FPGA Introduction Lab

A. La Rosa, B. Comnes
Physics 315
Portland State University

Abstract

In this lab we will be briefly exploring Field Programmable Gate Arrays (FPGA). This will be achieved by a brief background discussion of FPGAs, followed up by a tutorial illustrating a very basic workflow, including FPGA design using the schematic layout as well as an example of using VHDL to implement very basic digital circuits. The student is run through how to set up the design, how to write a test, and how to compile and upload the final design to the FPGA. The lab was designed around the Digilent Nexys 3 prototyping board, which provides relatively access to a Xilinx Spartan6 XC6LX16-CS324 FPGA chip along with a programmer, program memory, and array of switches, buttons, LEDs and I/O ports.

1 FPGA Introduction

1.1 What is an FPGA?

FPGAs are used as tools in modern digital circuit design. FPGAs provide a large array of generic logic gates inside of a single integrated circuit (IC) that can be configured arbitrarily, so long as it does not exceed the capabilities of your FPGA chip.

The FPGA is 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.

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.

1.2 How is it different than 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.

Also, unlike other 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 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.

1.4 Drawbacks

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.

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**. This integrated development environment (IDE) will allow us to set up a project and create different design, test and constraint files, as well as act as a debugger and handle the compiling of our designs.

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

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**.

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.

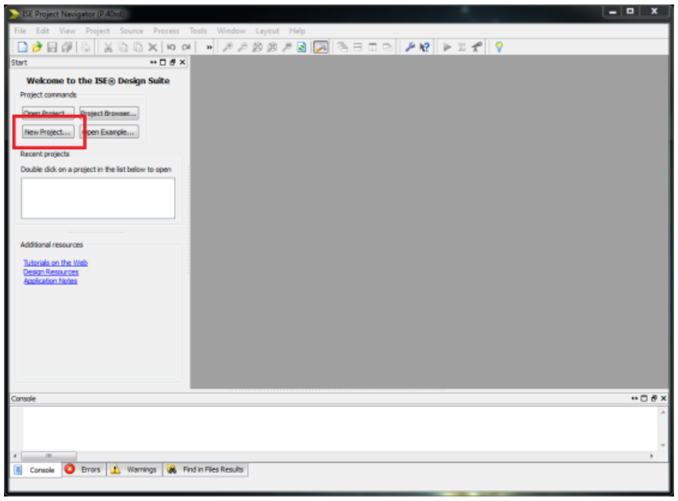
These languages are similar to a programming language, but are rather geared towards digital circuit design, so they have the same limitations has digital circuits. For this lab, these differences will not be of focus. Understanding these differences would be important if you were actually to use an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

3 Configuring the FPGA using schematics

In this section we will implement a simple AND gate, add constraints to hook op two onboard switches and an LED to the AND gate, test and deploy our design to the actual FPGA.

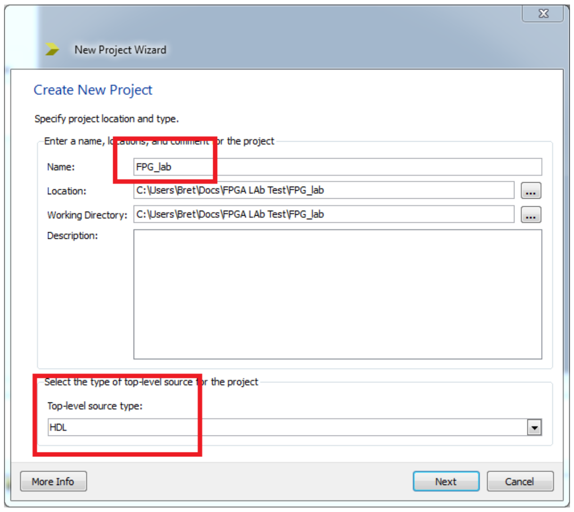
Find a computer.
You must use one of the 201 lab computers for this lab, as they are the only ones with the licensed software already installed.

Open ISE Project Explorer
There should be a desktop icon, otherwise search through the Start menu to find it, or ask for help if you cant find it.



Go to File-> New Project

Give your project 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.



Create New Project

Specify project location and type.

Enter a name, locations, and comment for the project

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:

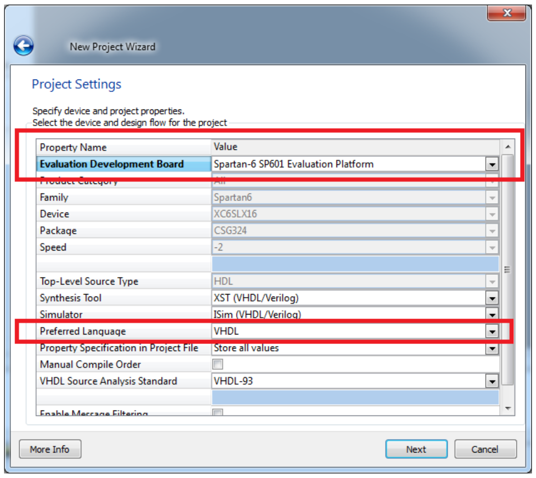
Select the type of top-level source for the project

Top-level source type: HDL

Also, Under Top-Level Source, Choose HDL

Click Next.

