Introduction to Active-HDL

Tutorial #2 – Hierarchical Designs and Test Fixtures

This tutorial will use the 1-bit full adder you designed in Tutorial #1 to construct larger adders. This will introduce the concept of hierarchy to use simple components to construct more complex components. In this tutorial, you will build a 4-bit adder/subtractor component using the 1-bit full adder you already designed. Since the 4-bit adder is implemented using another design component, this forms a simple design hierarchy. A design hierarchy is similar to a procedure call hierarchy in programming languages. The important difference is that each call or "instantiation" of a simpler component by a more complex component actually creates a new instance of the simple component. After completing this tutorial you will know how to:

- Design using hierarchy.
- Test components using test fixtures.
- Use block symbols in Active-HDL.
- Use buses.
- Run simulations with multi-bit signals using a test fixture.
- (Optional) Create and connect arrays of components.

Start Active-HDL

1. Open Active-HDL.
2. Select the "Open existing workspace" option and select your workspace from Tutorial #1 in the window, or click the "More…" button to attach and select it.

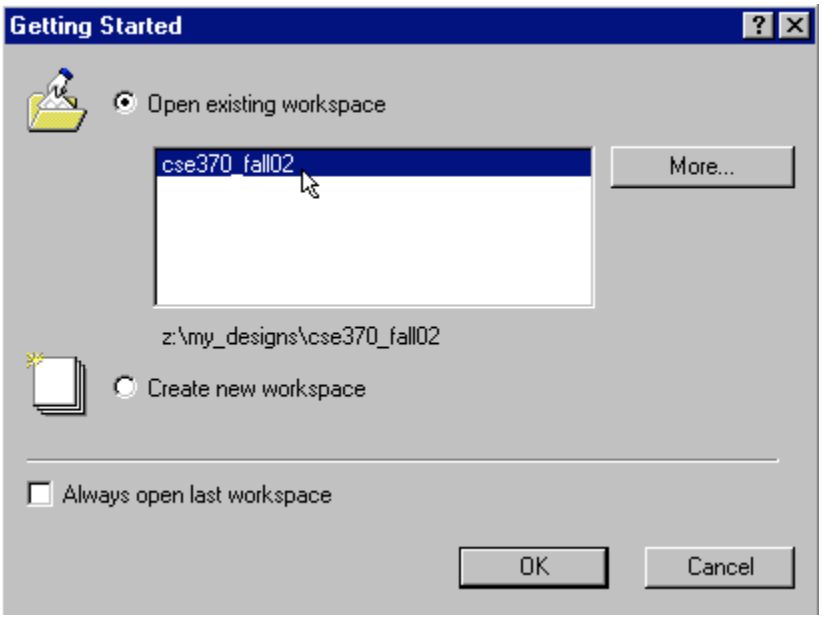

Figure 1

3. Click OK.

**** PART 1 ****

Test Fixtures

Test fixtures are used to automatically test and/or simulate a design or a component of a design. A test fixture is usually written in Verilog, which is a powerful hardware-oriented programming language. The test fixture drives the input signals of the design you are testing and samples its outputs. In this way, the fixture can stimulate your circuit and observe your circuit's corresponding output to verify its correctness. We are particularly interested in "self-checking" test fixtures that check the outputs and report an error automatically. This reduces the headache of analyzing waveforms in complex circuits.

Before using components as part of other components, you should test them, just like you test procedures before using them. In Tutorial #1, you used a simulation and a waveform to verify your 1-bit full adder. Test fixtures will not eliminate the need to use the debugging skills you learned in the first tutorial, but they will make verifying the correctness of a design faster and easier.

At some point when you have learned Verilog, you will be writing your own test fixtures. For this tutorial, and for most of the class problems, we will provide you with test fixture files to test the components you design. (It's pretty easy to follow these examples to generate your own Verilog test fixtures.) The code provided will test a number of cases for each component and print a series of messages to help you find errors, if any. These messages will be printed to the Console and a text file.

