

## ECE 340 Building Hierarchy Lab

### Introduction

You will write a complete RTL (Register Transfer Level) description for the modules MY\_AND2 and MY\_OR2. These modules will be used to build the circuit shown in Figure 1, using a structural Verilog description of the top-level module AND\_OR.

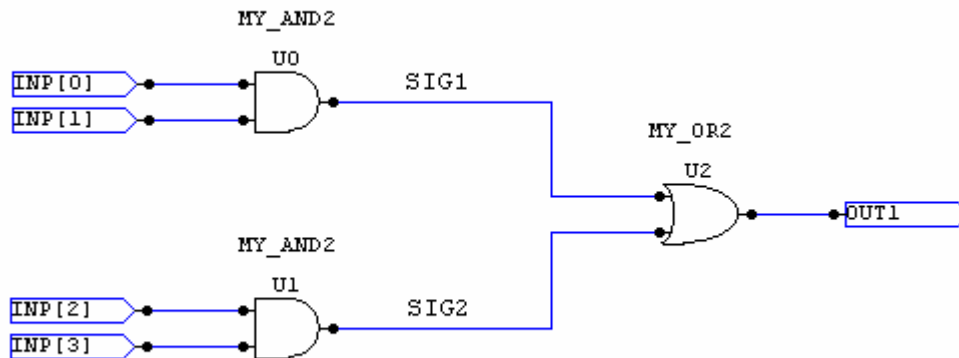


Figure 1: AND\_OR

Note: The *ISE 8.2i Quick Start Tutorial* will be useful during this lab exercise. Download it from <http://emp.byui.edu/fisherr/Tools.htm> or from our Blackboard course page under *Course Documents* → *Software Downloads*.

### Objectives

After completing this lab, you will be able to:

- Write RTL descriptions for simple gates
- Create a structural Verilog description for a simple circuit
- Build hierarchy by using Verilog
- Use the HDL editor in the ISE software

### Procedure

1. Start the ISE Navigator. See the *ISE 8.2i Quick Start Tutorial*.

2. Create a new project.

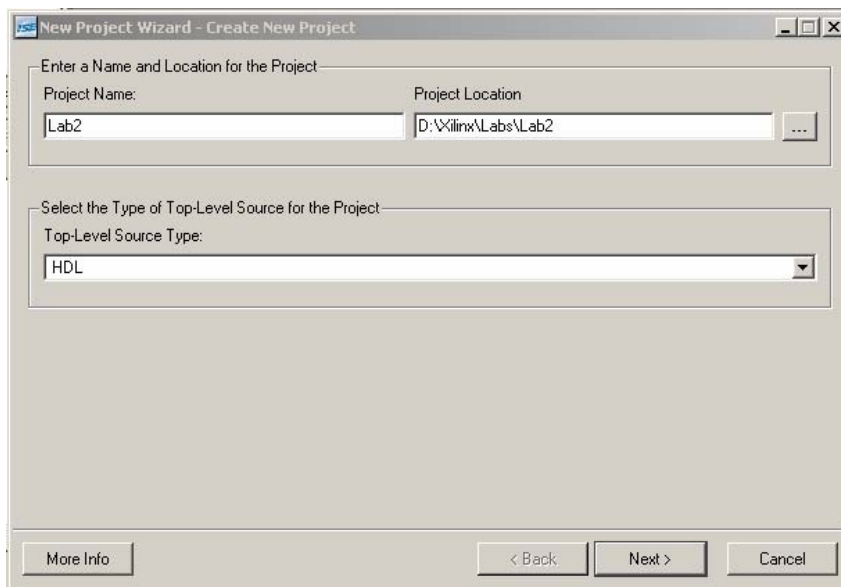
Select *File -> New Project*

Enter the following values in the *Create New Project* box:

Project Name: Lab2

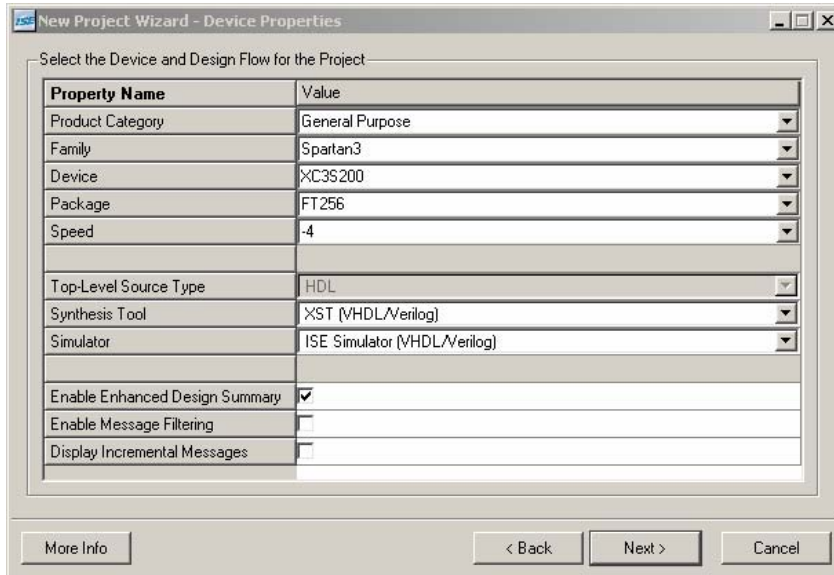
Project Location: C:\Xilinx\Labs\Lab2 (Note: If the directory does not already exist, it will be created automatically.)

Top-Level Source Type: HDL



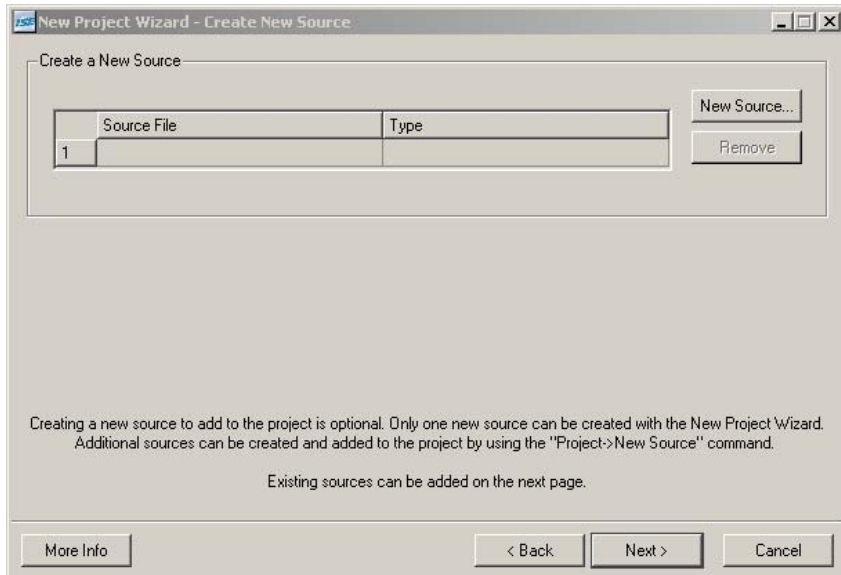
Click *Next*.

You will not be implementing this design in hardware, so the default values in the *Device Properties* box will be fine.



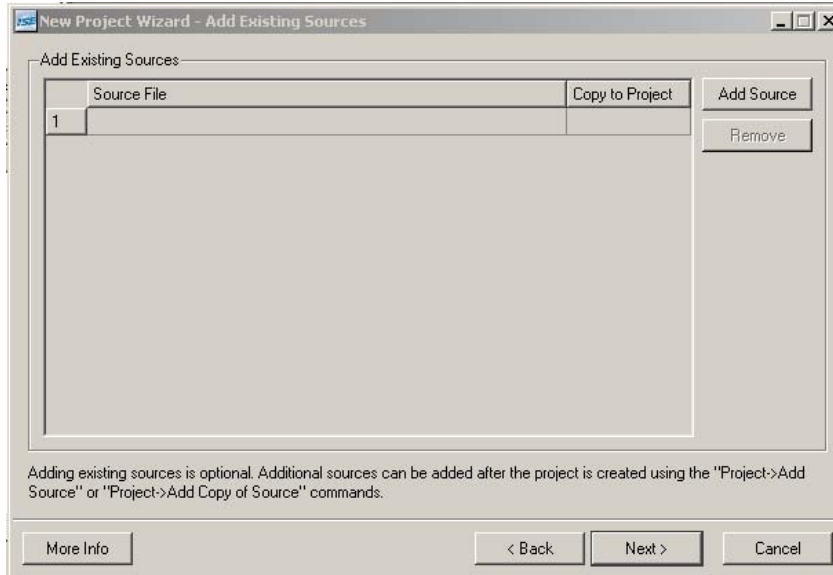
Click *Next*.