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.



Project Settings

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

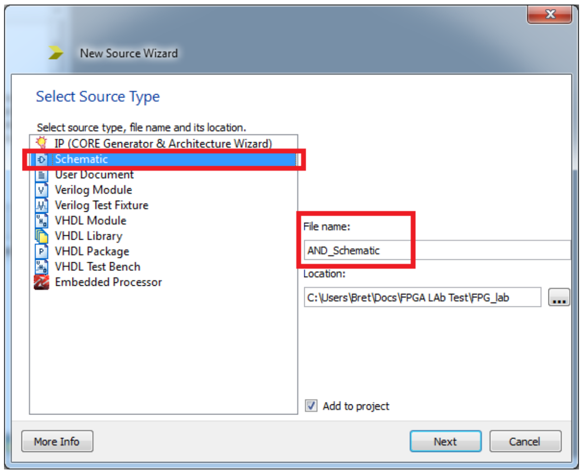
Property Name	Value
Evaluation Development Board	Spartan-6 SP601 Evaluation Platform
Product Category	All
Family	Spartan6
Device	XC6SLX16
Package	CSG324
Speed	-2
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	VHDL
Property Specification in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

Choose VHDL under the Preferred Language setting

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 experience.

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 name bugs if you go to far out of the naming scope.



Select Source Type

Select source type, file name and its location.

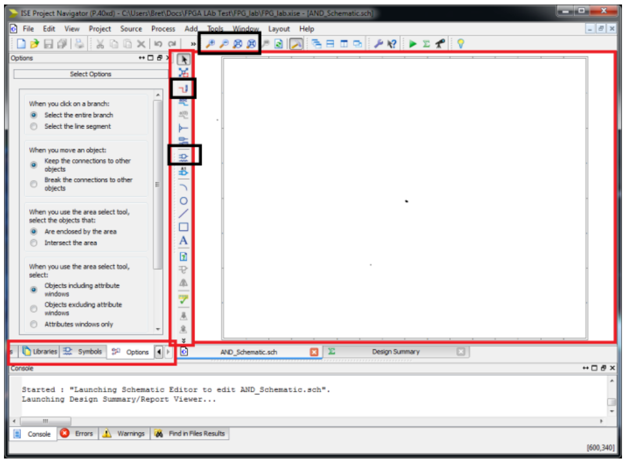
- IP (CORE Generator & Architecture Wizard)
- Schematic
- User Document
- Verilog Module
- Verilog Test Fixture
- VHDL Module
- VHDL Library
- VHDL Package
- VHDL Test Bench
- Embedded Processor

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, you your schematic will automatically open.


ISE Project Navigator (P.40xd) - C:\Users\Bret\Docs\FPGA LAb Test\FPG_lab\FPG_lab.xise - [AND_Schematic.sch]

Options — Select Options

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

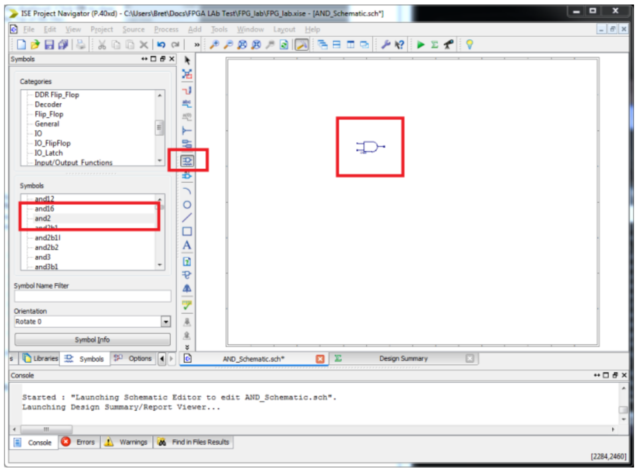
When you move an object: [✓ Keep the connections to other objects] Break the connections to other objects