1. Download the 1-bit full adder full adder test fixture FA_tf.v and place it in a folder you'll remember (e.g., on your Z: drive).
2. Double-click "Add New File" in the Design Browser. Now, click the "Add Existing File" button.
3. Navigate to the folder containing the file, check the "Make local copy" box (See Figure 2), and click the "add" button.
4. Notice it is added to the design hierarchy in the Design Browser. We use the naming convention: <design name>_tf.v to name test fixtures so that it is easy to associate test fixtures with the components they test.

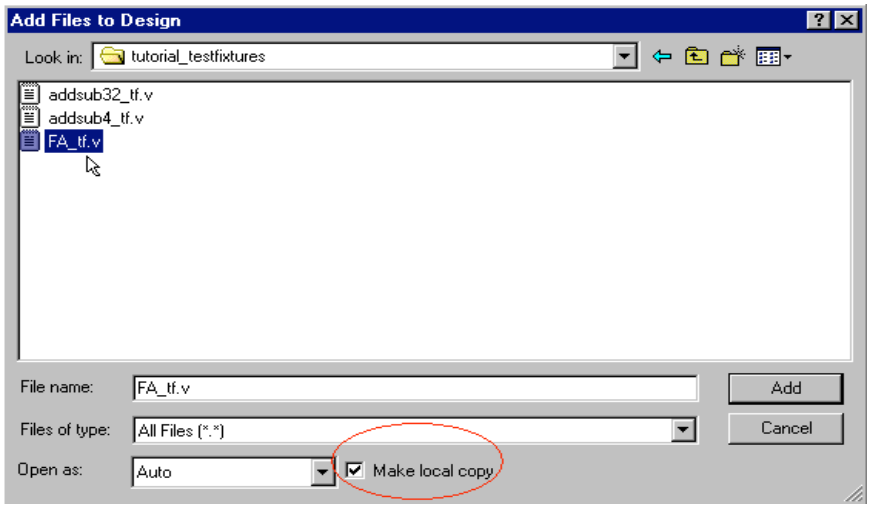

Figure 2

5. Double-click the test fixture in the Design Browser to open this file in the main window.
6. You do not need to understand the code completely, but read through the code and try to understand what it is doing and why it is doing it. If you renamed the file, change the title and the module name to reflect this change.
7. (**Optional**) You may also change the name of the text file that the test fixture prints to (See Figure 3). We typically do not save the output to a text file unless we want to print it.
8. Once you have familiarized yourself with the code, save it and compile it. If you choose to modify this file, save the file after each modification, and recompile it after you have saved it. Repeat this section to add the test fixture for the 4-bit adder/subtractor component to the design.

```
//------------------------------------------------------------------------
//
// Title        : FA_tf
// Design       :
// Author       : YOUR NAME HERE
// Company      :
//
//------------------------------------------------------------------------
//
// File         :
// Generated    : Wed Aug 14 06:59:53 2002
// From         : interface description file
// By           : Itf2Vhdl ver. 1.20
//
//------------------------------------------------------------------------
//
// Description : Drives inputs for a 1-bit full adder and samples the
// outputs to verify the correctness of the design.
//
//------------------------------------------------------------------------
`timescale 1ns / 1ns


module FA_tf ( A ,B ,Cin ,Cout ,Sum );

input Cout ;
wire Cout ;
input Sum ;
wire Sum ;

output A ;
reg A ;
output B ;
reg B ;
output Cin ;
reg Cin ;


integer results;

initial begin

    results = $fopen("FA_tf_output.txt");
```

Figure 3

Block Symbols:

Active-HDL can create block symbols from your compiled schematic designs and Verilog source code files in the Symbols Toolbox. We will create block symbols for the full adder you designed earlier and the full adder test fixture. This portion of the tutorial will walk you through placing a block symbol for the full adder into a new schematic and connecting it to a test fixture to ensure that it functions properly.

Note: These instructions assume that the full adder design from the previous tutorial is available in your current workspace. If it is not, you will need to add it manually. In this case, it is probably best to just create an empty block diagram in the current workspace for the full adder, copy/paste the old design into it, and compile.

