set the category, family, speed and package options once you select our board. Watch out not to select the other Spartan6 option.

Choose VHDL under the ***Preferred Language*** setting



Project Settings

Specify device and project properties.
Select the device and design flow for the project

Property Name	Value
Evaluation Development Board	Spartan-6 SP601 Evaluation Platform
Product Category	All
Family	Spartan6
Device	XC6SLX16
Package	CSG324
Speed	-2
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	VHDL
Property Specification in Project File	Store all values
Manual Compile Order	☐
VHDL Source Analysis Standard	VHDL-93
Enable Message Filtering	☐

Click next.

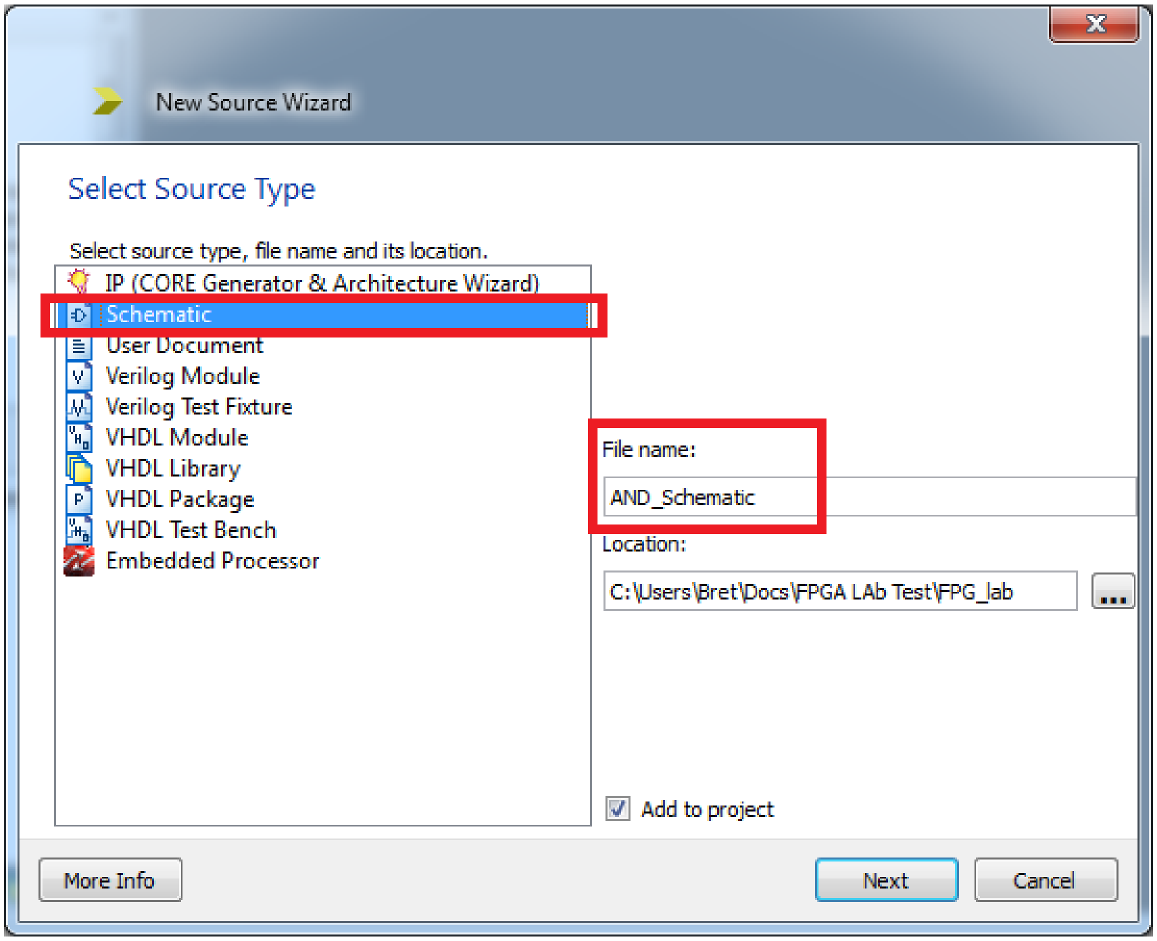
A project summary window will show. Check for any oddities and click finish to close the window.

This closes the new project wizard, and drops you into your ISE project navigator IDE. It is a very non-intuitive interface that you will only become accustom to with practice.

Creating new files using the ISE project navigator

Next to the File menu, **Select Project -> New Source** from the drop down menu. This will open the *New Source Wizard*.

Within the *New Source Wizard*, select **Schematic** and give it a name. Again, do not include special characters or spaces in the name. The ISE is a whole hodgepodge of tools with known file naming bugs if you go to far out of the naming scope.



New Source Wizard

Select Source Type

Select source type, file name and its location.

- IP (CORE Generator & Architecture Wizard)
- Schematic
- User Document
- Verilog Module
- Verilog Test Fixture
- VHDL Module
- VHDL Library
- VHDL Package
- VHDL Test Bench
- Embedded Processor

File name: AND_Schematic

Location: C:\Users\Bret\Docs\FPGA LAb Test\FPG_lab

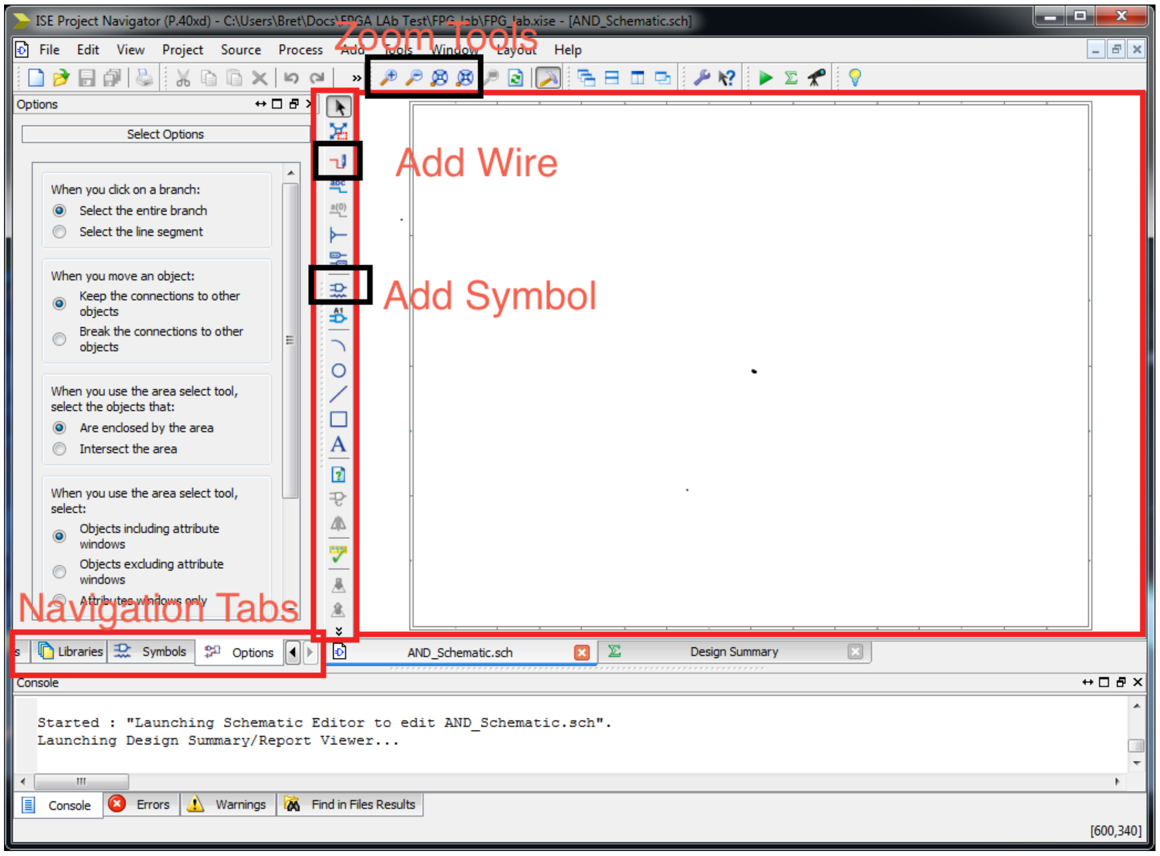
☑ Add to project

Continue through the wizard and check for oddities in the summary window

Once the new source wizard closes, your schematic will automatically open.

Notice that the vertical toolbar next to the schematic window has now changed. This vertical toolbar is where you find your context specific tool buttons. To the right of this vertical toolbar is where the file you are working on is displayed. To the left of the toolbar, is your project navigator, plus your toolbox for some of the tools on the toolbar.

The UI is really random, so just learn to deal with it. There is a whole slew of tabs, on this left hand side, but the **Design** tab is the most useful. It acts as your project navigator and task launcher for compiling and testing your code and schematics. The **Design** tab has two modes, selected by the radio buttons at the top: a **Simulation** mode and **Implementation** mode. We will be switching between the two.