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...
```

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 so get used to it.


ISE Project Navigator (P.40xd) - C:\Users\Bret\Docs\FPGA LAb Test\FPG_lab\FPG_lab.xise - [AND_Schematic.sch*]

Symbols

Categories: DDR_Flip_Flop, Decoder, Flip_Flop, General, IO, IO_FlipFlop, IO_Latch, Input/Output Functions

Symbols: and12, and16, and2, and2b1, and2b1l, and2b2, and3, and3b1

Symbol Name Filter

Orientation: Rotate 0

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

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 name filter is the quickest way to find what you are looking for if you know the name.

Click the logic category

Notice the symbols list shortened to only show the logic components in alphabetical order.

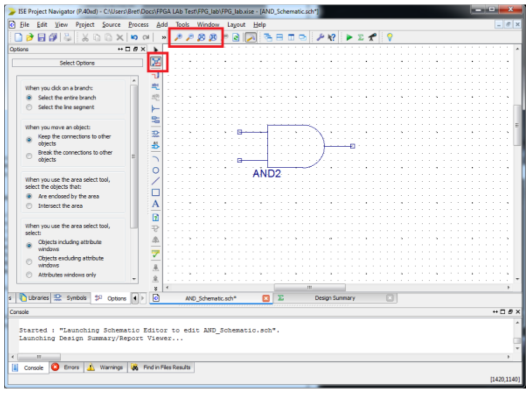
Choose the *AND2* symbol from the symbol menu.

Now when you roll over the schematic view, you get something that looks like that component following around a cross. 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.

Use the zoom tools to zoom in on your parts.


ISE Project Navigator (P.40xd) - C:\Users\Bret\Docs\FPGA LAb Test\FPG_lab\FPG_lab.xise - [AND_Schematic.sch*]

Options — Select Options

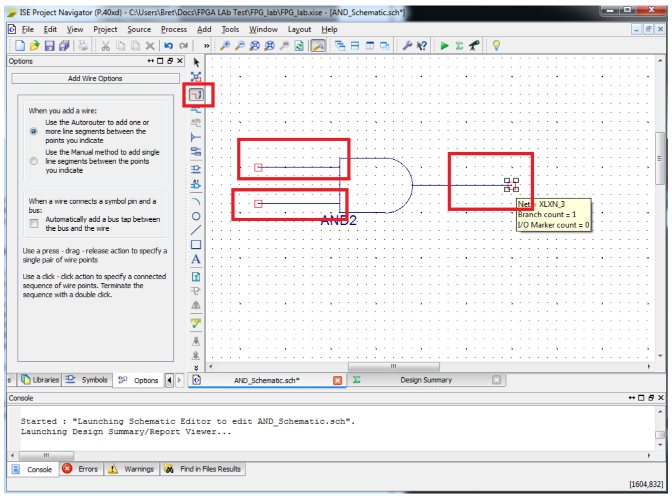
When you click on a branch: [✓ Select the entire branch] Select the line segment

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...
```


ISE Project Navigator (P.40xd) - C:\Users\Bret\Docs\FPGA LAb Test\FPG_lab\FPG_lab.xise - [AND_Schematic.sch*]

Add Wire Options

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

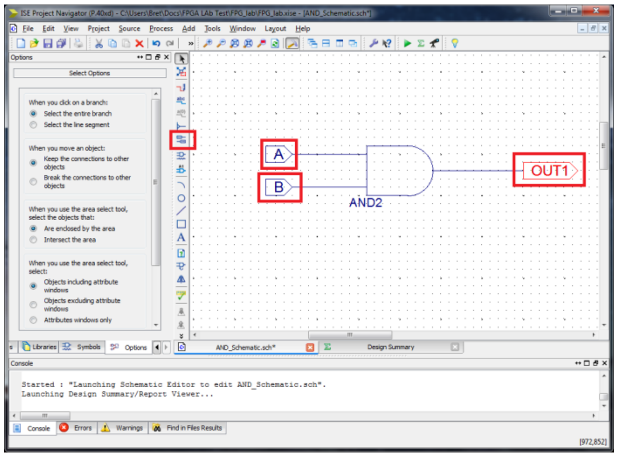
Use a click - click action to specify a connected sequence of wire points. Terminate the sequence with a double click.