1. Add a new empty block diagram file called: **<design name>_test**. Do this by double-clicking "Add New File" in the Design Browser, select the "Block Diagram" option under the Empty Files tab, and enter the name in the "Name" field. Then, click OK.
2. Open the Symbols Toolbox. Find the name of your design, and expand your design's part list. You should see a "Units without symbols" list. Expand it and find the name of your full adder schematic. Select it, and notice that a block symbol appears in the bottom window of the Symbols Toolbox. Add this part to your schematic and notice that this component no longer appears in the "Units without symbols" list.
3. Repeat step 2 to add the symbol for full adder test fixture to the schematic.
4. You can edit block symbols by right clicking the part in the schematic and selecting the "Edit" option. (**Note**: if this option is not available, save, close, and reopen the file and try again.) While in edit mode, you can drag the pins to a new position within the symbol, add text, etc.
5. The test fixture will drive the input signals of the full adder and test the full adder's outputs. Connect the two units using wires. The outputs of the test fixture connect to the corresponding inputs of the full adder, and the inputs of the test fixture connect to the corresponding outputs of the full adder.

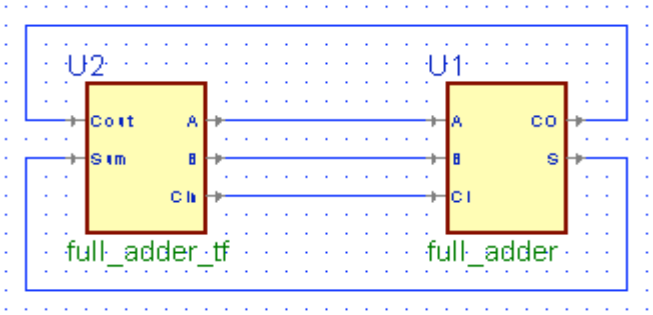

Figure 4

6. Now that the units are connected, save, run the check diagram tool, and compile the test schematic.
7. Set the test schematic as the top level, close any waveforms, initialize a simulation, and run the simulation for 80 ns. If you have forgotten how to do any of these steps, refer to Tutorial #1.

8. The Console, along with a text file, will contain the results of the simulation. If all eight cases passed, end the simulation, and continue with the tutorial. If not, there may be some problems with your full adder (see Tutorial #1 for examples on how to debug your design).

A Bit of Advice: Always assume that you will make mistakes. The right way to design is to write a test fixture for every component you design. If you change the design, all you have to do is re-run the test fixture to make sure you haven't screwed anything up. As a practical matter, we typically write test fixtures only when the components get reasonably complicated. Whether we'd write a test fixture for a full adder is mostly a matter of how optimistic we are. It's best to admit that you're going to make mistakes, and the sooner you find them, the easier your life is going to be.

**** PART 2 ****

Design a 4-bit adder/subtractor component

Figure 5 is an example of the schematic you will be designing in this portion of the tutorial. You already know how to place parts and connect them. However, this design uses buses as well as wires; therefore, you will learn how to use and connect buses. Additionally, you should gain a better understanding of how hierarchy is used to reduce the complexity of our designs.