ISE Project Navigator (P.40xd) - C:\Users\Bret\Docs\FPGA LAb Test\FPG lab\FPG lab.xise - [AND_Schematic.sch]
File Edit View Project Source Process Add Tools Window Layout Help
Zoom Tools
Add Wire
Add Symbol
Navigation Tabs
Options
Select Options
Libraries
Symbols
Options
AND_Schematic.sch
Design Summary
Console
Errors
Warnings
Find in Files Results
[600,340]

When you click on a branch:
- [✓ Select the entire branch]
- Select the line segment

When you move an object:
- [✓ Keep the connections to other objects]
- Break the connections to other objects

When you use the area select tool, select the objects that:
- [✓ Are enclosed by the area]
- Intersect the area

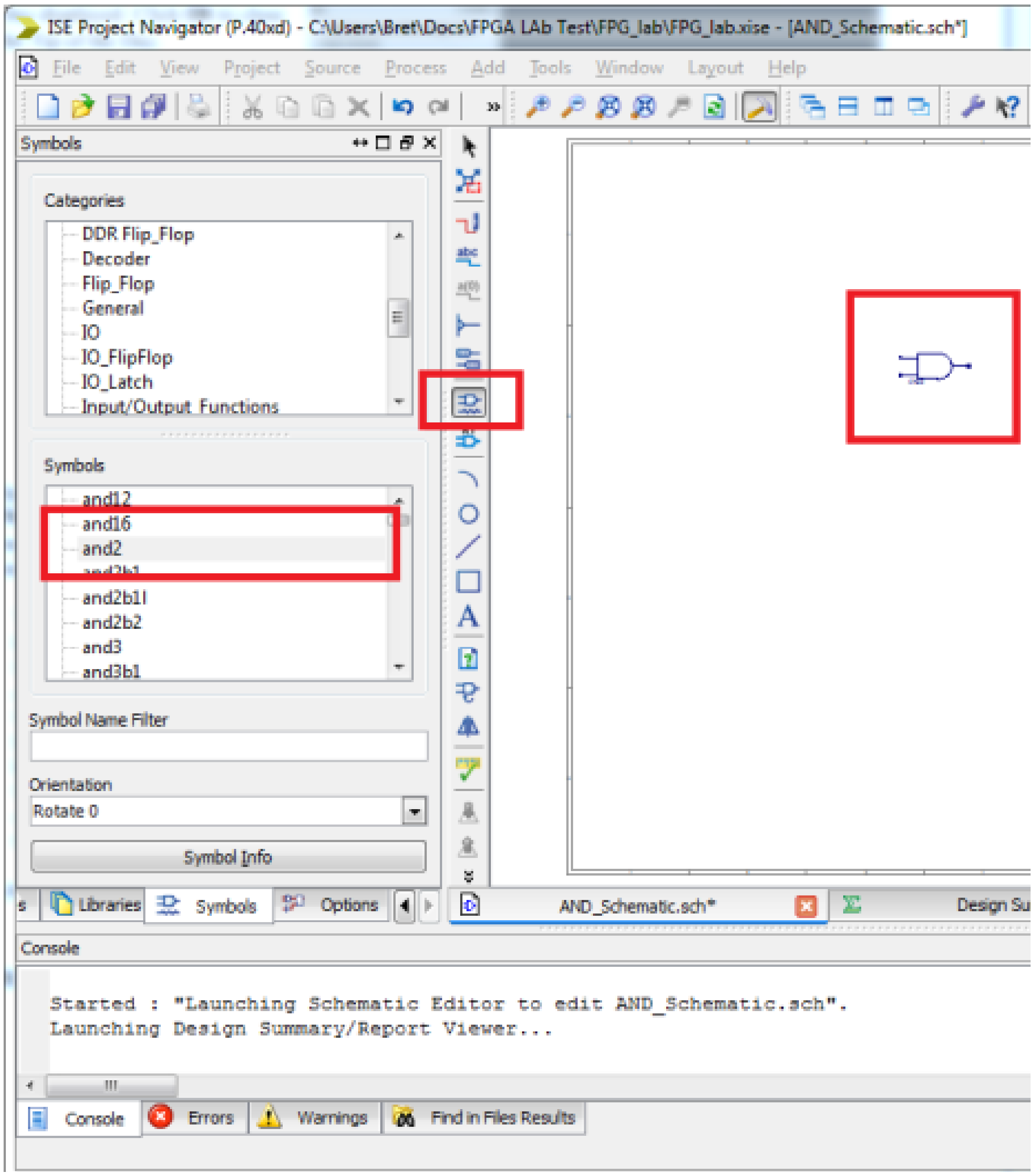
When you use the area select tool, select:
- [✓ Objects including attribute windows]
- Objects excluding attribute windows
- Attributes windows only

```
Started : "Launching Schematic Editor to edit AND_Schematic.sch".
Launching Design Summary/Report Viewer...
```

Adding digital electronic components

Click the "Add Symbol" vertical Toolbar button.

This opens a menu to the left of the toolbar with digital electronic components. They are sorted into categories for somewhat quick access, but the ***Symbol Name Filter*** is the quickest way to find what you are looking for if you know the name.



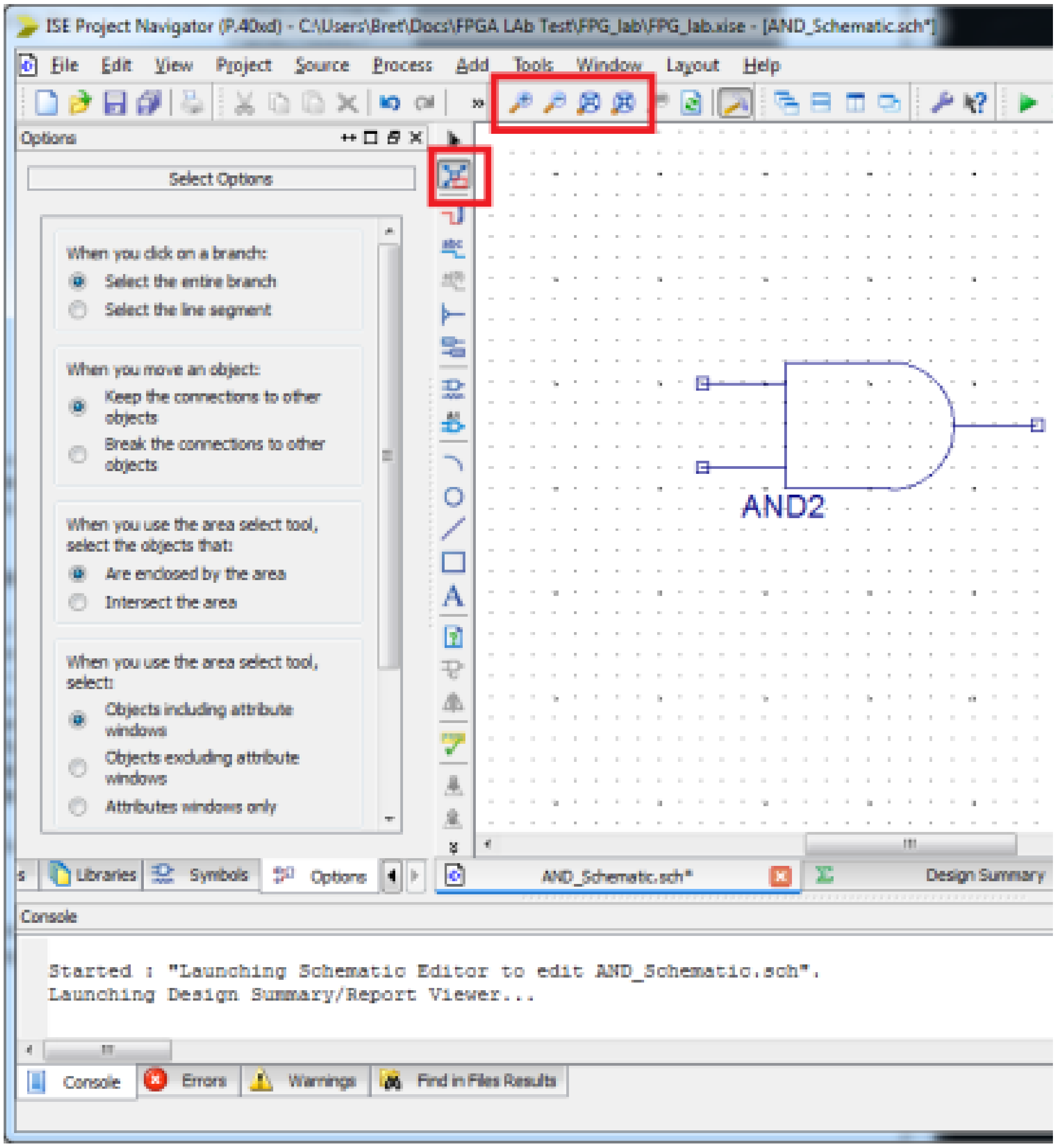
```
Started : "Launching Schematic Editor to edit AND_Schematic.sch".
Launching Design Summary/Report Viewer...
```

Click on the **logic** category
Notice the symbols list shortened to only show the logic components in alphabetical order.

Choose the *AND2* symbol from the symbol menu.
Now when you roll over the schematic view, you get something that looks like that component attached to your cursor. This lets you know what symbol you are holding and ready to place on your schematic.

Place one of these AND2 gates.

Place by hovering to where you want to place them, and clicking once. Press escape to clear your placement selection and go back to mouse mode.


ISE Project Navigator (P.40xd) - C:\Users\Bret\Docs\FPGA LAb Test\FPG_lab\FPG_lab.xise - [AND_Schematic.sch*]
File Edit View Project Source Process Add Tools Window Layout Help
Options
Select Options
AND2
Libraries
Symbols
Options
AND_Schematic.sch*
Design Summary
Console
Errors
Warnings
Find in Files Results

When you click on a branch:
- [✓ Select the entire branch]
- Select the line segment

When you move an object:
- [✓ Keep the connections to other objects]
- Break the connections to other objects

When you use the area select tool, select the objects that:
- [✓ Are enclosed by the area]
- Intersect the area