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...
```

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.

Add IO markers.
These markers give names to the inputs and outputs of your design. You refer to these when you constrain your design to the FPGA.


ISE Project Navigator (P.40xd) - C:\Users\Bret\Docs\FPGA LAb Test\FPG_lab\FPG_lab.xise - [AND_Schematic.sch*]

Select Options

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

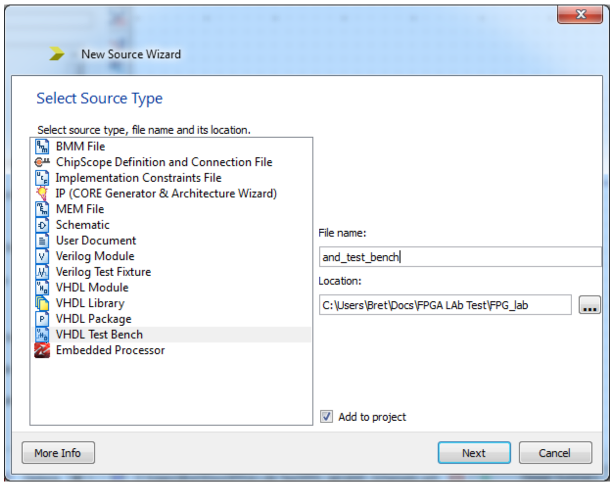
When you use the area select tool, select:
[✓ Objects including attribute windows]
Objects excluding attribute windows
Attributes windows only

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

Click the add IO button in 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.

Save your work. File -> Save all
Now we test our schematic to see if it works.


New Source Wizard

Select Source Type

Select source type, file name and its location.

- BMM File
- ChipScope Definition and Connection File
- Implementation Constraints File
- IP (CORE Generator & Architecture Wizard)
- MEM File
- Schematic
- User Document
- Verilog Module
- Verilog Test Fixture
- VHDL Module
- VHDL Library
- VHDL Package
- VHDL Test Bench
- Embedded Processor

File name: and_test_bench

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

[✓] Add to project

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.

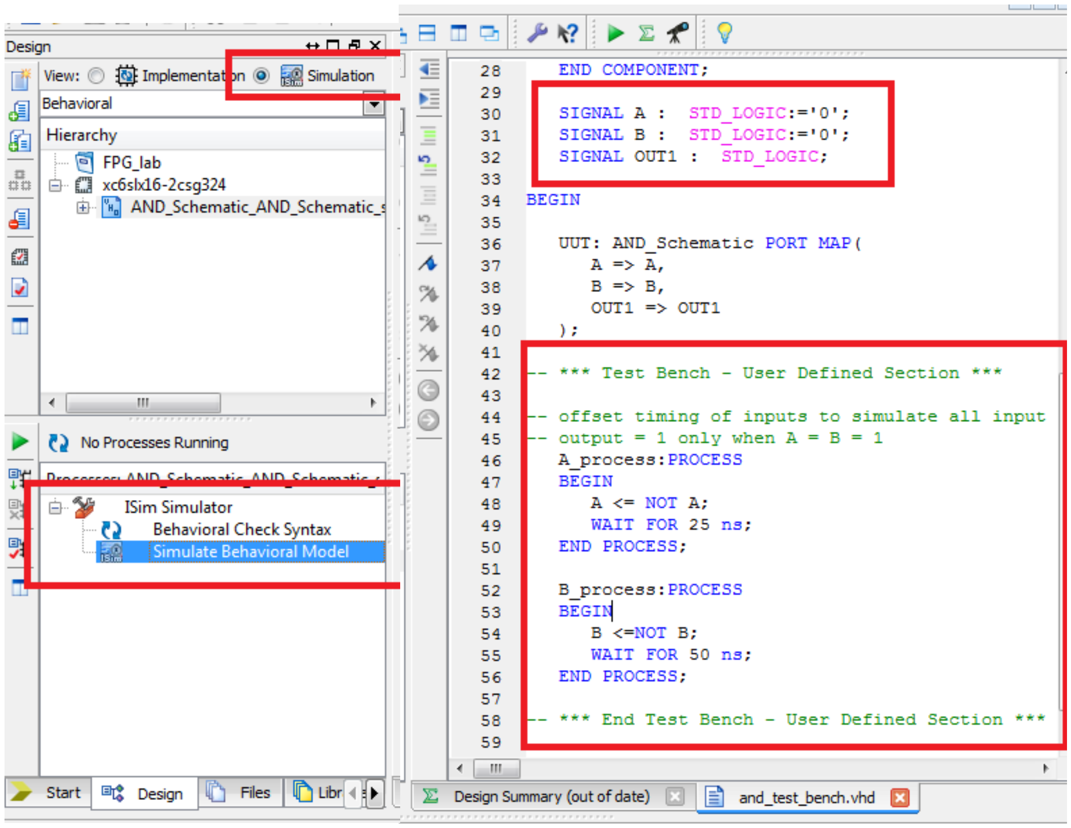
Click next to view a summary.
The wizard will close and you will be presented with a file called [name of file].vhd which is your vhdl code you will use to specify how to test your schematic you selected in the wizard.
This VHDL file will be where we describe how we want to test our circuit during simulation.

Make the following changes to the code.
Doing this sets give the signals an initial condition, and defines what they value they hold throughout the simulation.

Save your work. File-> Save all

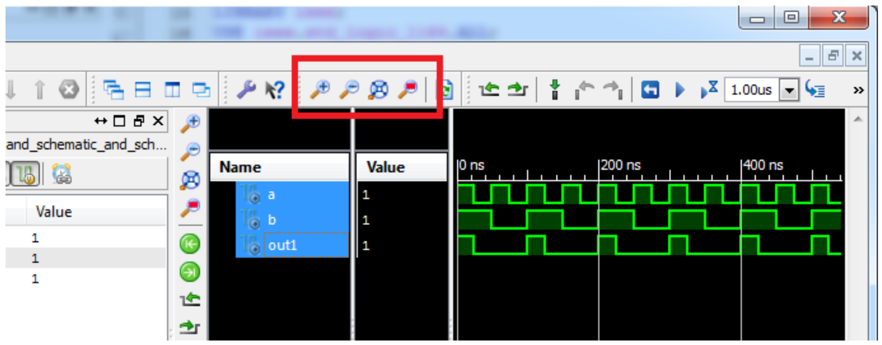
Click design tab on left

Click simulation.


Design
View: Implementation Simulation
Behavioral
Hierarchy
FPG_lab
xc6slx16-2csg324
AND_Schematic_AND_Schematic_s
No Processes Running
Processes: AND_Schematic_AND_Schematic
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model
Start
Design
Files
Libr
Design Summary (out of date)
and_test_bench.vhd