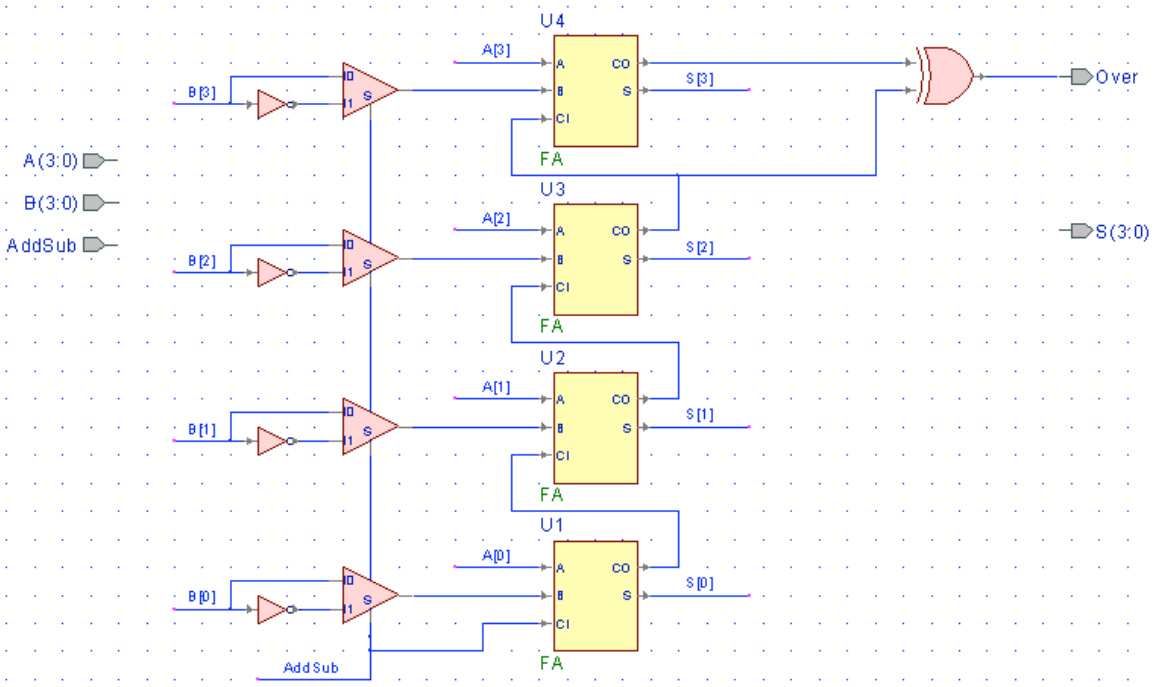

Figure 5

1. Add a new schematic block diagram file to the design and then create this schematic just like you created the schematic in Tutorial #1. You can do this using the Wizard, or by starting with an empty schematic (my favorite). The only thing you do not know how to do is to add the ports (terminals), specifically the bus ports. You can add these when you first create the block diagram with the Wizard. Or you can add the terminals in the schematic afterwards using the Terminals menu.

2. To add a multi-bit terminal while in the wizard, type in the name of the terminal and select whether it is an input or output terminal. Now in the "Array Indexes" fields, place a 3 in the field on the left (the "to" field), and a 0 in the field on the right (the "from" field). You can enter in the numbers or use the arrow buttons to the right of each field. The fields will read "from 0 to 3", or in other words, a 4-bit port (see Figure 6). To add multi-bit terminals in the schematic, click the drop down arrow next to the Terminals icon in the toolbar. Mouse over the options, select the desired terminal, and drop it into the schematic. Double-click the terminal in the design, and adjust the "Index Range" fields in the Terminal Properties window (not shown) to the desired number of bits.

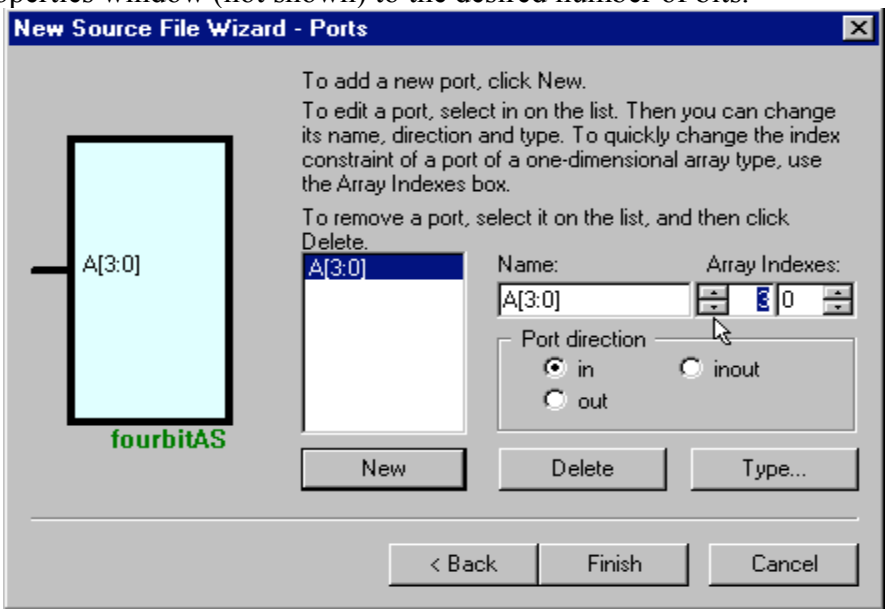

Figure 6

3. Using the methods described here and in Tutorial #1, add two 4-bit inputs named "A" and "B", add one 1-bit input named "AddSub", add one 4-bit output named "S", and a 1-bit output named "Over". Then, click Finish.

