

Digital Logic Design – ECEN 3233

Oklahoma State University

James E. Stine, Jr.

Lab 5: Arithmetic and ALUs

Introduction

In this lab, you will design a 32-bit Arithmetic Logic Unit (ALU) that is described in Chapter 5 of the text. ALUs are traditionally the heartbeat for most digital systems, since they perform the basic operations for most systems. Your ALU will also become an important part of your final project. In this lab you will design an ALU in Verilog. You will also write a Verilog testbench and testvector file to test the ALU.

Remember to read through the entire lab before beginning, and refer to the “What to turn in section” before you start.

Background

You should already be familiar with the ALU from Chapter 5 of the textbook. The design in this lab will demonstrate the ways in which Verilog encoding makes hardware design more efficient. It is possible to design a 32-bit ALU from 1-bit ALUs (i.e., you could program a 1-bit ALU incorporating your full adder from Lab 1, chain four of these together to make a 4-bit ALU, and chain 8 of those together to make a 32-bit ALU.) However, it is altogether more efficient (both in time and lines of code) to code it succinctly in Verilog.

1) Verilog code

The only hardware you need to design for this lab is the 32-bit ALU. Your 32-bit ALU will have inputs A[31:0], B[31:0], and F[2:0], and outputs Y[31:0] and Zero. A and B are the 32-bit data inputs. F is the 3-bit control input that determines the function the ALU performs. Y is the 32-bit output result. The output Zero should be TRUE if Y is equal to zero.

Open Xilinx Project Navigator ISE 9.2i. You can create a new project for this lab using **File**→**New Project** as before. Name the project something like “lab5_xx” (where xx are your initials). The Top-Level Module needs to be HDL (Verilog). Choose “Modelsim-XE Verilog” as the simulator and XST as the synthesis tool. Hit next until finished. Once you have created a new project in Project Navigator, create a Verilog file for your 32-bit ALU by choosing **Project**→**New Source**→**Verilog Module**. Name the module alu32 and click Next. Under Port Name, enter A, B, F, Y, and Zero. Set these ports to the bus widths and direction (input or output) shown in Figure 1.

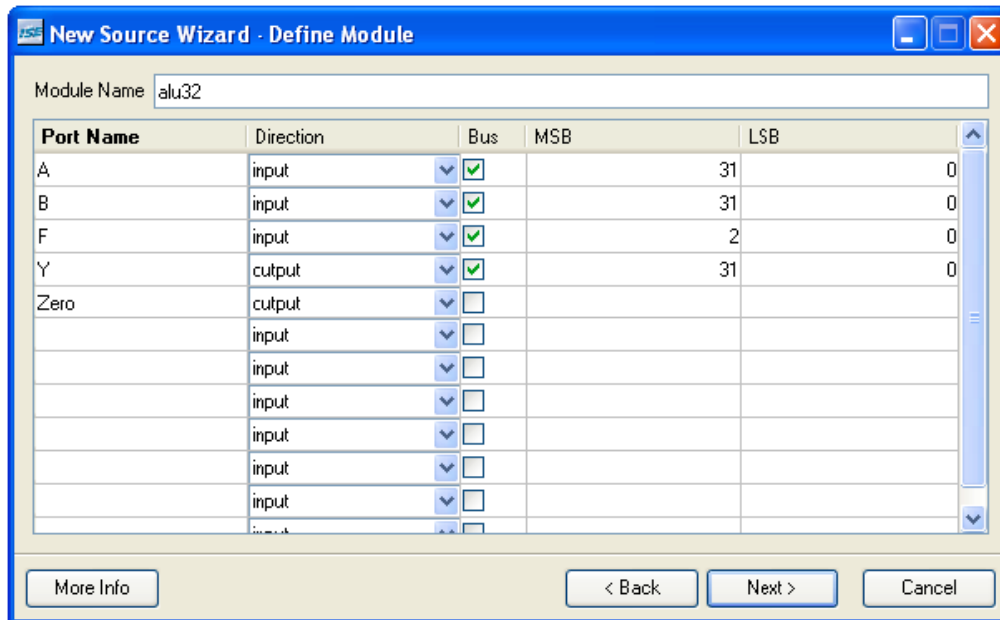


Figure 1. alu32 inputs and outputs

Now click Next, and Finish. An outline of your Verilog module is created called alu32.v. If it does not show up in the Xilinx window, double-click on alu32.v in the Sources pane.