```
    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 ***
```

Select your test bench. Expand ISIM simulator. Double click simulate behavioral model.
This compiles the schematic, and test bench. After, it a 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.



Close ISIm if it works.
Now we need to create a constraints file (A .UFC file). This file assigns IO pins of the FPGA to the IO markers of our code.

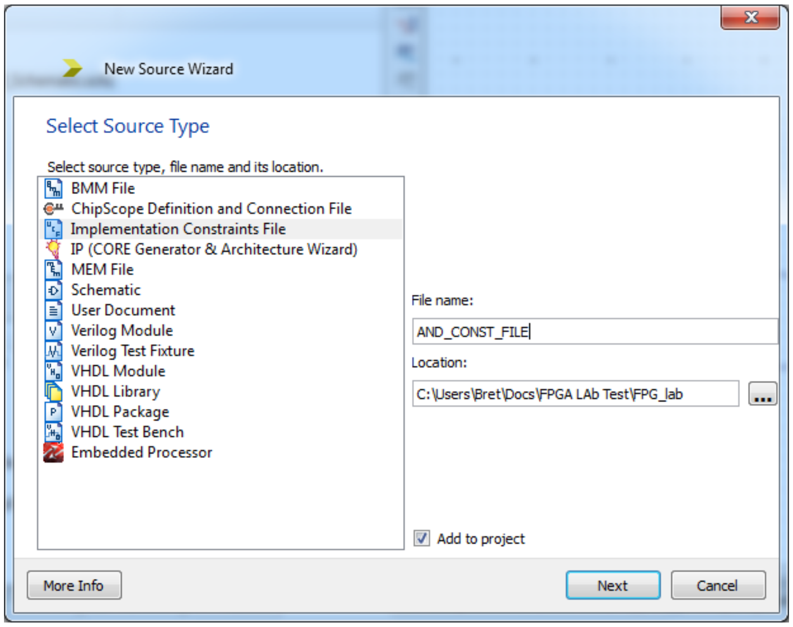
Switch back to the implementation view.
Now we add a constraint file. This file constrains our IO makers to the physical pins we want on our FPGA.

Right click the hierarchy pane and click new source.

Select implementation constraints file and give it a name.

When you see a constraint summary, click finish.

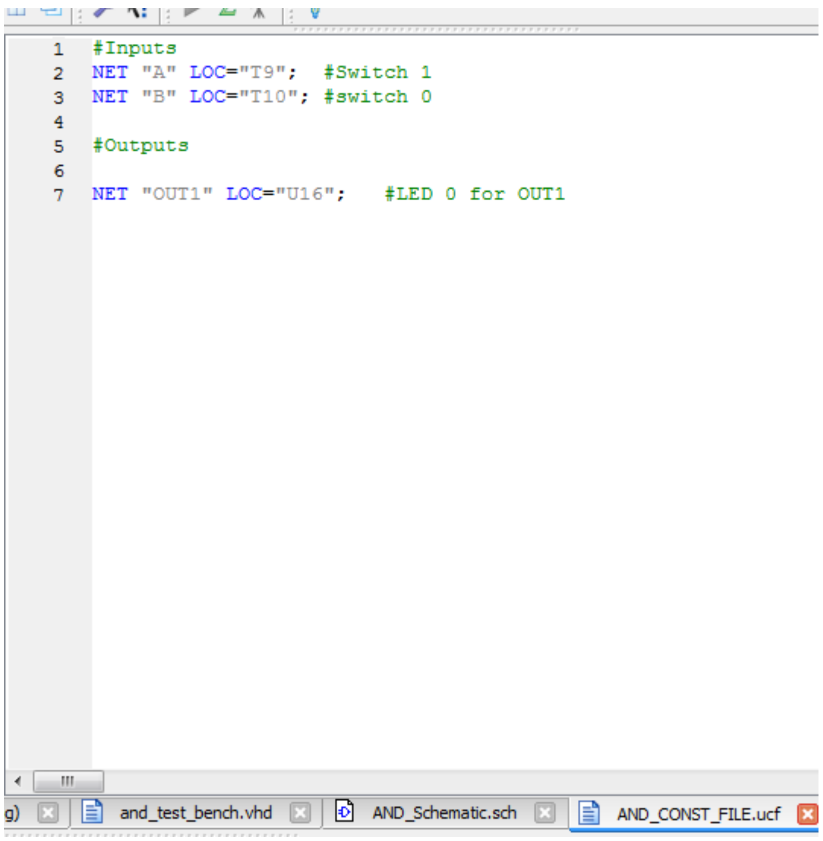
You should now the .UFC file under your schematic in the hierarchy view.



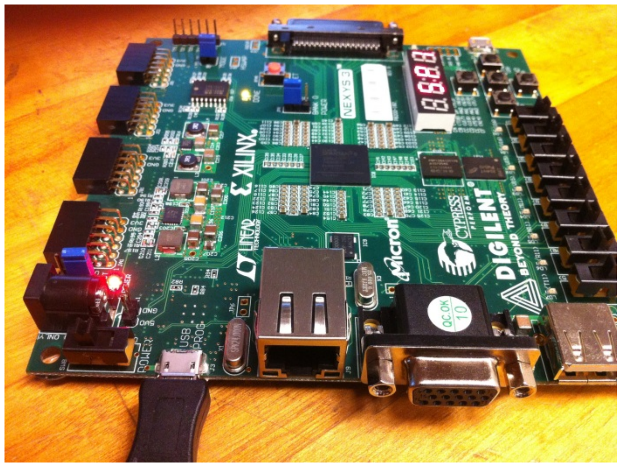
File name: AND_CONST_FILE

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

Enter the following code.
Notice we reference the IO labels we used 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 here



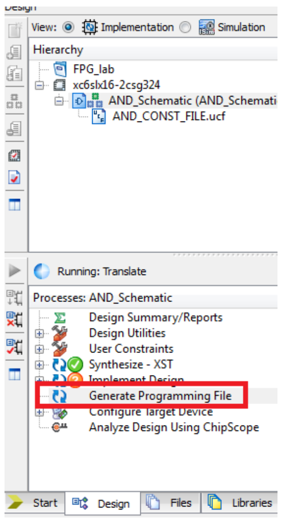
```
#Inputs
NET "A" LOC="T9";  #Switch 1
NET "B" LOC="T10"; #switch 0