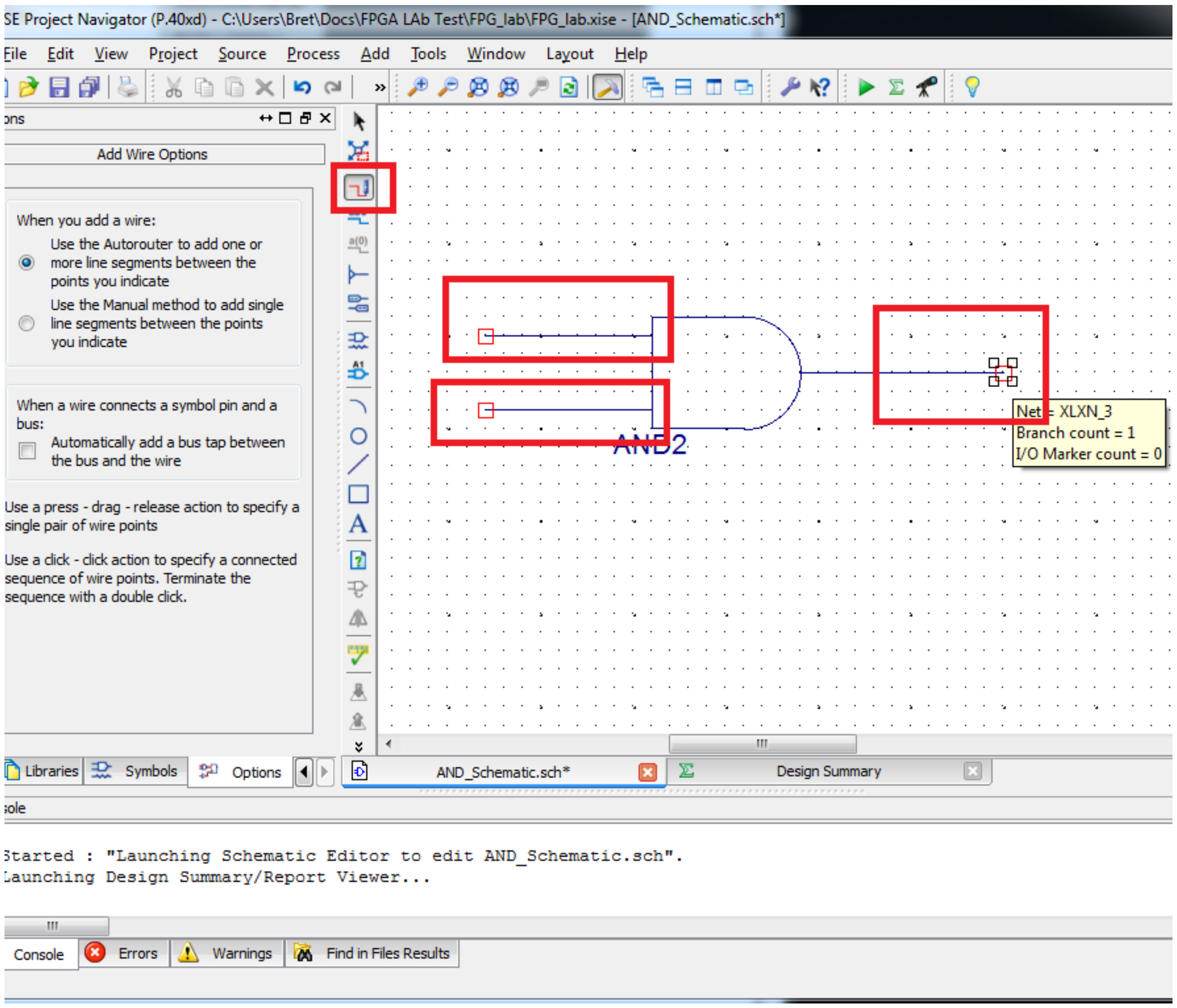
When you use the area select tool, select:
- [✓ Objects including attribute windows]
- Objects excluding attribute windows
- Attributes windows only

```
Started : "Launching Schematic Editor to edit AND_Schematic.sch".
Launching Design Summary/Report Viewer...
```

Use the zoom tools to zoom in on your parts.

Use the add wire tool in the vertical toolbar to add short wires to the three leads on your and gate.

Clicking on the terminal on the AND gate, then clicking where you want the wire to go drops a wire segment. Double clicking will end the wire placement. Press escape to end wire placement all together and clear your tool.


SE Project Navigator (P.40xd) - C:\Users\Bret\Docs\FPGA LAb Test\FPG_lab\FPG_lab.xise - [AND_Schematic.sch*]
File Edit View Project Source Process Add Tools Window Layout Help
ons
Add Wire Options
AND2
Libraries Symbols Options
AND_Schematic.sch*
Design Summary
sole
Console Errors Warnings Find in Files Results

When you add a wire:
- [✓ Use the Autorouter to add one or more line segments between the points you indicate]
- Use the Manual method to add single line segments between the points you indicate

When a wire connects a symbol pin and a bus:
- [ ] Automatically add a bus tap between the bus and the wire

Use a press - drag - release action to specify a single pair of wire points

Use a click - click action to specify a connected sequence of wire points. Terminate the sequence with a double click.

Net = XLXN_3
Branch count = 1
I/O Marker count = 0

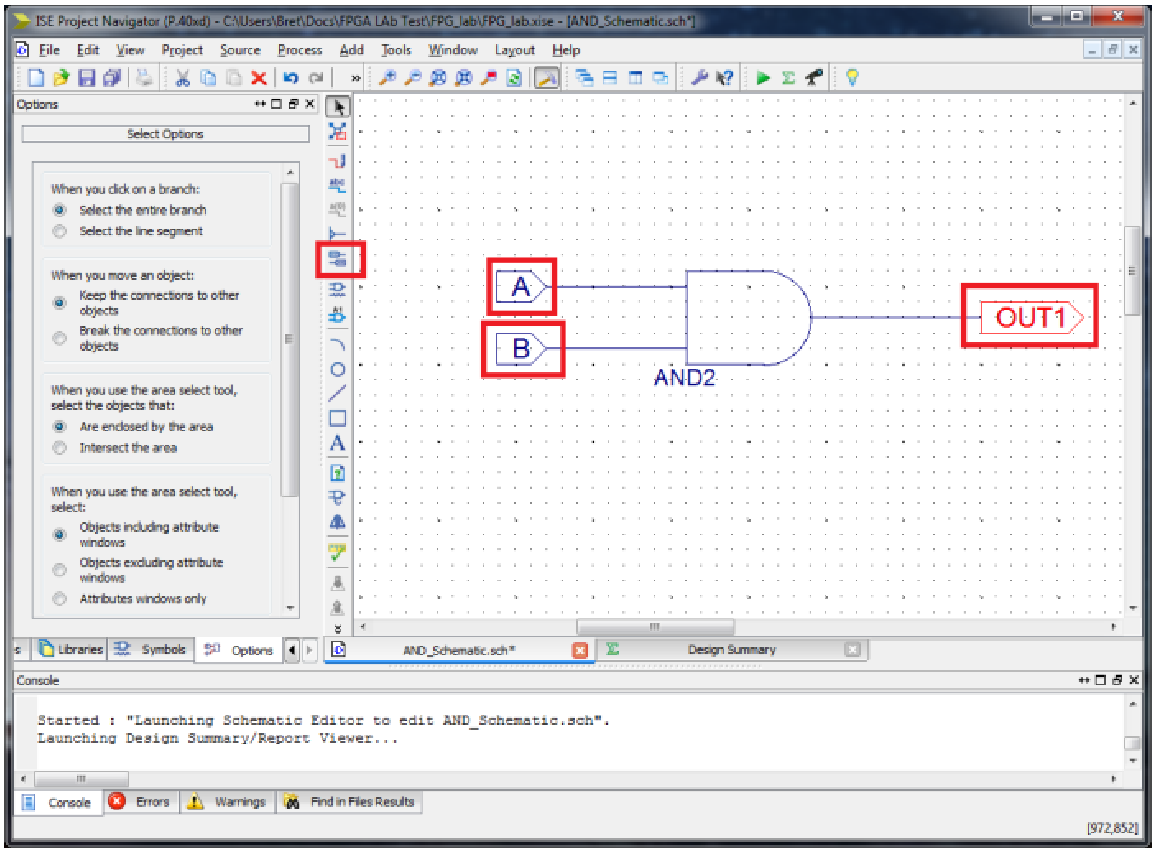
```
Started : "Launching Schematic Editor to edit AND_Schematic.sch".
Launching Design Summary/Report Viewer...
```

Adding IO markers

These markers give names to the inputs and outputs of your design. They reference an input out output on your design, but have nothing to do with the physical FPGA board yet. They are purely apart of your circuit design. You refer to these when you constrain your design to the actual FPGA hardware.

Click the add IO button in the vertical tool bar.
Click all 3 terminal leads to drop a marker. Right click to rename each port. Name the inputs A and B and the output OUT1.

Check your schematic for errors in **Tools -> Check schematic**.
Errors show up in the console at the bottom. Fix any errors.


ISE Project Navigator (P.40xd) - C:\Users\Bret\Docs\FPGA LAb Test\FPG_lab\FPG_lab.xise - [AND_Schematic.sch*]
File Edit View Project Source Process Add Tools Window Layout Help
Options
Select Options
A
B
AND2
OUT1
Libraries Symbols Options
AND_Schematic.sch*
Design Summary
Console
Console Errors Warnings Find in Files Results
[972,852]

When you click on a branch:
[✓ Select the entire branch]
Select the line segment

When you move an object:
[✓ Keep the connections to other objects]
Break the connections to other objects

When you use the area select tool, select the objects that:
[✓ Are enclosed by the area]
Intersect the area

When you use the area select tool, select:
[✓ Objects including attribute windows]
Objects excluding attribute windows
Attributes windows only

```
Started : "Launching Schematic Editor to edit AND_Schematic.sch".
Launching Design Summary/Report Viewer...
```