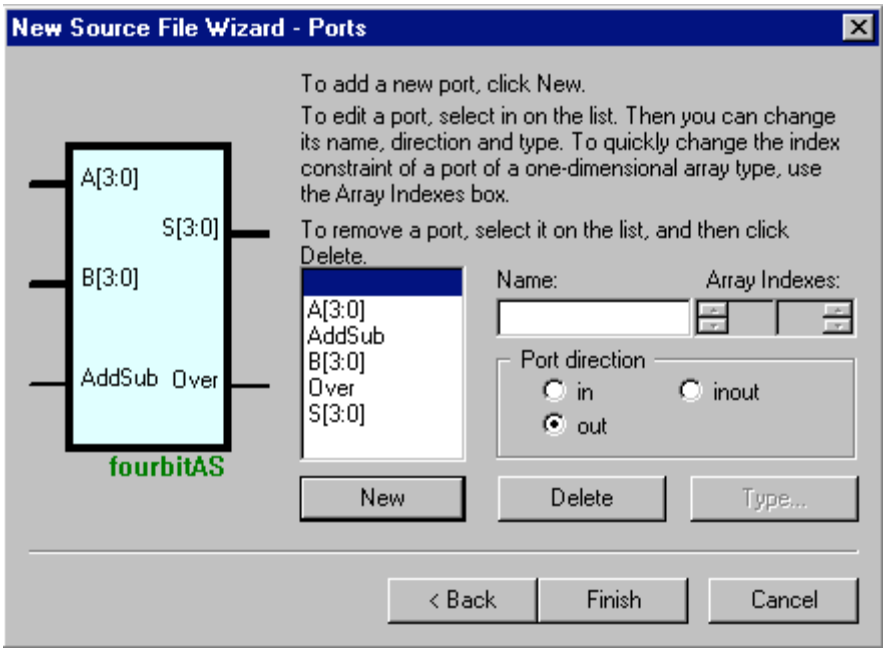

Figure 7

4. Place the parts from Figure 5 into the design (use the standard 370 class library), and make the appropriate wire connections. You will find the multiplexor under the name "MUX2", and the inverter is under the name "INV".

Using Buses

As you can see in Figure 5, you can use the naming connections described in Tutorial #1 for connecting buses as well as wires. This is the simplest method and thus the method we will use. There are other ways to create so-called bus taps. Since we don't like them, we won't describe them. But you can (and should) take a look at the online documentation, which explains other kinds of taps, as well as the method used in this tutorial. To view the online documentation, go to Help/On-line Documentation in the menu bar.

The steps provided below explain how to:

- Draw buses.
- Name buses.
- Move buses.
- Delete buses.

Drawing buses:

1. Click on the Bus icon in the toolbar.
2. Buses work just like wires. The only difference is that a bus contains several wires, named via indices. Click anywhere in the schematic and drag to the point where you want the bus to end. Clicking on a port or terminal will connect an end of the bus to that port or terminal. When connecting buses to pins and terminals of different width, the bus or the bus pins are connected so their leftmost bits are aligned. Just like wires, clicking on an empty space in the schematic will anchor the bus at that point, and double-clicking an empty space in the schematic will create an end to the bus at that point.
3. Press the Esc key to return to Select Mode.

Naming buses:

1. Double-click the bus segment to be named. This opens the Bus Properties window.
2. Type the name of the bus in the "Segment" field, and select the index range. By default, buses are 8-bits wide when drawn (see Figure 8 on the next page).