Leave all the fields blank in the *Create New Source* box.



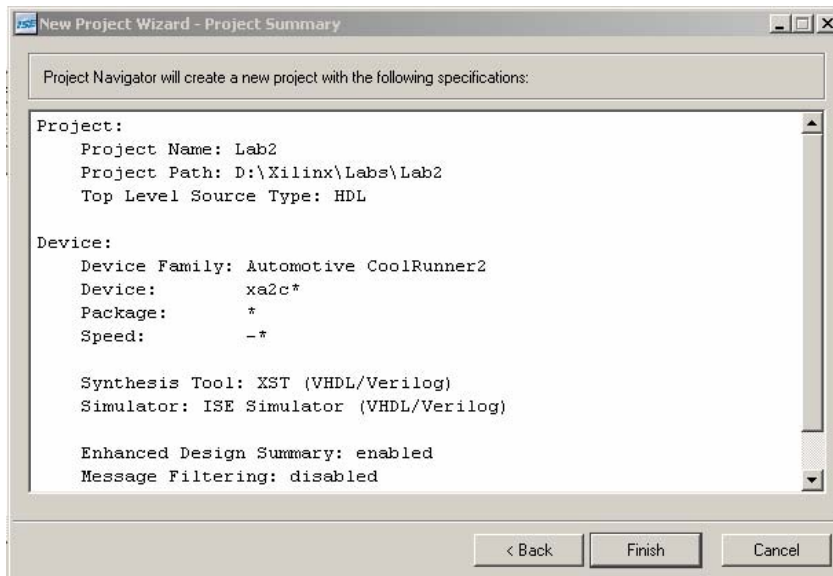
Click *Next*.

Leave all the fields blank in the *Add Existing Sources* box.



Click *Next*.

Click *Finish* in the *Project Summary* box.



3. Write the RTL description for MY\_AND2.

Select *Project* -> *New Source*

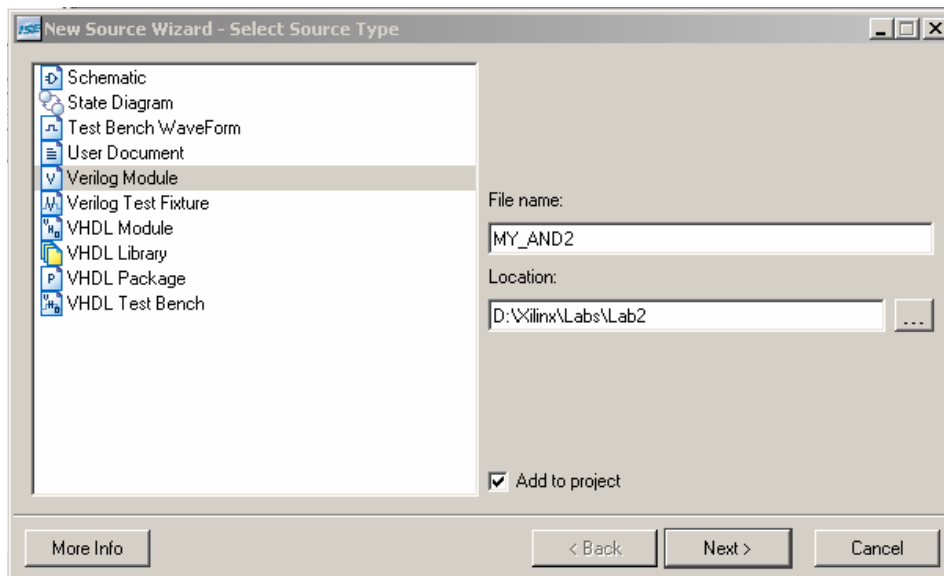
Select *Verilog Module* from the list in the *New Source* box.

Enter the following values in the *New Source* box:

File Name: MY\_AND2

Location: C:\Xilinx\Labs\Lab2

Make sure that *Add To Project* contains a check mark.