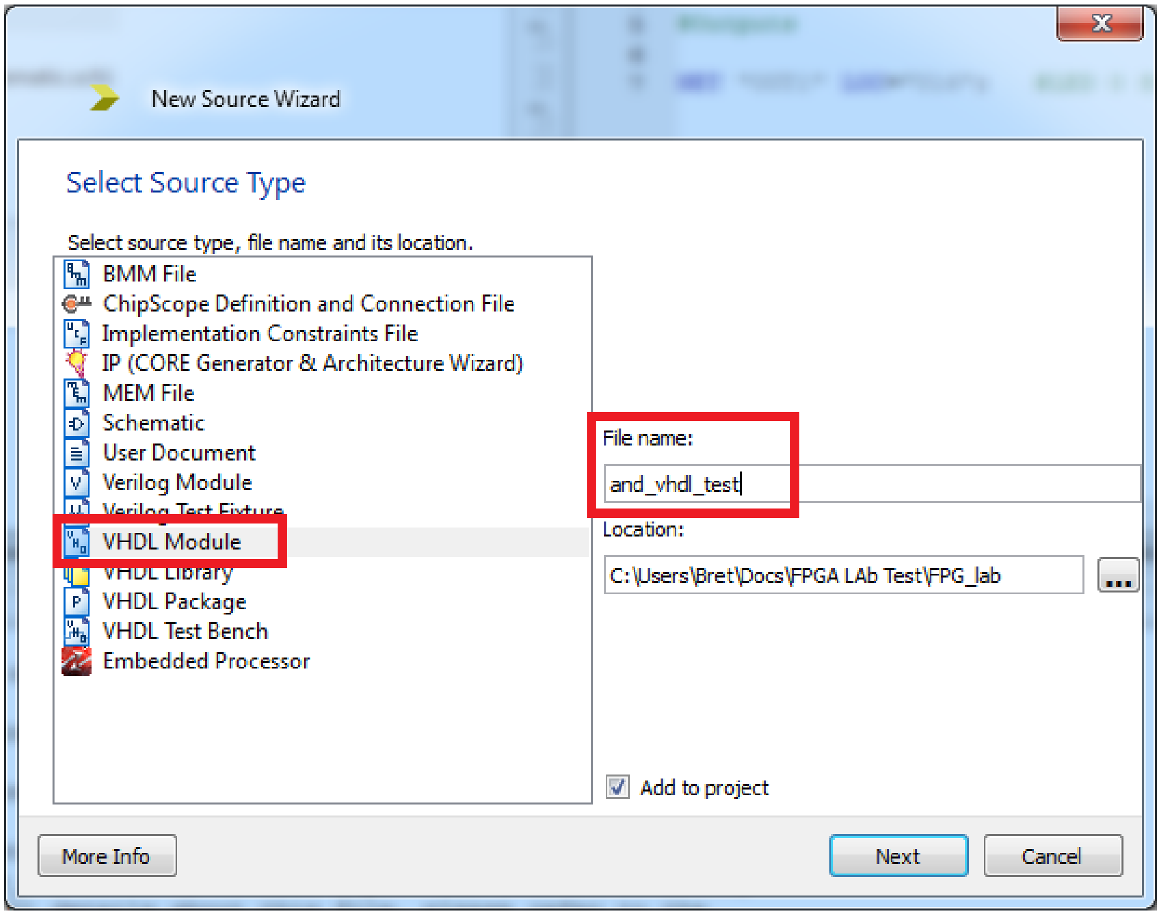
Save your work. File -> Save all

Running test procedures

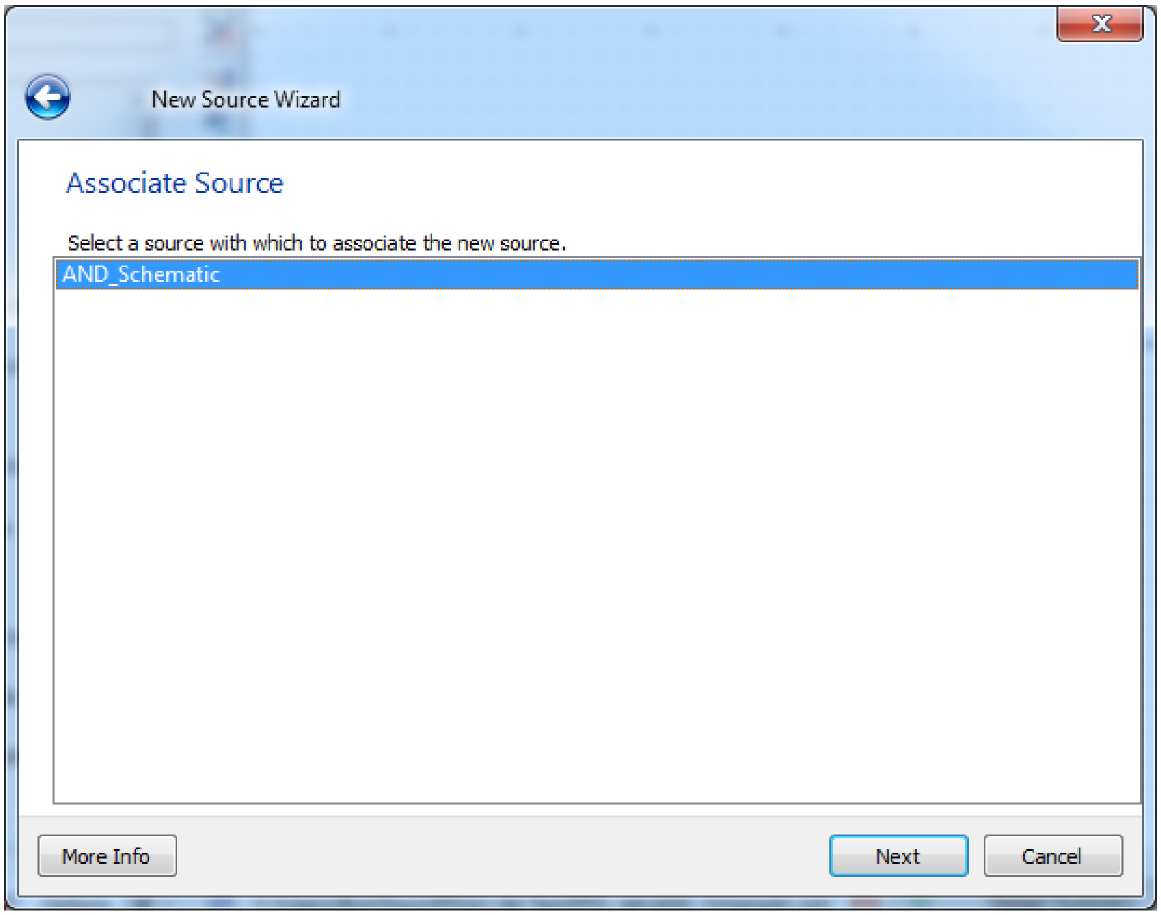
Now we test our schematic to see if it works. We will be running a simulation in our computer to test the logic, to make sure we didn't make any mistakes. This is faster than compiling and uploading to the board, and it is very powerful to be able to test the circuit without actually having to test it on hardware yet.

Testing a design requires writing a VHDL test bench. Even though we designed out circuit using a schematic, writing the test requires that we use VHDL to define some test inputs.

Go to Project -> Add Source
Select VHDL Test Bench
Give it a name and press next.



The next window will ask you to select which VHDL file or schematic to test. You only have one (AND_schematic) so select that.



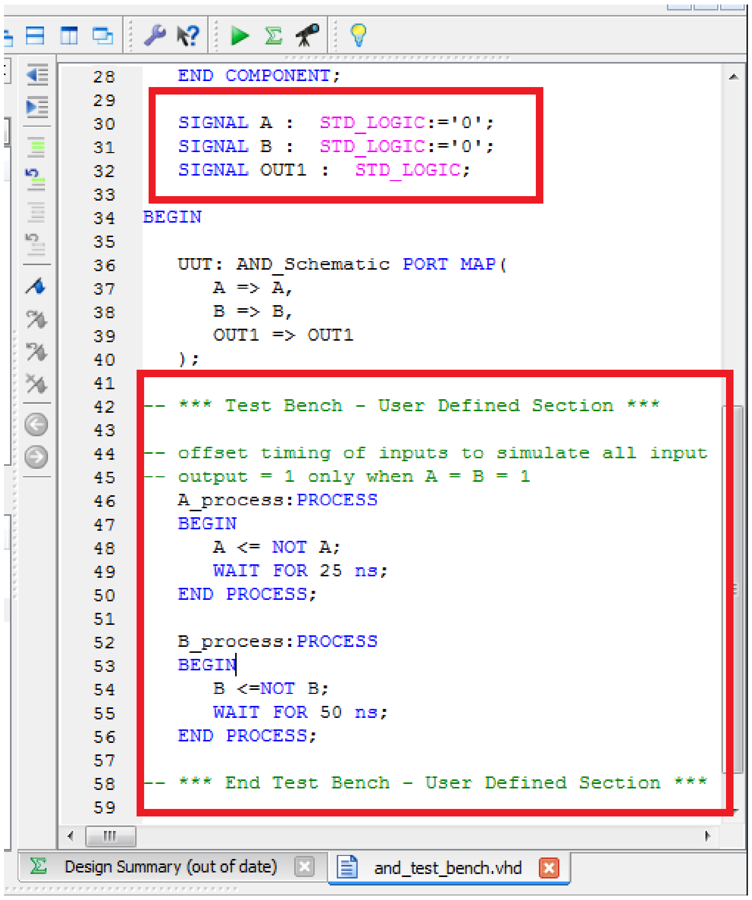
Select a source with which to associate the new source.

AND_Schematic

Click **next** to view a summary
The wizard will close and you will be presented with a file called [whatever you named the file].vhd which already has some VHDL code you will use to specify how to test your schematic you selected in the wizard.

Modifying the code

Even though there is a fair amount of VHDL code in the test bench file already, we need to make some changes. Make the following changes to the code. (See Section 5.1 for the full code). The changes highlighted in red are the only changes you need to make.

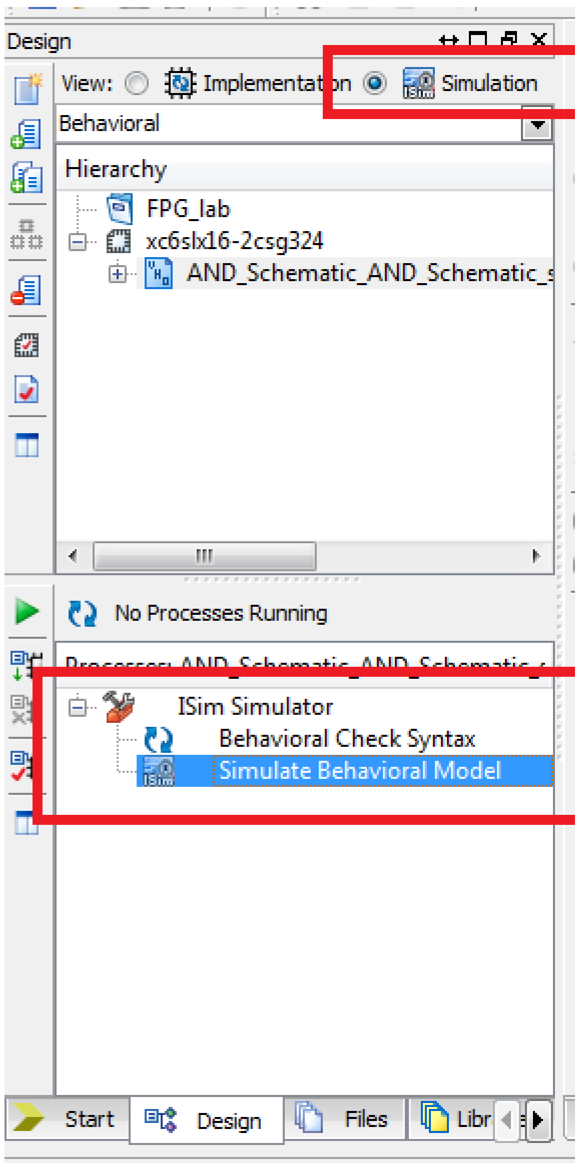


```
   END COMPONENT;

   SIGNAL A :  STD_LOGIC:='0';
   SIGNAL B :  STD_LOGIC:='0';
   SIGNAL OUT1 :  STD_LOGIC;

BEGIN

   UUT: AND_Schematic PORT MAP(
      A => A,
      B => B,
      OUT1 => OUT1
   );

-- *** Test Bench - User Defined Section ***

-- offset timing of inputs to simulate all input
-- output = 1 only when A = B = 1
   A_process:PROCESS
   BEGIN
      A <= NOT A;
      WAIT FOR 25 ns;
   END PROCESS;

   B_process:PROCESS
   BEGIN
      B <=NOT B;
      WAIT FOR 50 ns;
   END PROCESS;

-- *** End Test Bench - User Defined Section ***
```

Performing these sets give the signals an initial condition, and define the timing that that the inputs will be high and low at through the simulation. Again, this is only a simulation to test the design. The FPGA has not come into play yet.