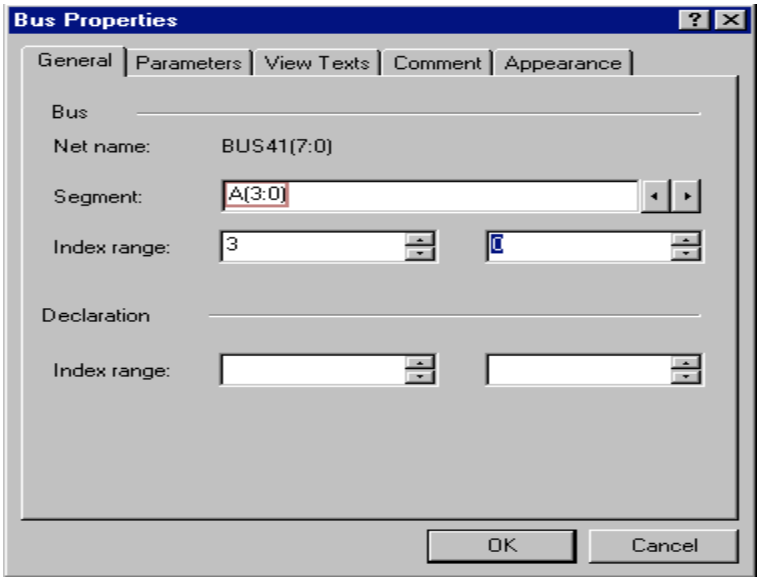

Figure 8

Moving and deleting buses are done the same way as with wires. Remember that Active-HDL uses the standard Window user-interface to select, move, copy, and delete items in the schematic.

For this assignment, you only need to know how to extract members of buses from the input terminals using naming connections and connect wires to multi-bit output terminals via naming connections as well. However, you can learn how to use wiretaps, and extract slices from buses by doing the optional portion of this assignment and reading the online documentation under "Help" in the menu bar.

Simulate the design using a test fixture

Your design should now look like Figure 5. Now you are ready to test your design. In the "Test Fixtures" section of this tutorial, you learned how to place a block symbol for a design into a new schematic along with a block symbol for a test fixture and how to edit block symbols. Download the test fixture for the 4-bit adder/subtractor called addsub4_tf.v. Save, run the check diagram tool, debug (if necessary), and compile your 4-bit adder/subtractor design and follow the steps described in the "Test Fixtures" section to connect the block symbol of this design to the 4-bit adder/subtractor test fixture in a new schematic. Remember to follow the naming convention. The remaining steps will guide you through running a simulation with your test fixture as well as adding multi-bit signals to a waveform.

1. Once you have finished connecting the test fixture block symbol to your design's block symbol, save, run the check diagram tool, and compile the new schematic. Your schematic should look like Figure 9.

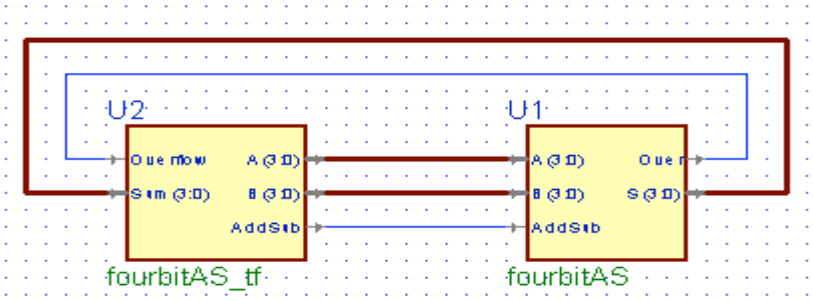

Figure 9

2. Set this schematic as the top-level, and initialize a simulation.
3. Close any open waveforms and open a new waveform by clicking the New Waveform icon in the toolbar.
4. With the "Structures" tab active in the Design Browser, expand the list of files under your new schematic file in the top window of the Design Browser.
5. Selecting the name of your design or the test fixture will show the corresponding symbol's signals in the lower window of the Design Browser. Multi-bit signals are preceded by a plus symbol.
6. Use the same methods you used in Tutorial #1 to add the inputs and outputs from the 4-bit adder/subtractor component. Notice that if you add a multi-bit signal all of its bits will be added to the waveform, or you can add specific bits by expanding the multi-bit signal's list and add only the desired bit. Your waveform should look something like Figure 10.