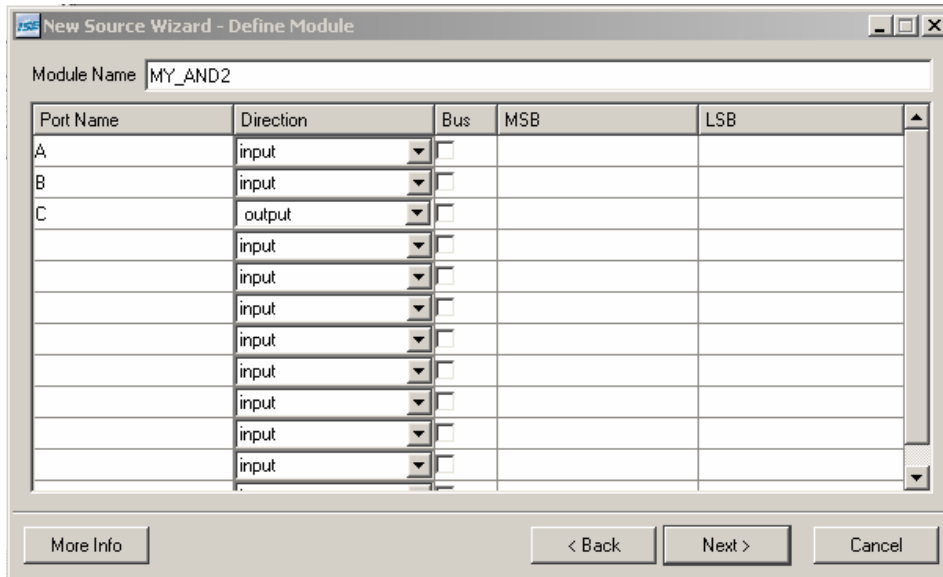
Click *Next*.

Enter the following values in the *Define Module* box:

Port Name: A      Direction: input

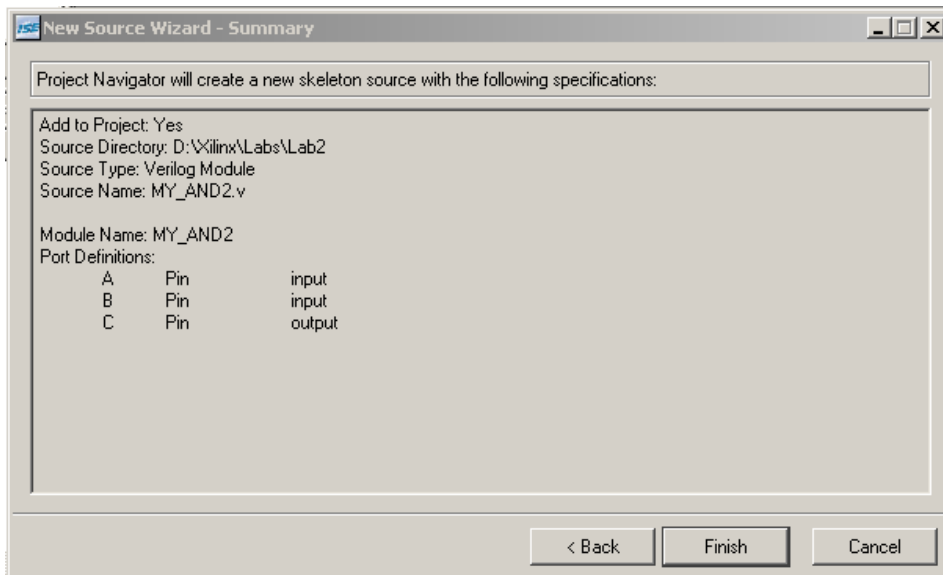
Port Name: B      Direction: input

Port Name: C      Direction: output



Click *Next*.