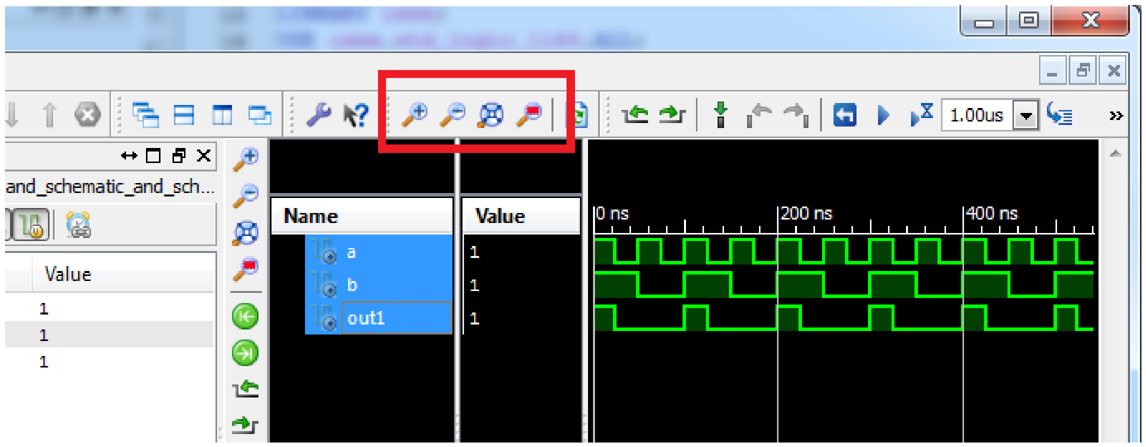
Save your work: **File-> Save all**

Run the simulation
Click design tab on left.
Click the **Simulation** radio button.
Select your test bench.
Expand ISIM simulator.
Double click simulate behavioral model.


Design
View: Implementation Simulation
Behavioral
Hierarchy
FPG_lab
xc6slx16-2csg324
AND_Schematic_AND_Schematic_s
No Processes Running
Processes: AND_Schematic_AND_Schematic_s
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model
Start
Design
Files
Libr

This compiles the schematic and test bench. Subsequently, it launches the ISim program and simulates the inputs to your circuit as we defined in the test bench file. Make sure to press the zoom out button to see the full signal. If you get errors, double check your test bench VHDL code. Errors will show up in your console.

Close ISIm if it works. If you have problems, close ISim and fix the problems and run the simulation again. Leaving ISim open after you change things might cause problems.



Name	Value
a	1
b	1
out1	1

Creating a constraint file

So far we have made a circuit design using a schematic, and then we tested it with a VHDL file used to define a process called a test bench for use with ISim. None of these things have had anything to do with our FPGA card yet.

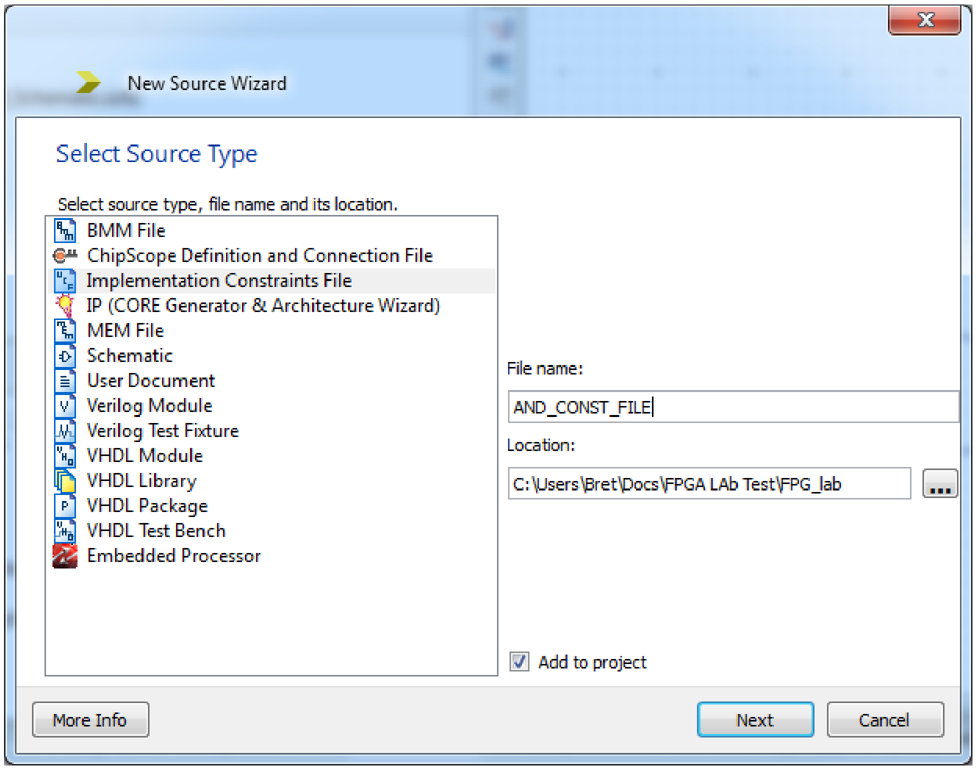
Constrain files take your designs, and associates the IO makers to actual pins on the FPGA board. It's the last step before uploading our design to the FPGA board, and it's the step that connects the design to the actual hardware of the FPGA. Constraint files have a **.ufc** file extension, and are associated with a particular schematic or VHDL design file.

The following process will walk you through the creation of a constrain file for our schematic.

Switch back to the implementation view.

Right click on our Schematic file in the hierarchy pane and click **new source.**

Select **implementation constraints file** from the list and give it a name.
When you see a constraint summary, click finish.



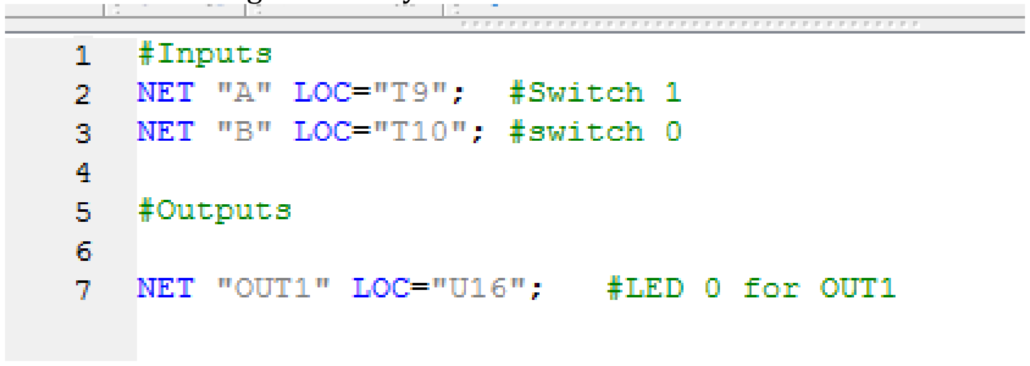
File name: AND_CONST_FILE

Location: C:\Users\Bret\Docs\FPGA LAb Test\FPG_lab

☑ Add to project

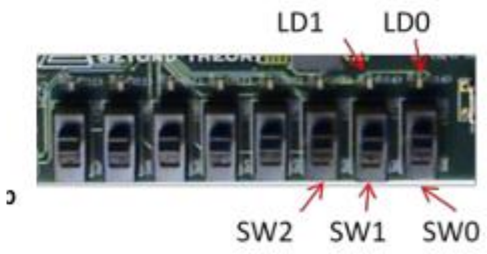
You should see now the .UFC file under your schematic in the hierarchy view.

Code your Constraint file:
Enter the following code into your schematics constraint file.

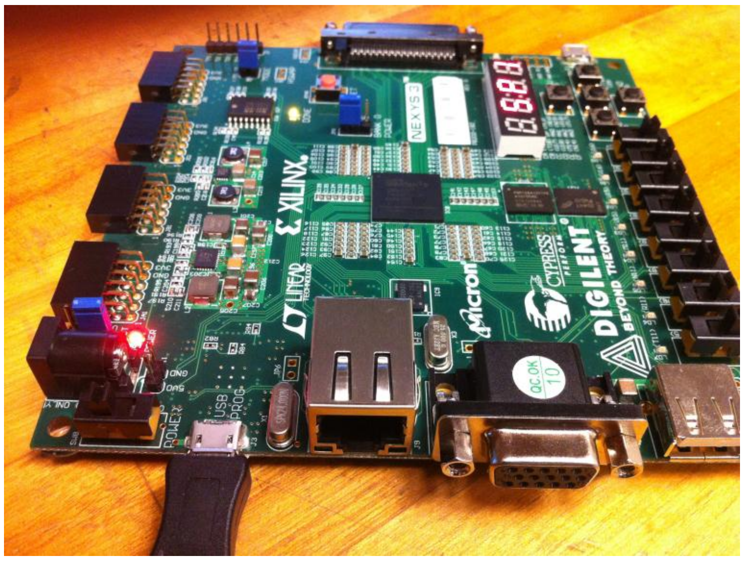


```
#Inputs
NET "A" LOC="T9";   #Switch 1
NET "B" LOC="T10"; #switch 0

#Outputs

NET "OUT1" LOC="U16";    #LED 0 for OUT1
```

Notice we reference the IO marker names we set in our schematic from before. But what about those LOC strings? Those are the actual pin names of switches and LEDs on the FPGA board. See for yourself! The names are silk screened right onto the board. We can also look at the following picture.