#Outputs

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



Plug in the FPGA board.
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 screen.

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.



View: Implementation / Simulation

Hierarchy
- FPG_lab
- xc6slx16-2csg324
  - AND_Schematic (AND_Schemati
    - AND_CONST_FILE.ucf

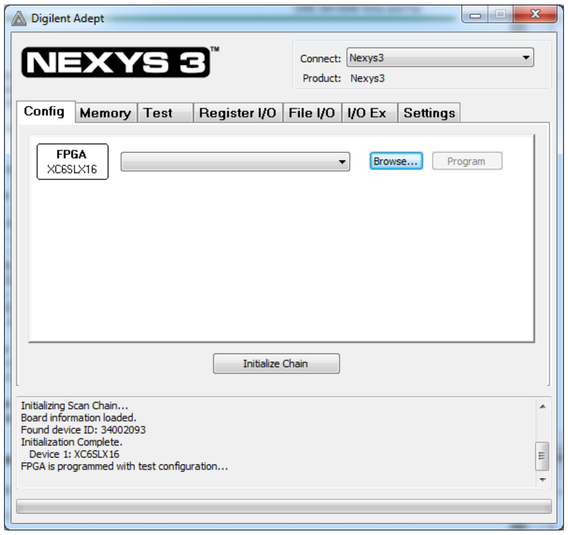
Running: Translate

Processes: AND_Schematic
- Design Summary/Reports
- Design Utilities
- User Constraints
- Synthesize - XST
- Implement Design
- Generate Programming File
- Configure Target Device
- Analyze Design Using ChipScope

Now upload the program to your board.
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.