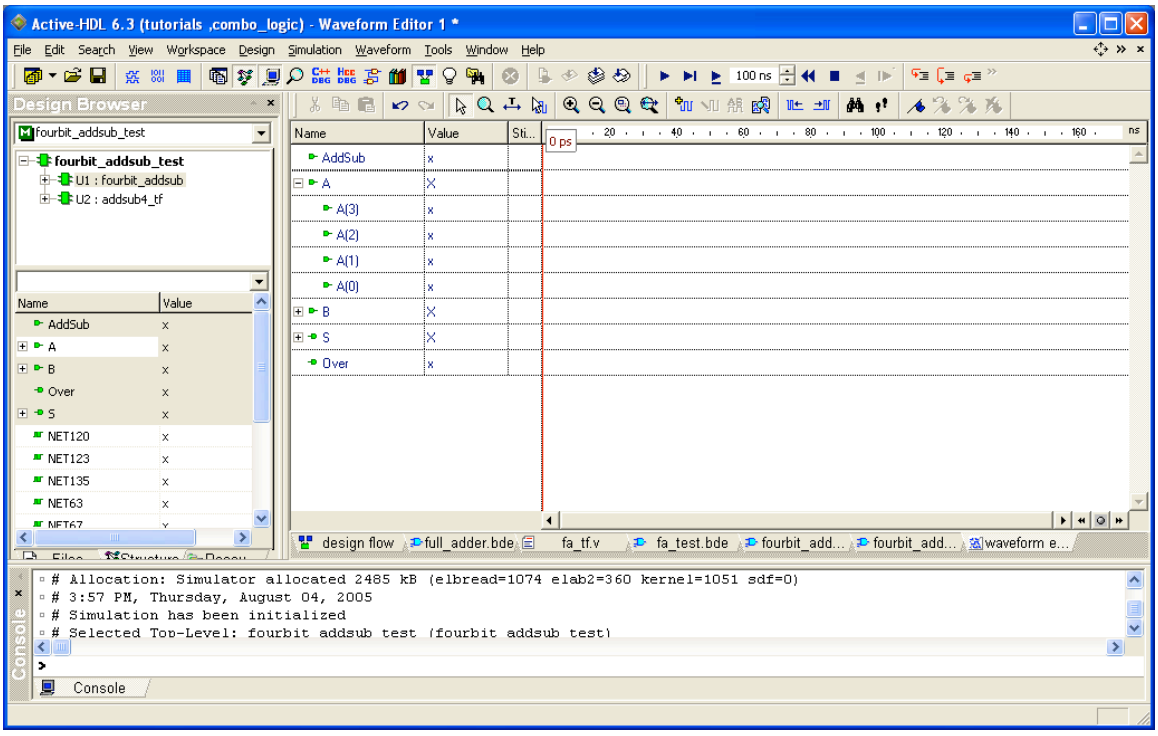

Figure 10

7. Run the simulation for 100 ns. Notice how the waveform shows the decimal values as well as the bit values for the individual signals (not shown). You could verify the correctness or incorrectness of your design using the waveform, but this would be confusing and easy to miss any inconsistencies. To make verifying your

design easier, the test fixture prints the results of the simulation in the Console and to a file. You may need to scroll up or down to see the results of each case tested in the Console, or go to your design folder and open the text file shown in Figure 3, which is located in the "Test Fixtures" section of this tutorial. Using the outputs and the waveform, verify the correctness of your design. Test fixtures typically only print errors.

8. Save the waveform using the naming convention <design name>_wv.
9. End the simulation, and close the waveform. You are now finished with the mandatory portion of the assignment. You may continue to the optional portion below (recommended), or go to the Concluding Remarks section on the last page.

Design a 32-bit adder/subtractor component (Advanced and optional)

In this portion of the tutorial, you will design a 32-bit adder/subtractor using a full adder. Designing this like we just designed the 4-bit unit would be tedious, painful and error-prone, even taking advantage of cut-and-paste techniques. Active-HDL, however, provides the ability to create arrays of components, which will help to make easy to read designs even when these designs consist of many components. Figure 11 shows the final schematic for the thirty-two-bit adder/subtractor component.