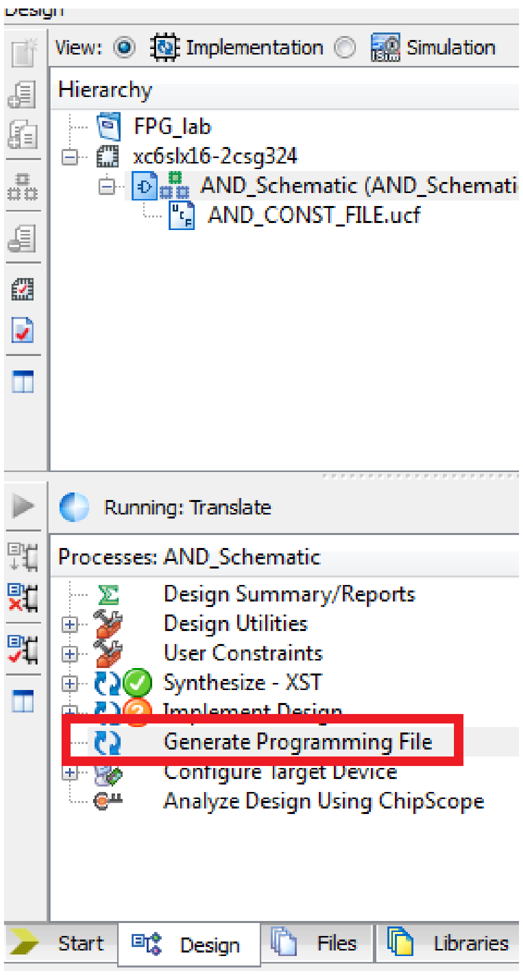


Plug in the FPGA board.
Lets get ready to create the worlds most overkill AND gate.
There are two USB ports. Use the one right next to the power switch.
Turn it on. Make sure it says "pass 128" on its small red LED screen.



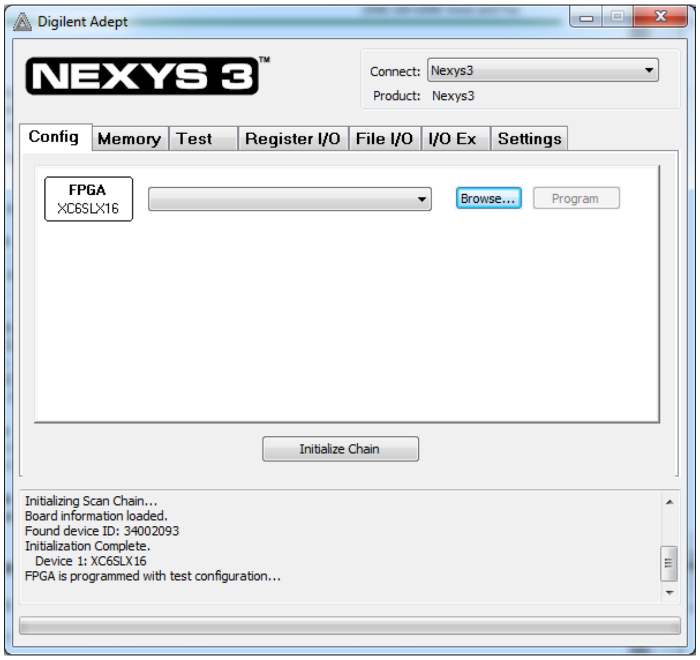
Compile your Design for Uploading

Highlight your **.sch** in your hierarchy view and double click ***Generate Programming File*** in your lower process menu**.**



This will take a few minutes. Once it stopped, you will have 3 green checks. If there were errors, try to fix them and try again.

Uploading the program to the board

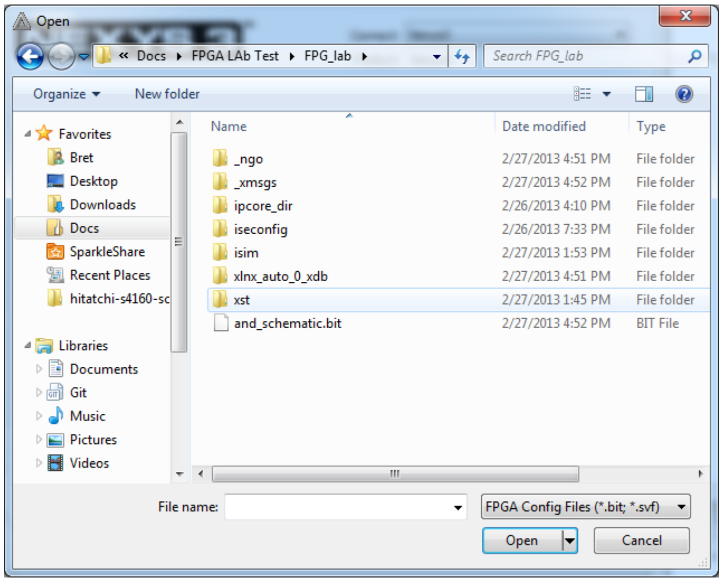


```
Initializing Scan Chain...
Board information loaded.
Found device ID: 34002093
Initialization Complete.
  Device 1: XC6SLX16
FPGA is programmed with test configuration...
```

Open **Digilent Adept** and switch to the **Test tab** and click **run RAM/Flash test.** Digilent Adept is the software used to upload .bit files to the Digilent FPGA cards. If it passes you are talking to your board. If not, something went wrong.

Go to the **Config tab** and click **browse.**

Navigate to your project directory folder and open the **.bit file** that has the same name as your schematic file.



Name	Date modified	Type
_ngo	2/27/2013 4:51 PM	File folder
_xmsgs	2/27/2013 4:52 PM	File folder
ipcore_dir	2/26/2013 4:10 PM	File folder
iseconfig	2/26/2013 7:33 PM	File folder
isim	2/27/2013 1:53 PM	File folder
xlnx_auto_0_xdb	2/27/2013 4:51 PM	File folder
xst	2/27/2013 1:45 PM	File folder
and_schematic.bit	2/27/2013 4:52 PM	BIT File

Click program and your compiled schematic is used to configure the FPGA.

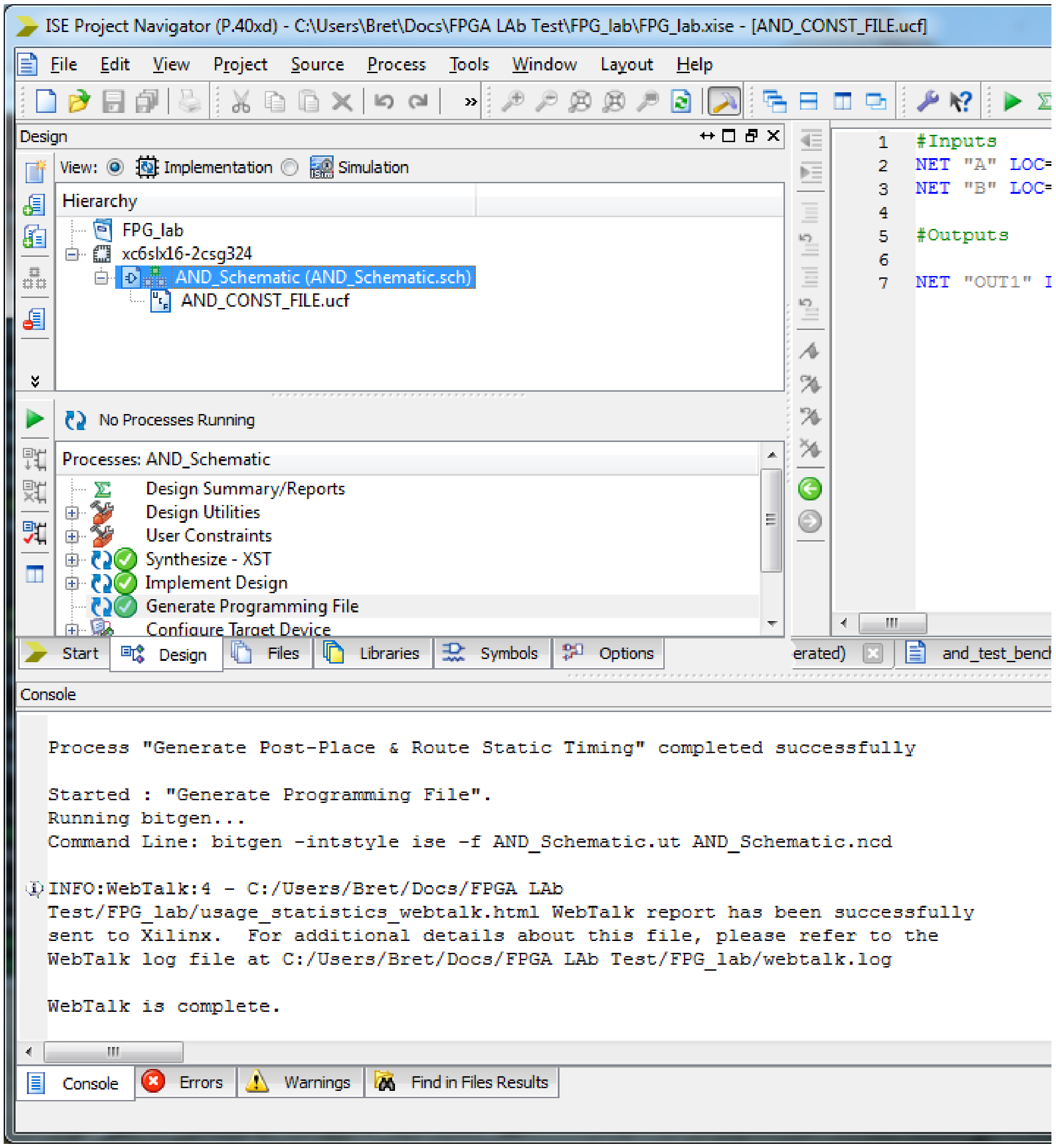
Test your AND gate, using the two switches on the board, confirming that the led comes on when both switches are on.