You can use any of the Verilog constructs discussed in class or in the IEEE 2001 Verilog Standard. For example, you might find the conditional operator or the addition operator (+) useful.

Remember:

- Verilog is case sensitive.
- You can create multi-bit internal signals (like temporary variables) by declaring wires at the top of your module. For example, you might write the following code:

```
wire[31:0] ANDresult;

assign ANDresult = A & B;
```

- Verilog commands such as & (AND), | (OR), ^ (XOR), ~ (NOT), etc. work on individual bits independently (i.e., 8'b00010101 | 8'b10010110 produces the result 8'b10010111).
- An adder is a relatively expensive piece of hardware. Be sure that your code implies no more than one adder. That adder is shared to perform ADD, SUB, and SLT by multiplexing the inputs and outputs.
- Don't forget to implement the zero detection logic (e.g., produce the Zero output).

Once you have completed your Verilog module for the 32-bit ALU, you are ready to save and synthesize your design. Recall that synthesis turns your HDL into hardware. You should use the built-in synthesis tool, XST, as in Lab 4. First, make sure the desired synthesis tool is selected by right-clicking on the high level project in the Sources pane and select Properties, as shown in Figure 2.

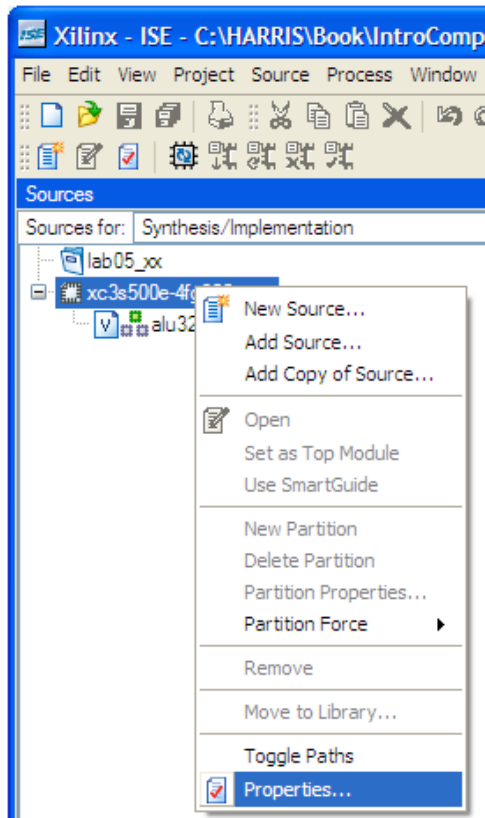



Figure 2. Modifying project properties

In the window that pops up, make sure that in that your desired tool, “XST (VHDL/Verilog)”, is selected as the Synthesis Tool and click “OK”. The following steps are the same regardless of which tool you are using.

Highlight the alu32.v module in the Sources window, double-click on Synthesize – XST in the Processes window. After synthesis is complete, there should be a green check mark or a yellow exclamation mark to the left of “Synthesize – XST” in the Processes pane. A yellow exclamation mark indicates that there are warnings. View the warnings by clicking on the Warnings tab in the Transcript console. If a warning needs to be fixed, fix it and then resynthesize. Also recognize which warnings can be ignored. For example, “Property “use_dsp48” is not applicable...” can be ignored.

View the RTL schematic by expanding “Synthesize – XST” in the Processes pane (click on the “+” next to it) and click on “View RTL Schematic”. You can double-click on the symbol to view lower-level circuits. Click on  to go back to higher levels in the hierarchy. Does the circuit look as you expected it to look? Write a few sentences describing how it is the same or different from what you expected. Now view the technology schematic. For now, just observe what is there.

2) Simulation and Testing

Now you can test the 32-bit ALU in the simulator. We would like you to develop an appropriate set of test vectors to verify the basic functionality. Complete Table 1 (on the next page) to verify that all 5 ALU operations work as they are supposed to. Note that the values are expressed in **hexadecimal** to reduce the amount of writing.

