Testing the ALU Using a Test Bench

Given the complexity of large scale digital designs, more than 50% of the total design time for a new device is spent on testing and validation.

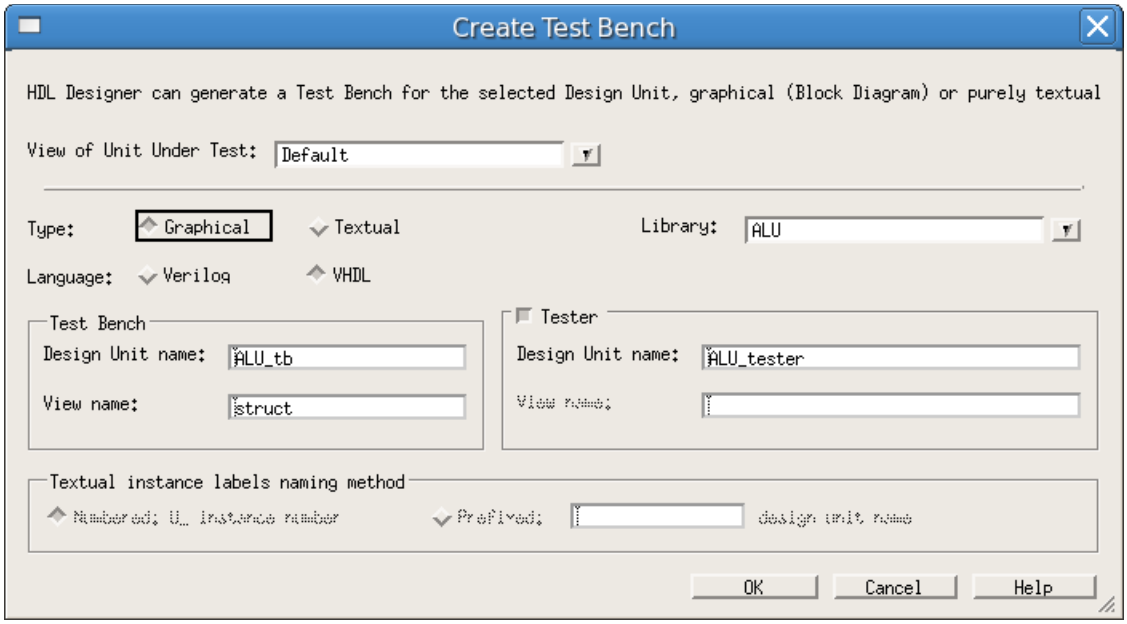
So far you've used Modelsim scripts to drive the inputs of your designs and used the wave window to verify the results. This worked for small-scale designs but manually reading the wave window becomes unmanageable as the number of test cases grows. A better solution is to design a new component for each design-under-test called a *tester*, that can both provide test inputs and verify the accuracy of the resultant outputs. Together, these two components form a *testbench*.

The tester can use the VHDL ASSERT statement to verify the outputs of the design-under-test. The syntax of the ASSERT statement is:

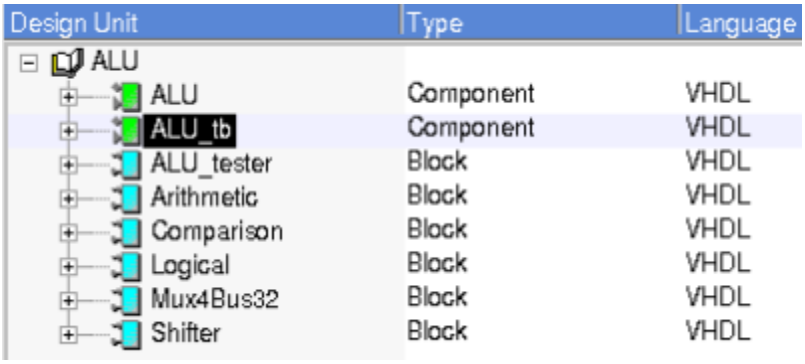
ASSERT <*condition*> REPORT <*error message for failure*> SEVERITY <*level*>

HDL Designer can automatically generate test benches, but it's up to you to implement the tester.