Notice that powering down your FPGA will reset it. FPGAs will lose their configuration when they lose power. Luckily, our board has nonvolatile memory which can store our configuration such that it is reconfigured every time it is powered on. There is an option in the Adept software to set up a program to load on power on.

4. Configuring the FPGA using VHDL

We will now do the *same exact thing* but using VHDL instead of a schematic. Instead of worrying about a schematic layout, all we need to do is write some VHDL code. In the long run, VHDL will give you far more millage than a schematic since managing code is far easier than managing schematics.

Right now, your workspace should look something like this:



```
#Inputs
NET "A" LOC=
NET "B" LOC=

#Outputs

NET "OUT1" I
```

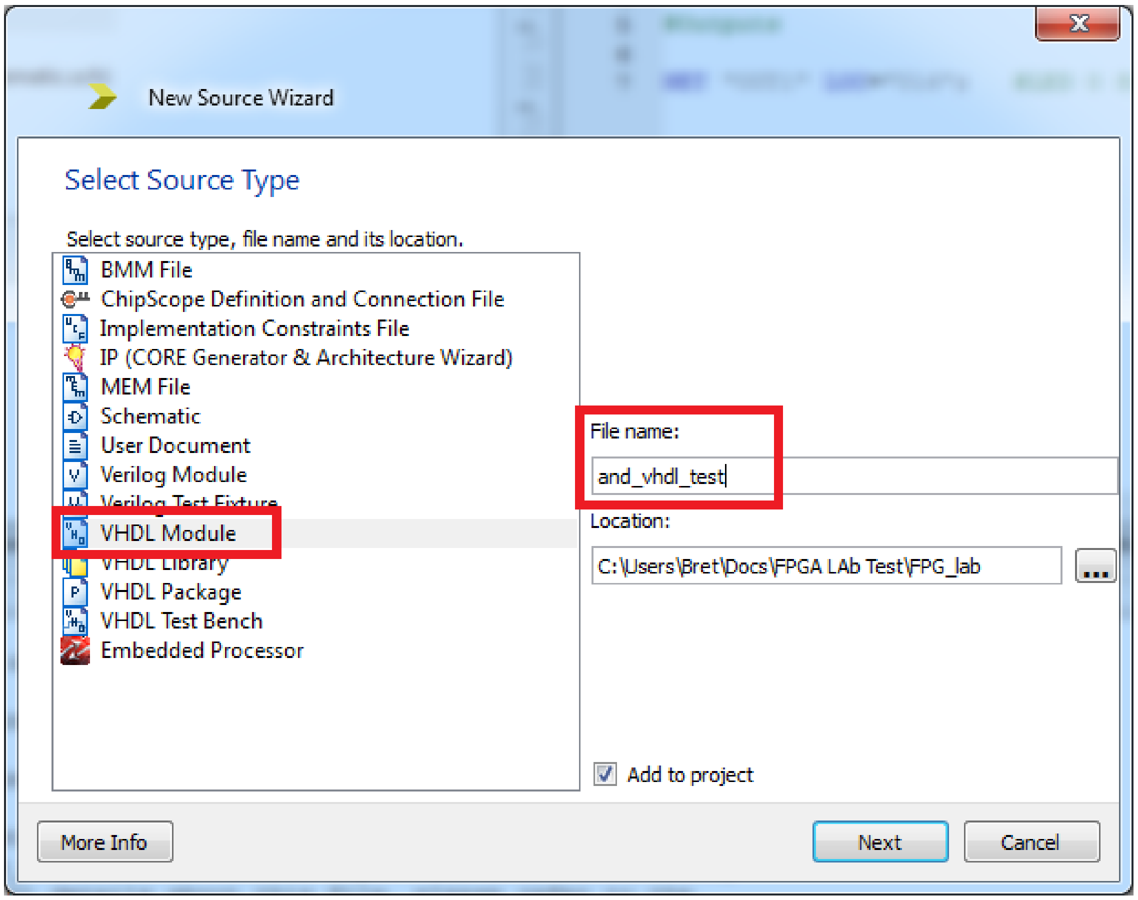
```
Process "Generate Post-Place & Route Static Timing" completed successfully

Started : "Generate Programming File".
Running bitgen...
Command Line: bitgen -intstyle ise -f AND_Schematic.ut AND_Schematic.ncd

INFO:WebTalk:4 - C:/Users/Bret/Docs/FPGA LAb
Test/FPG_lab/usage_statistics_webtalk.html WebTalk report has been successfully
sent to Xilinx.  For additional details about this file, please refer to the
WebTalk log file at C:/Users/Bret/Docs/FPGA LAb Test/FPG_lab/webtalk.log

WebTalk is complete.
```

In your hierarchy view, you have a schematic and a .ufc constraint file nested under it. If you were to switch to the simulation, you would find the schematic and the test bench simulation file nested under it.

Create a new VHDL Module

Right click the project folder in the hierarchy view and click **New Source.**

Select "VHDL Module" and give it a name.



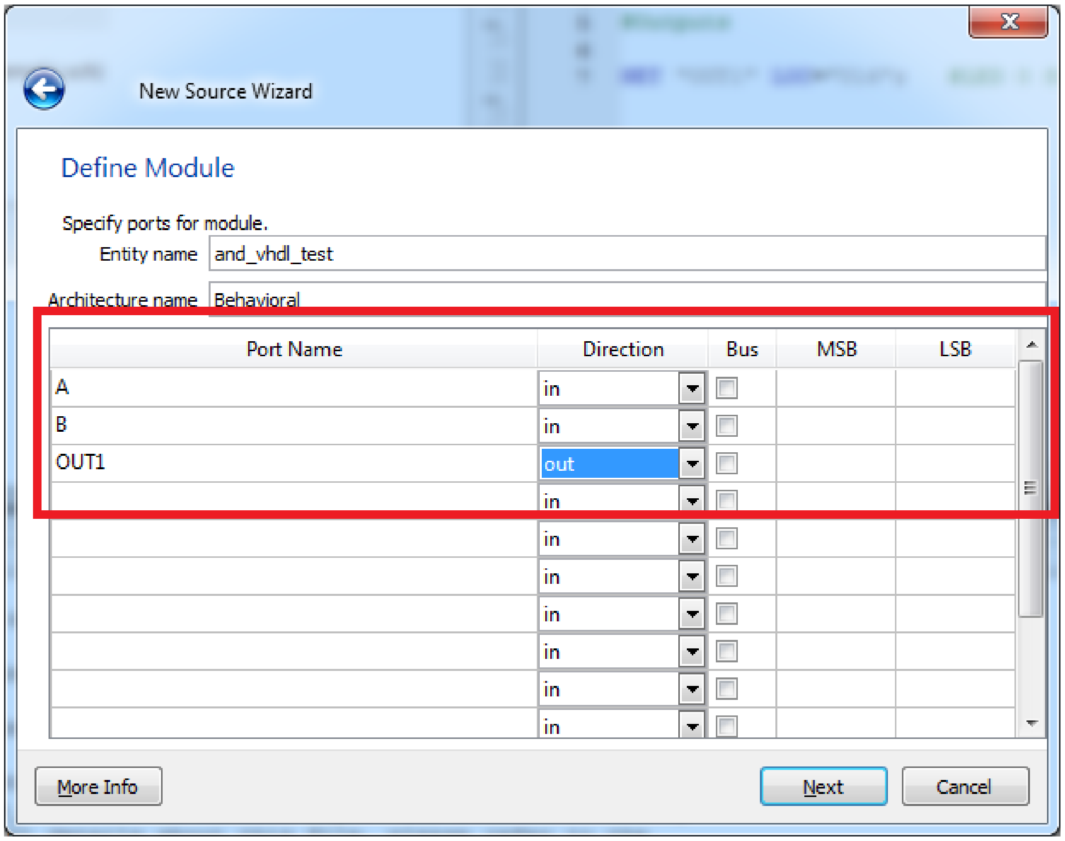
Select source type, file name and its location.

File name: and_vhdl_test

Location: C:\Users\Bret\Docs\FPGA LAb Test\FPG_lab

☑ Add to project

After clicking next, add two input ports and one output port in the port mapping window.
Like the IO markers, we need to define inputs and outputs for our VHDL design. These ports will be referenced in our VHDL code.