Test	F(2:0)	A	B	Y	Zero
ADD 0+0	2	00000000	00000000	00000000	1
ADD 0+(-1)	2	00000000	FFFFFFFF	FFFFFFFF	0
ADD 1+(-1)	2	00000001	FFFFFFFF	00000000	1
ADD FF+1	2	000000FF	00000001		
SUB 0-0	6	00000000	00000000	00000000	1
SUB 0-(-1)		00000000	FFFFFFFF	00000001	0
SUB 1-1		00000001			
SUB 100-1		00000100			
SLT 0,0	7	00000000	00000000	00000000	1
SLT 0,1		00000000		00000001	0
SLT 0,-1		00000000			
SLT 1,0		00000001			
SLT -1,0		FFFFFFFF			
AND FFFFFFFF, FFFFFFFF		FFFFFFFF			
AND FFFFFFFF, 12345678		FFFFFFFF	12345678	12345678	0
AND 12345678, 87654321		12345678			
AND 00000000, FFFFFFFF		00000000			
OR FFFFFFFF, FFFFFFFF		FFFFFFFF			
OR 12345678, 87654321		12345678			
OR 00000000, FFFFFFFF		00000000			
OR 00000000, 00000000		00000000			

Table 1. ALU operations

Instead of using a testbench waveform, as you have done in previous labs, you will write a Verilog test fixture to test your 32-bit ALU. First create a file called test_alu32.tv with all your testvectors. For example, the file for describing the first three lines in Table 1 might look like this:

```
2_00000000_00000000_00000000_1
2_00000000_FFFFFFFF_FFFFFFFF_0
2_00000001_FFFFFFFF_00000000_1
```

Hint: To make your testvector file easier for your testbench to read in and parse in hexadecimal, you might move the 1-bit Zero output to the beginning of each testvector. The first three lines in Table 1 would then look like this:

```
A_00000000_00000000_00000000
2_00000000_FFFFFFFF_FFFFFFFF
A_00000001_FFFFFFFF_00000000
```

You can create the test_alu32.tv file in any text editor, but make sure you save it as text only, and be sure the program does not append any unexpected characters (like .txt) on the end of your file. For example, in Wordpad select **File**→**Save As**. In the “Save as type” box choose “Text Document – MS-DOS Format” and type “test_alu32.tv” in the File name box. It will warn you that you are saving your document in Text Only format, click “Yes”.

Now create a Verilog testbench for your alu32 Verilog module. Select **Project**→**New Source**→**Verilog Test Fixture**. Name it test_alu32. Click Next, and choose alu32 as the source with which to associate the new Verilog test fixture. Click Next and Finish. The outline of a Verilog testbench called test_alu32 will open up in Xilinx.

Fill in the code needed to read in your testvectors from the test_alu32.tv file you already created, to compare the test vectors, and to output text indicating whether the tests failed or succeeded. You will probably also want to generate a clock to synchronize when you change the inputs and check the outputs. You might want to look at the example testbench in the textbook.

Now under the “Sources For:” pull-down menu, select Behavioral Simulation. Highlight test_alu32.v and expand ModelSim Simulator in the Processes tab, and run “Simulate Behavioral Model.” Check for any errors in the Transcript window. Fix any errors and rerun the Verilog test fixture (i.e. testbench). Your testbench should verify that the Y and Zero outputs match the calculations from your table. If the results don’t agree with your expectations, check and see if there is a typo in your test vectors file or a mistake in your completed Table 1. If you still have trouble obtaining the correct results, go back to your Verilog file and make sure each port or wire goes by only a single name (for example, B instead of b). Also make sure that all internal signals are declared as “reg” or “wire” in the module.

After changing your Verilog testbench, rerun the simulation and repeat the process until you are successful. If you have used the \$finish command in your Verilog test fixture, ModelSim will give you a window that says, “Are you sure you want to finish?” Click No, so you can view the waveform.

What to Turn In

Please turn in each of the following items (in the following order and clearly labeled):

1. **Please indicate how many hours you spent on this lab. This will not affect your grade, but it will be helpful for calibrating the workload for next semester's labs.**
2. Your table of test vectors (Table 1).
3. Your test_alu32.tv file
4. Printouts of your Verilog code. (Include any instantiated modules should you choose to use them.)
5. A printout of your RTL schematic along with a few sentences of observations about the RTL schematic.
6. Printouts of your Verilog test fixture (your testbench).
7. Printouts of your test waveforms. Make sure these are readable and that they're printed in hexadecimal. Your test waveforms should include only the following signals in the following order, from top to bottom: *F, A, B, Y, Zero*.
8. If you have any feedback on how we might make the lab even better for next semester, that's always welcome. Please submit it in writing at the end of your lab.