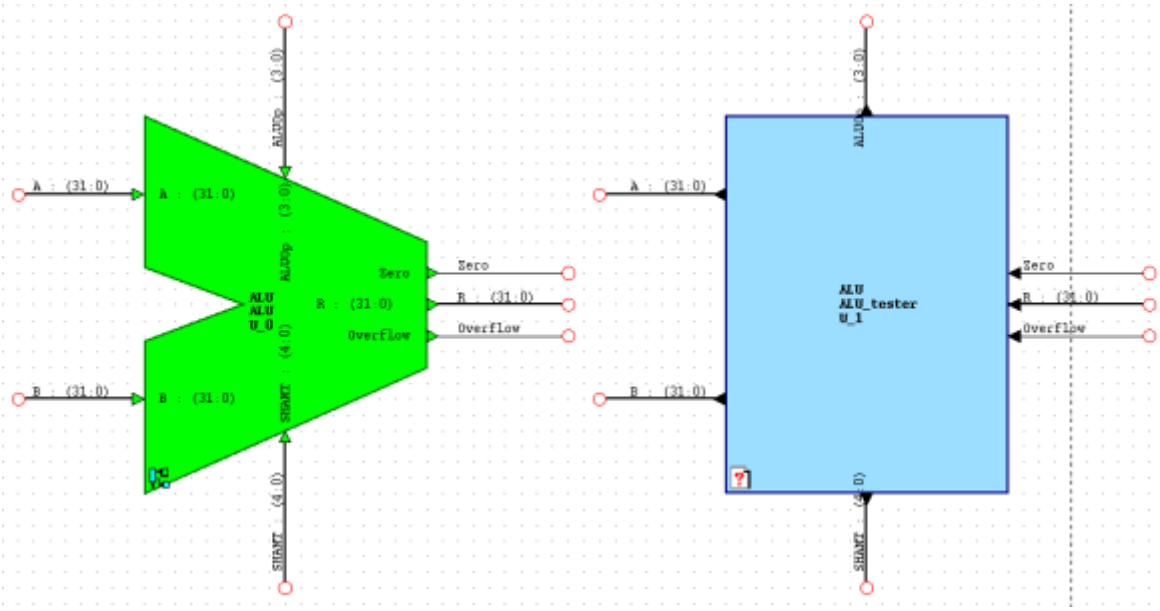
Let's begin by creating our test bench that will facilitate our testing of the ALU. This can be achieved with the following two methods. Firstly, left-click and select the **ALU** component in the **Design Manager**. Then, you may either click **File > New > Test Bench** via the menu or right-click the **ALU** component and select **New > Test Bench**. Click the **OK** button to proceed.



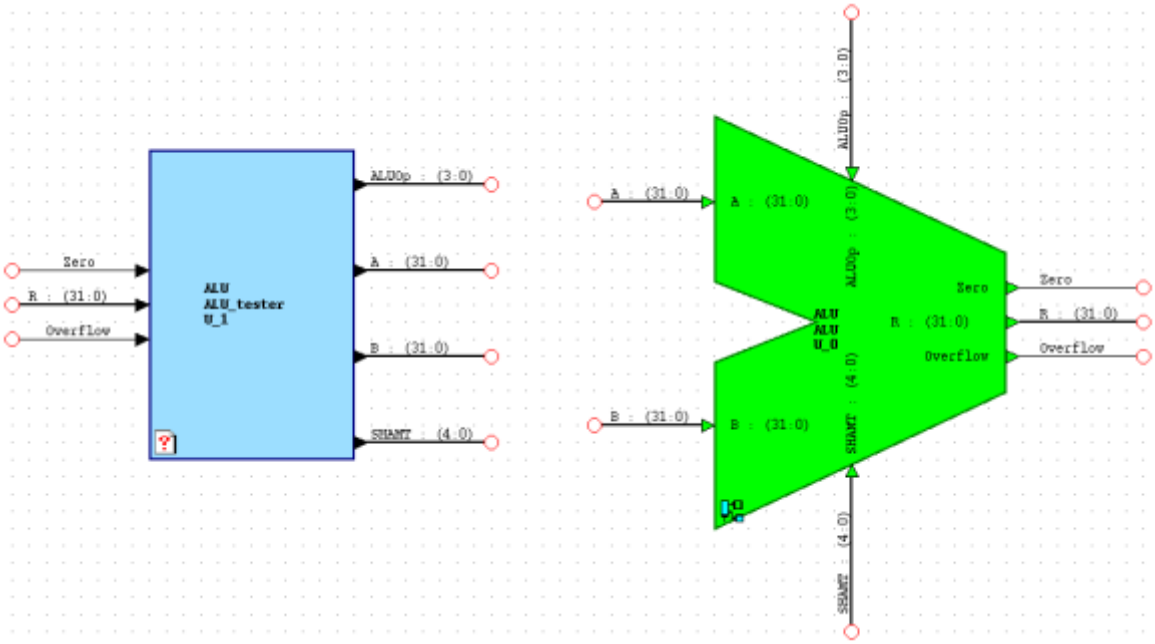
You should now see a green component created in your design named "ALU_tb". Double-clicking on "ALU_tb" will show you the test bench that was generated according to your design.



As you can see from the block diagram that was created, an instance of your ALU has been instantiated along with a blue sub-block. Notice that the signals that are outputs from your green ALU design component have the same name as those which are input into the blue sub-block called **ALU_tester** and vice-versa. We have not seen this before, but another feature of the **Block Diagram Editor** is that signals of the same name are the same signal, even if they are not visually connected together.



You may resize and rearrange your test bench for better readability. The figure below demonstrates a reorganized view of the two blocks:



Although our Test Bench has been automatically generated for us, we still need to add the logic that it requires to test our components. We shall now proceed to creating the ALU_tester's functionality.