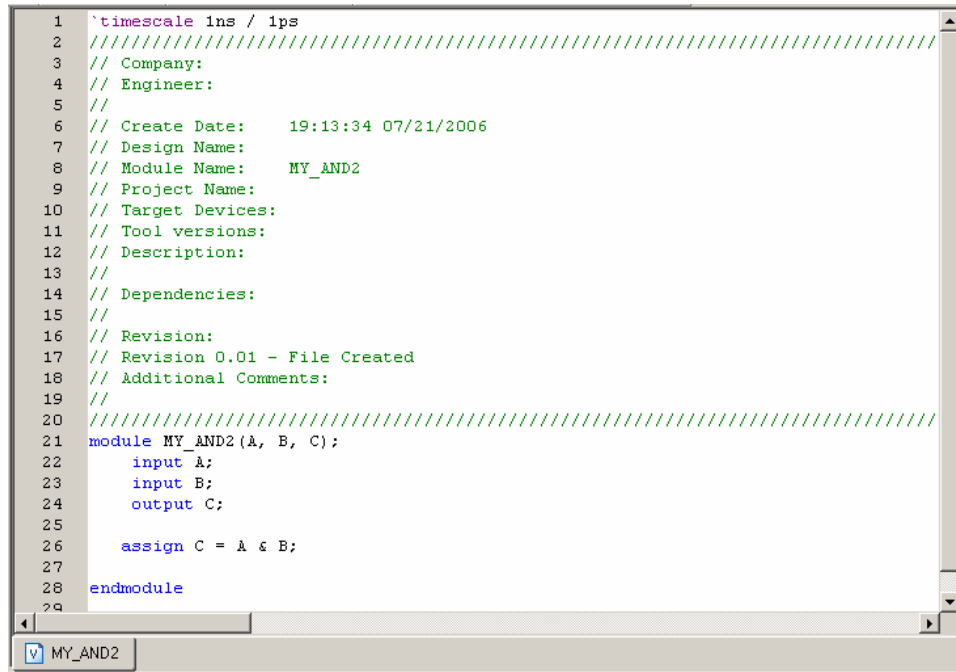
Click *Finish* in the *Project Summary* box.



The tool automatically creates the module declaration based on the user-defined data entered in the table.

Note: After exiting the wizard, all subsequent edits must be done in the HDL editor, including changes to items that were originally specified in the wizard.

Use the Verilog bitwise operators (& for AND) (| for OR) to describe the functionality of the AND and OR gates: that is, *assign C = A & B;*



```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
3  // Company:
4  // Engineer:
5  //
6  // Create Date:    19:13:34 07/21/2006
7  // Design Name:
8  // Module Name:    MY_AND2
9  // Project Name:
10 // Target Devices:
11 // Tool versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
21 module MY_AND2 (A, B, C);
22     input A;
23     input B;
24     output C;
25
26     assign C = A & B;
27
28 endmodule
29
```

Click on the disk icon to save the file.

4. Write the Verilog RTL description for the MY\_OR2 module.

Repeat the steps in # 3, above, to create MY\_OR2.v

This time, insert “*assign C = A | B;*” for the OR function in the HDL editor.



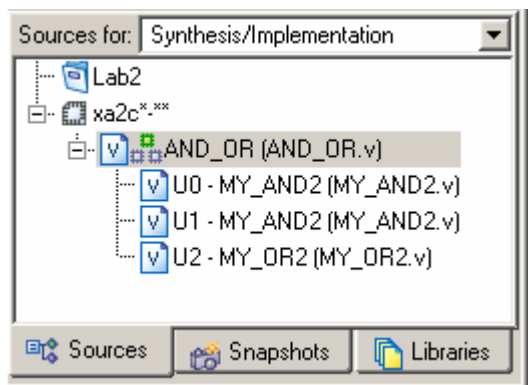
Within the AND\_OR top-level module, declare and instantiate the lower-level modules MY\_AND2 and MY\_OR2.

The internal wires that connect the outputs of the two MY\_AND2 components to the inputs of the MY\_OR2 component must be declared.

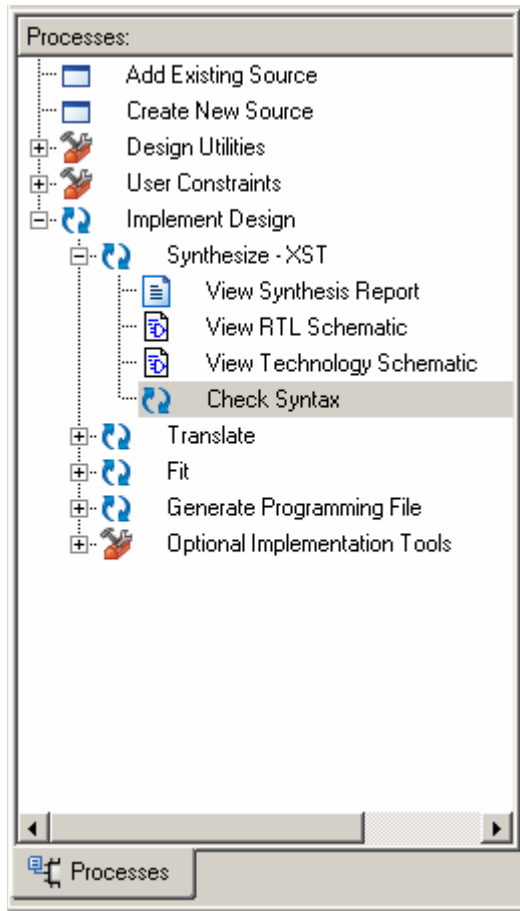
```
1  `timescale 1ns / 1ps
2  ////////////////////////////////////////////////////////////////////
3  // Company:
4  // Engineer:
5  //
6  // Create Date:    00:18:04 07/22/2006
7  // Design Name:
8  // Module Name:   AND_OR
9  // Project Name:
10 // Target Devices:
11 // Tool versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 ////////////////////////////////////////////////////////////////////
21 module AND_OR(INP, OUT1);
22     input [3:0] INP;
23     output OUT1;
24
25     wire SIG1, SIG2; // Internal signal wires used to connect gates.
26
27     // Declare and instantiate lower-level modules MY_AND2 and MY_OR2.
28
29     MY_AND2 U0 (.A(INP[0]), .B(INP[1]), .C(SIG1));
30
31     // The other two go here.  Finish them yourself.
32
33 endmodule
```