Entity name: and_vhdl_test
Architecture name: Behavioral

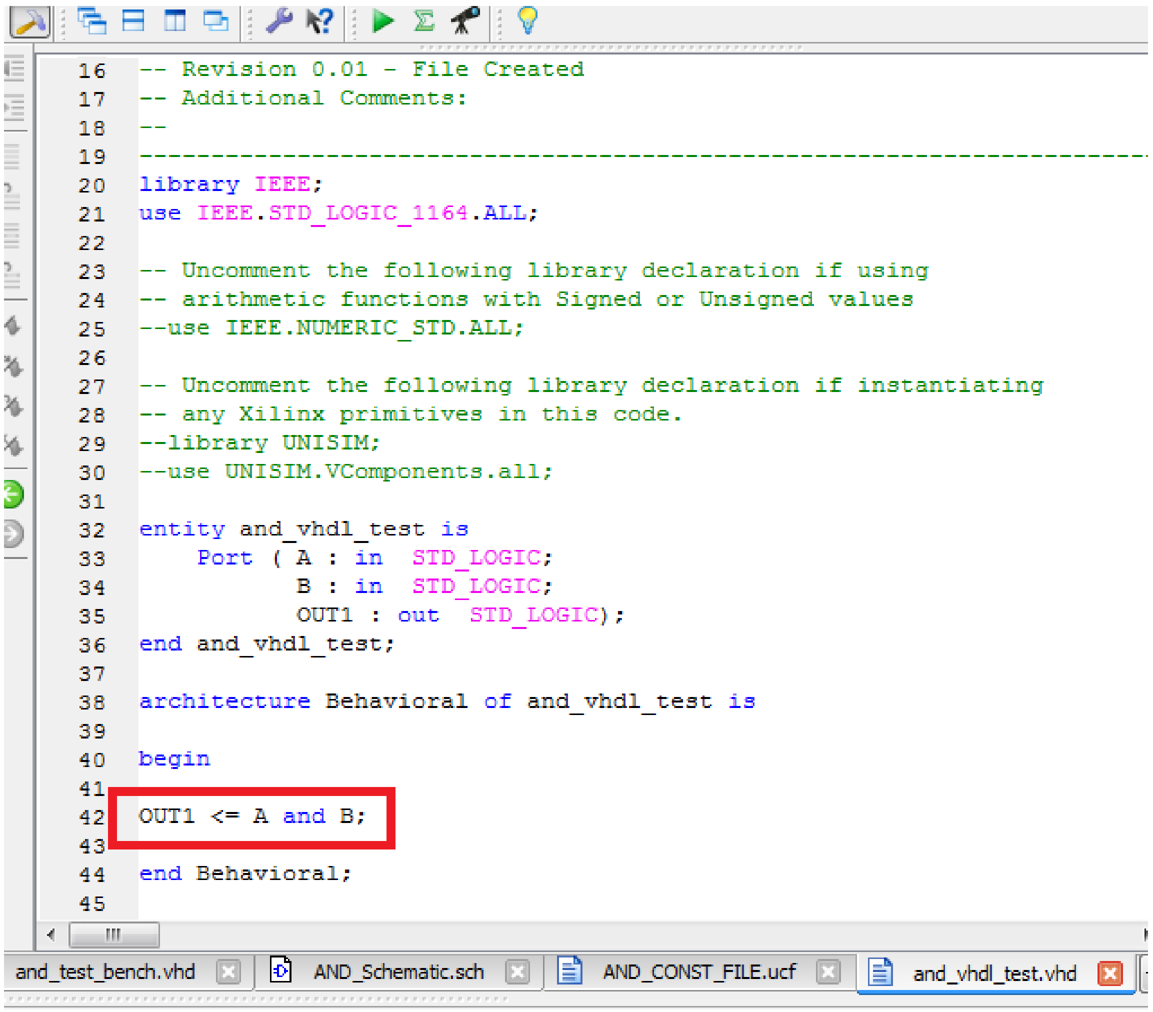
Port Name	Direction	Bus	MSB	LSB
A	in	☐		
B	in	☐		
OUT1	out	☐		
	in	☐		
	in	☐		
	in	☐		
	in	☐		
	in	☐		
	in	☐		
	in	☐		

Click next and the finish.

Once the new source wizard is closed, it should automatically open your new VHDL program. If not, you will find the VHDL file you just created in the hierarchy menu.

Just to clarify, we are creating a brand new VHDL file. It will be pre-populate the file with some skeleton VHDL code that is based off of the settings we set when when created the project.

Add the following lines of code to your VHDL file and Save it:
The full code is included in section 5.2



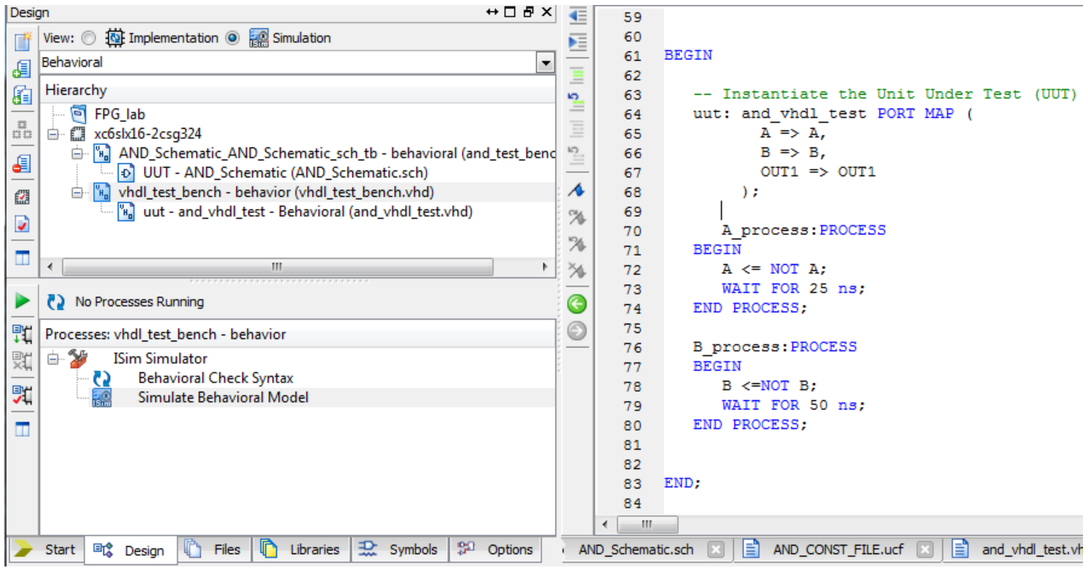
```vhdl
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- 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 primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity and_vhdl_test is
    Port ( A : in  STD_LOGIC;
           B : in  STD_LOGIC;
           OUT1 : out  STD_LOGIC);
end and_vhdl_test;

architecture Behavioral of and_vhdl_test is

begin

OUT1 <= A and B;

end Behavioral;

```

That's it! You just made an AND gate using VHDL. Now let's test and constrain our VHDL file, like we did for the schematic before.

Switch to the Simulate Mode, Right click your VHDL file, and create a new test bench file.
It will ask you which file you want to create the test bench for during the wizard.

Give your Test bench file a name and finish the wizard.

Modify your new Test bench file like below:


Design
View: Implementation Simulation
Behavioral
Hierarchy
FPG_lab
xc6slx16-2csg324
AND_Schematic_AND_Schematic_sch_tb - behavioral (and_test_benc
UUT - AND_Schematic (AND_Schematic.sch)
vhdl_test_bench - behavior (vhdl_test_bench.vhd)
uut - and_vhdl_test - Behavioral (and_vhdl_test.vhd)
No Processes Running
Processes: vhdl_test_bench - behavior
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model
Start
Design
Files
Libraries
Symbols
Options
AND_Schematic.sch
AND_CONST_FILE.ucf
and_vhdl_test.vh

```


BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: and_vhdl_test PORT MAP (
          A => A,
          B => B,
          OUT1 => OUT1
        );

      A_process:PROCESS
   BEGIN
      A <= NOT A;
      WAIT FOR 25 ns;
   END PROCESS;

   B_process:PROCESS
   BEGIN
      B <=NOT B;
      WAIT FOR 50 ns;
   END PROCESS;


END;

```

Run the simulation for the new test bench and confirm that it is working.

Constraint Files on your own

See if you can create a constraint file to compile and upload your VHDL to your FPGA card on your own. This should be strikingly similar to the process before.

Make sure when you go back to the implementation view to add a source, right click on the VHDL file to add a new source, and not the schematic file.

Additional Projects

- Add an additional OR gate to your VHDL file, or schematic.
- Create an adder using the schematic view, and tie the inputs to the switches, and the outputs to the LEDs.
- (Optional) Use a function generator to drive an input of the FPGA to a digital circuit with a square wave, and watch the output on the oscilloscope. How fast can you drive the FPGA before your output begins to degrade?

5 Example Code

5.1 Schematic Test Bench:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
LIBRARY UNISIM;
USE UNISIM.Vcomponents.ALL;
ENTITY AND_Schematic_AND_Schematic_sch_tb IS
END AND_Schematic_AND_Schematic_sch_tb;
ARCHITECTURE behavioral OF AND_Schematic_AND_Schematic_sch_tb IS

   COMPONENT AND_Schematic
   PORT( A   :   IN  STD_LOGIC;
          B :   IN  STD_LOGIC;
          OUT1  :   OUT STD_LOGIC);
   END COMPONENT;

   SIGNAL A :   STD_LOGIC:='0';
   SIGNAL B :   STD_LOGIC:='0';
   SIGNAL OUT1  :   STD_LOGIC;

BEGIN

   UUT: AND_Schematic PORT MAP(
        A => A,
        B => B,
        OUT1 => OUT1
   );

-- *** Test Bench - User Defined Section ***

-- offset timing of inputs to simulate all input options
-- output = 1 only when A = B = 1
    A_process:PROCESS
    BEGIN
        A <= NOT A;
        WAIT FOR 25 ns;
    END PROCESS;

    B_process:PROCESS
    BEGIN
        B <=NOT B;
        WAIT FOR 50 ns;
    END PROCESS;

-- *** End Test Bench - User Defined Section ***

END;
```

5.2 VHDL AND Gate

```
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 primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity and_vhdl_test is
    Port ( A : in  STD_LOGIC;
           B : in  STD_LOGIC;
           OUT1 : out  STD_LOGIC);
end and_vhdl_test;

architecture Behavioral of and_vhdl_test is

begin

OUT1 <= A and B;


end Behavioral;
```

5.3 VHDL Test Bench

```
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;

ENTITY vhdl_test_bench IS
END vhdl_test_bench;

ARCHITECTURE behavior OF vhdl_test_bench IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT and_vhdl_test
    PORT(
         A : IN  std_logic;
         B : IN  std_logic;
         OUT1 : OUT  std_logic
        );
    END COMPONENT;


   --Inputs
   signal A : std_logic := '0';
   signal B : std_logic := '0';
```

```
    --Outputs
   signal OUT1 : std_logic;
   -- No clocks detected in port list. Replace <clock> below with
   -- appropriate port name


BEGIN

    -- Instantiate the Unit Under Test (UUT)
   uut: and_vhdl_test PORT MAP (
          A => A,
          B => B,
          OUT1 => OUT1
        );

        A_process:PROCESS
    BEGIN
        A <= NOT A;
        WAIT FOR 25 ns;
    END PROCESS;

    B_process:PROCESS
    BEGIN
        B <=NOT B;
        WAIT FOR 50 ns;
    END PROCESS;


END;
```

6 References

- Labs 1 and 2 developed by the Configurable Space Microsystems Innovations and Applications Center.
  http://www.cosmiac.org/Projects_FPGA.html#Lab1

- How to Program in VHDL
  Making a simple counter. FPGA Xilinx
  Video tutorial on how to make a simple counter in VHDL for the Basys2 board, which contains a Xilinx Spartan 3E FPGA
  http://www.youtube.com/watch?v=0b7B6x5g6tw