Digilent Adept

NEXYS 3™

Connect: Nexys3
Product: Nexys3

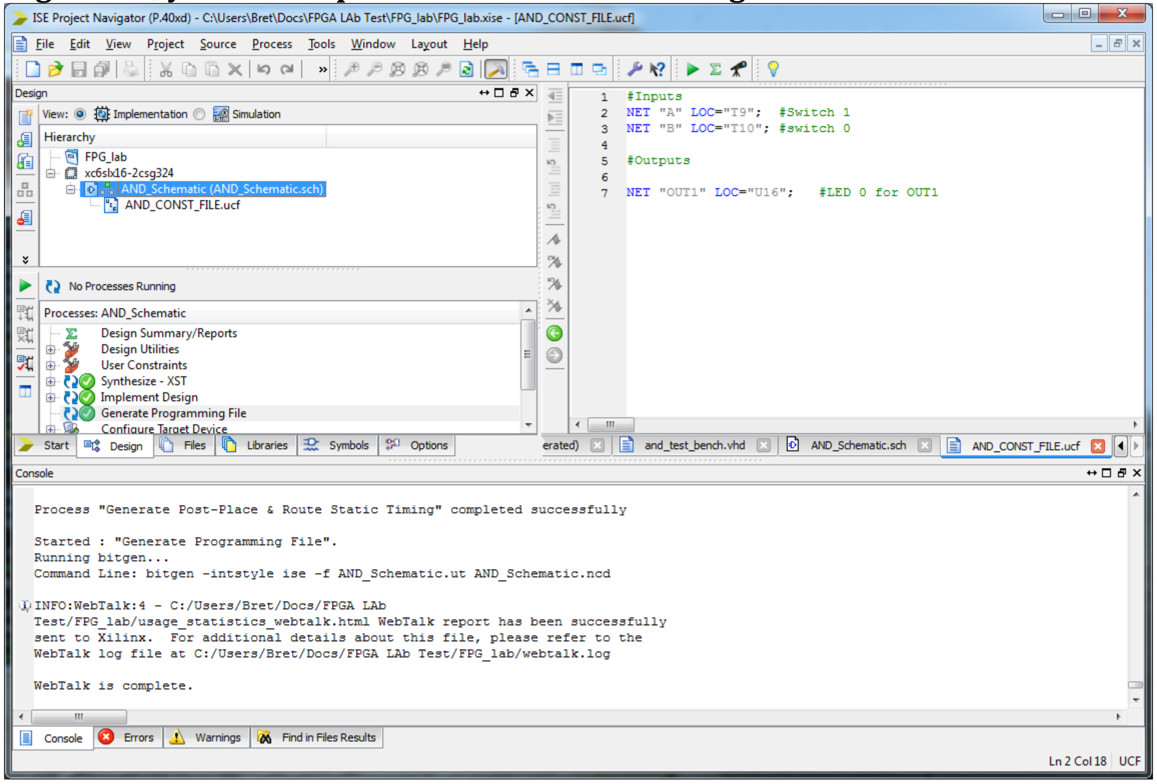
```
Initializing Scan Chain...
Board information loaded.
Found device ID: 34002093
Initialization Complete.
  Device 1: XC6SLX16
FPGA is programmed with test configuration...
```

Click program and your schematic is now being configured on the FPGA.
Test your and gate, confirming that the led comes on when both switches are on.
Note, 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.

• 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:



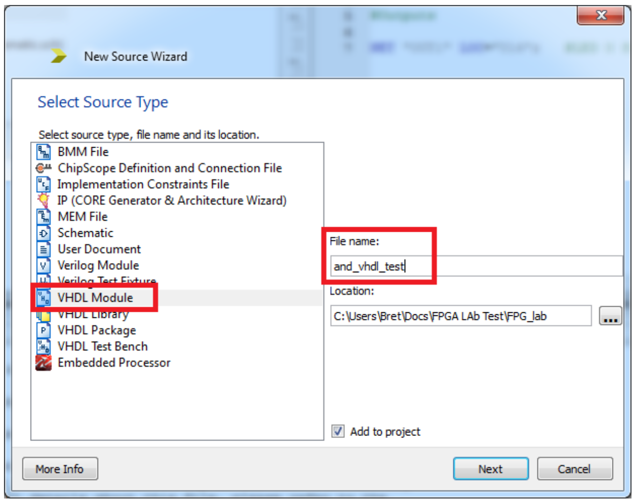
```
#Inputs
NET "A" LOC="T9";  #Switch 1
NET "B" LOC="T10"; #switch 0

#Outputs

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

```
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 as well.

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

Select "VHDL Module" and give it a name.

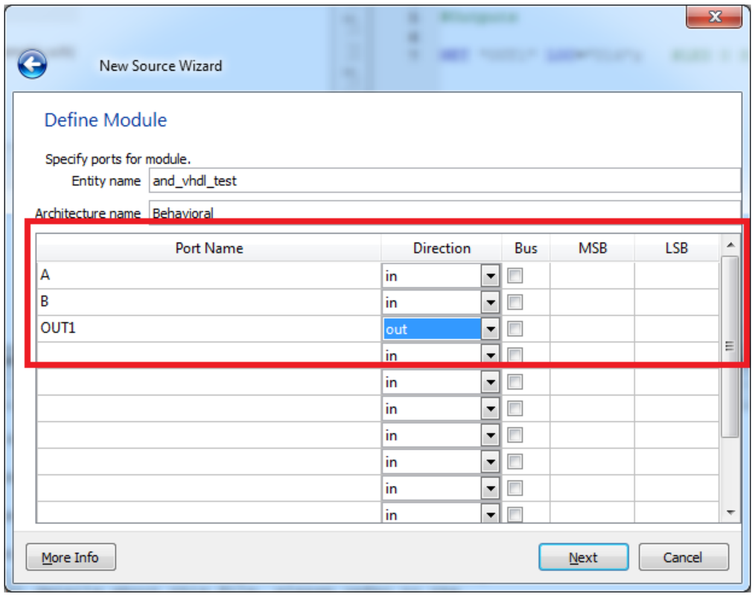


File name: and_vhdl_test

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.