6. Perform a syntax check.

Select *AND\_OR.vin* the *Sources* window.



Expand *Synthesize - XST* in the *Processes* window, and double-click *Check Syntax*.



Correct any errors.

Note: Performing a syntax check does not identify all problems with the design source files. When the design is elaborated for synthesis and the complete hierarchy is resolved, other errors may be reported.

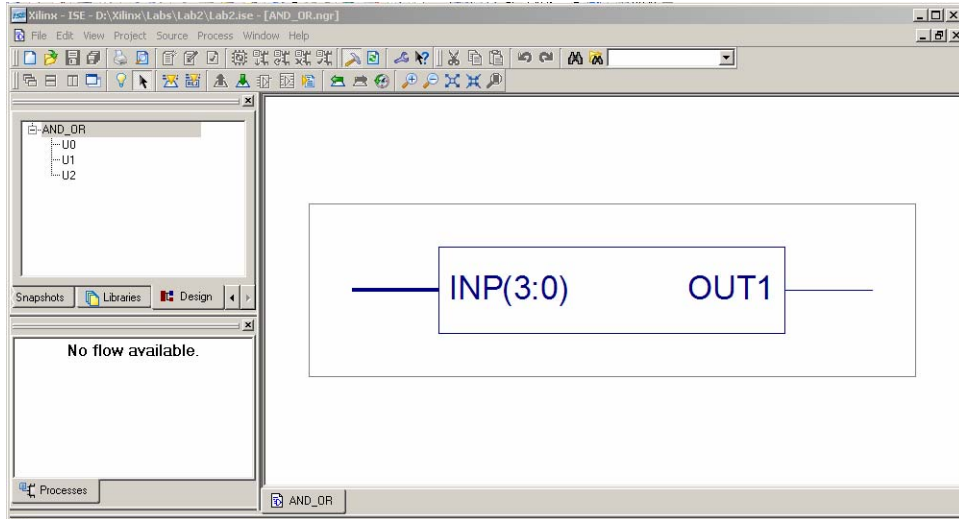
7. Generate a schematic to verify the structural integrity of the design.

Note: It is wise to visually ensure that the MY\_AND2 and MY\_OR2 components, along with the SIG1 and SIG2 wires, are all properly connected in the hierarchical model. Generating a schematic also causes the design to be elaborated and synthesized, so any errors reported at this stage must also be corrected.

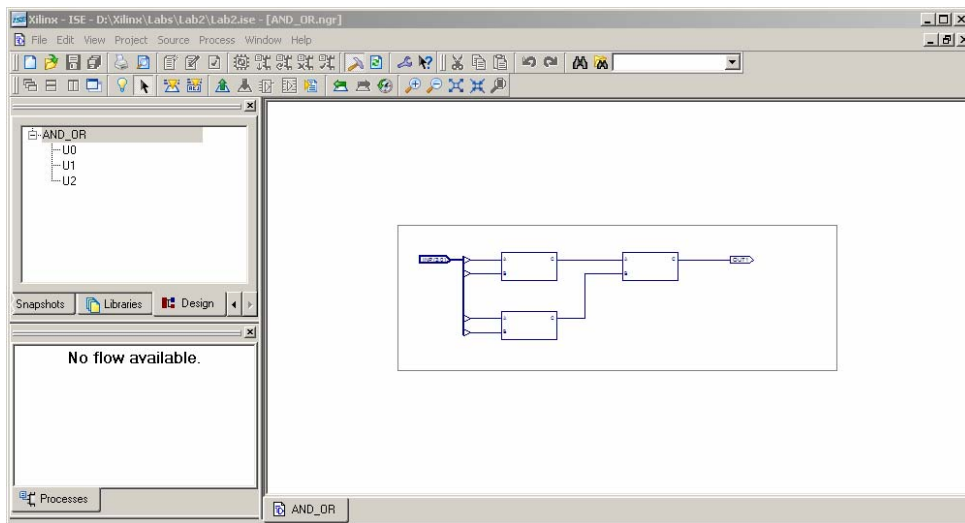
Select *AND\_OR.v* in the *Sources* window.