Creating the ALU tester block

The purpose of the ALU_tester unit is to provide input data for the inputs of the ALU and to verify the resulting output data from the ALU is correct. There are several possible approaches for this. One approach is to randomly generate input data and then check the ALU outputs to verify correctness.

For example, we might randomly assign A = X“00001000”, B= X“00002000”, ALUOp = “0100”, and SHAMT = “10101”. Given these inputs, we expect the ALU to add A and B and ignore the value of SHAMT. Therefore, we would ASSERT that R=X“00003000”, Overflow=’0’, and Zero=’0’. Failing any of these assertions would suggest that the addition part of the ALU has design errors.

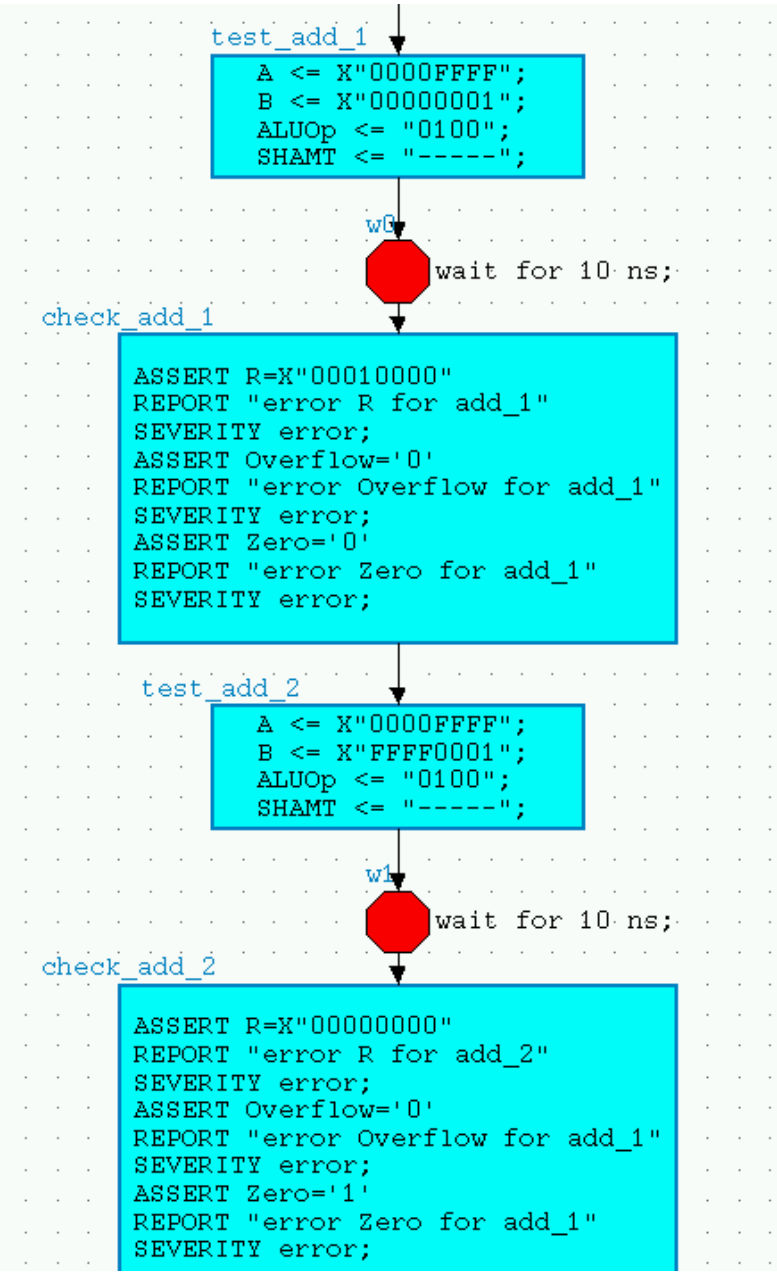
This testing methodology is called **black box testing**, because the test cases are selected as if the tester does not have knowledge of the internal design of the ALU. Black-box testing is usually used for performance testing and stress testing.

However, since we designed the ALU, it is more appropriate to use the **glass box testing** methodology and individually test each of the data paths in the ALU. Specifically, we must test each of the 13 ALU operations, as well special cases for each operation.

Our blue sub-block currently has no view associated with it. Double-click on the blue block tester so that you may create a new view. From the window, select **Graphical View** > **Flow Chart**, click the **Next** button, then click the **Finish** button.

Our tester will run through a series of tests that consist of driving the inputs, waiting for 10 ns to all the outputs to be updated, followed by an ASSERT/REPORT/SEVERITY block to test the outputs.

Since you're already familiar with the flow chart editor from the shifter design, we'll just show you picture of how each of the tests will look in the editor.



Once you have finished with ALU tester design, it needs to be compiled into a ModelSim simulation file. This can be done in one step by highlighting the **ALU_tb** component in the **Design Manage**r and using the ModelSim design flow button 

Good luck!