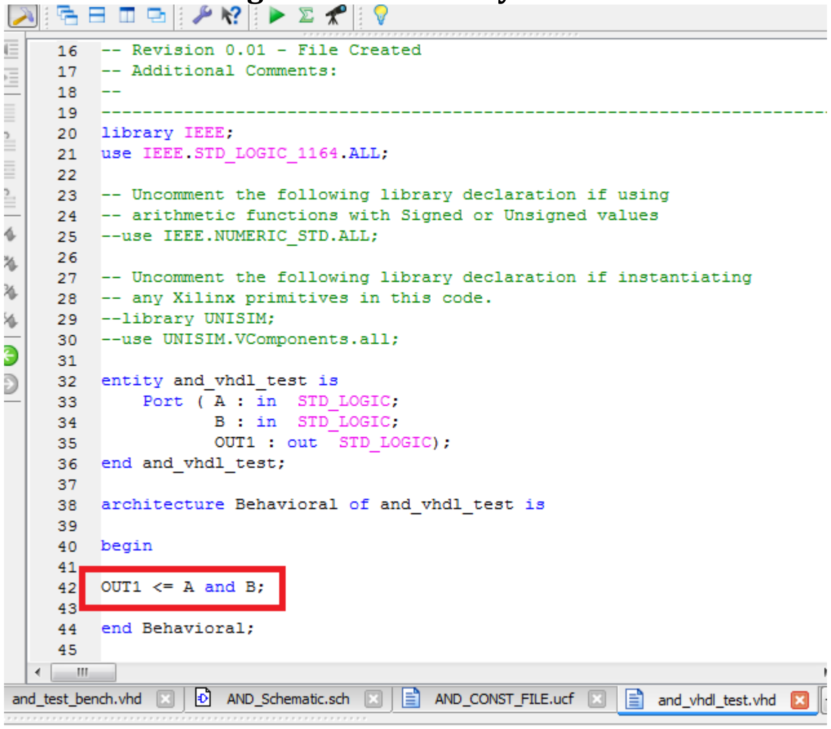
Entity name: and_vhdl_test
Architecture name: Behavioral

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.

Add the following lines of code to your VHDL file and Save it:



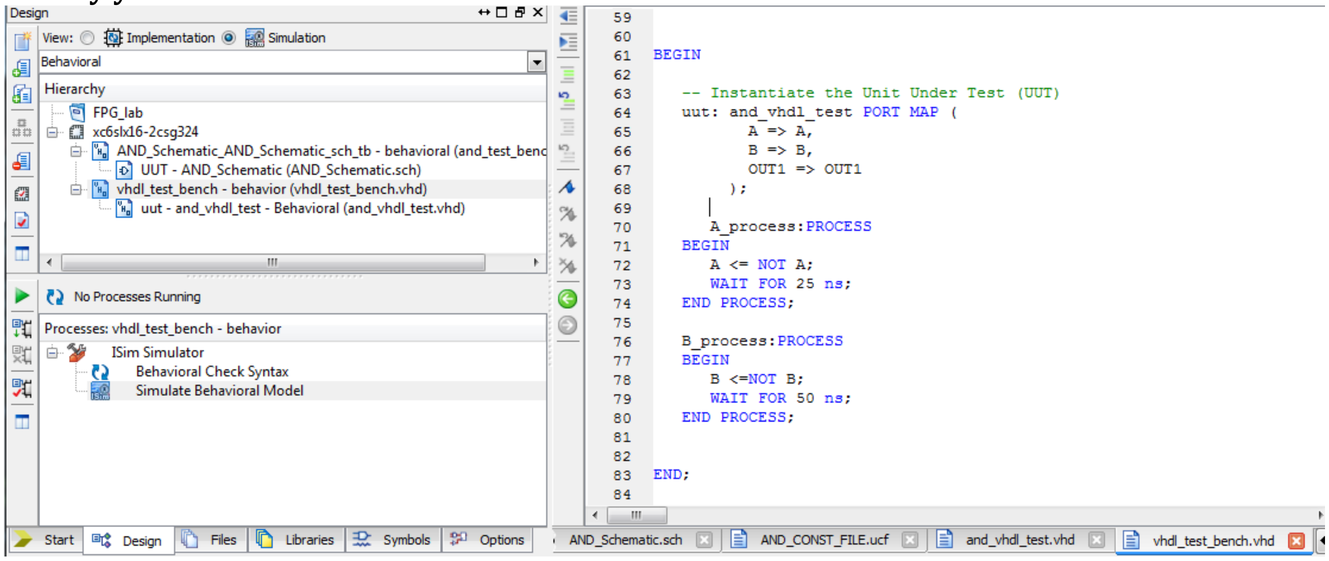
```vhdl
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- 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.

Switch to the Simulate Mode, Right click your VHDL file, and create a new test bench file, just as you did for your schematic before.
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.vhd
vhdl_test_bench.vhd

```


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.

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

4 Example Code:

4.1 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;
```

4.2 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;
```

4.3 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;
```

5 References

This lab was based off of Labs 1 and 2 developed by the Configurable Space Microsystems Innovations and Applications Center.
http://www.cosmiac.org/Projects_FPGA.html#Lab1