Select *View RTL Schematic* in the *Processes* window.

The following schematic should appear.

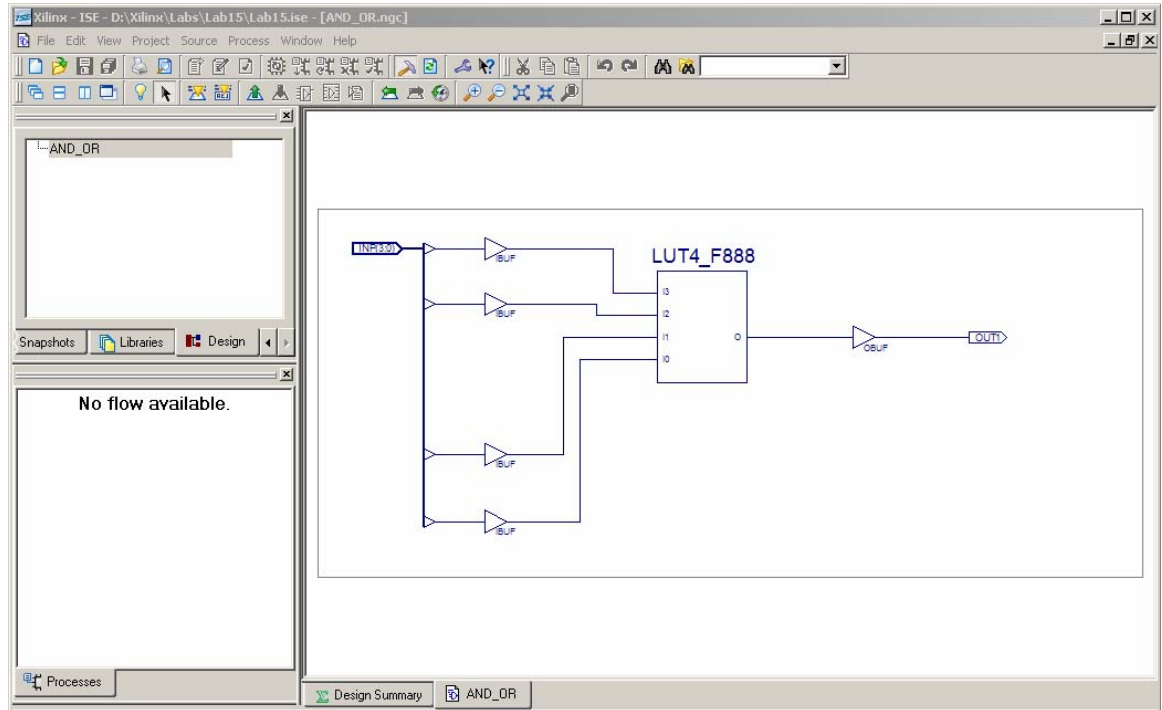


Double-click on it to “push down” the AND\_OR top-level block and reveal the lower-level module interconnections.



If the schematic does not look like this, fix the errors.

To view the technology-specific implementation, select *View Technology Schematic* in the *Processes* window.



This view shows the details of the SRAM-based LUT in the Xilinx FPGA.

## Summary

This lab established the basic practices and techniques that will be applicable for all Verilog coding. Simple low-level gates were created and then referenced hierarchically in an upper-level module. Internal wires were also declared to connect the lower-level modules.

For your lab report:

- Print the source code for MY\_AND2.v
- Print the source code for MY\_OR2.v
- Print the source code for AND\_OR.v
- Compare the circuit development process using a schematic capture-based CAD tool, such as Cadence, to an HDL-tool, such as Verilog. Discuss the advantages and disadvantages of each.