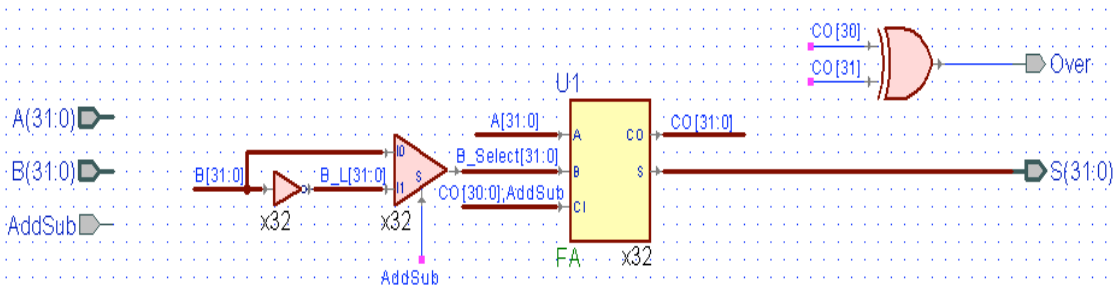

Figure 11

1. Using the Block Diagram Wizard, create a new block diagram schematic with two 32-bit inputs named "A" and "B", one 1-bit input named "AddSub", one 32-bit output named "S", and one 1-bit output named "Over".
2. Place your full adder component into the newly created schematic.
3. Right click the full adder, and select "Properties". This will open the Symbol Properties window.
4. Under the heading "Source file", select the "Create an array of (#) instances", and enter 32 in this field. (See Figure 12)

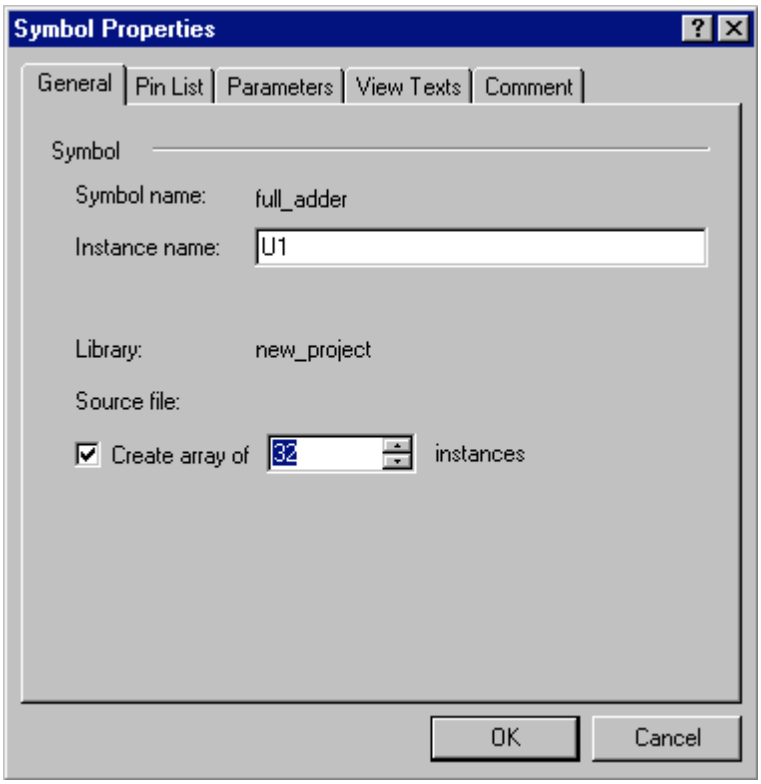

Figure 12

5. Under the "View Texts" tab in Figure 12, check the box next to the "Array Value" option. Then click the OK button.
6. Place an inverter and a 2:1 multiplexer from the standard class library into the schematic and repeat the above steps to create arrays of 32 for each component.
7. Use buses to connect the arrays of components to each other. When connecting the inverter to the multiplexer and the multiplexer to the full adder, you will need to name the buses and set their width to 32. Remember, buses by default are 8-bits and need to be named in order to adjust their width. (Refer back to Figure 11)
8. The bus that connects to the carry-in of the full adder is called a slice. Double-click the bus to bring up the Bus Properties window. In the segment field, enter the slices of the buses and/or wires separated by a comma with **no spaces**.

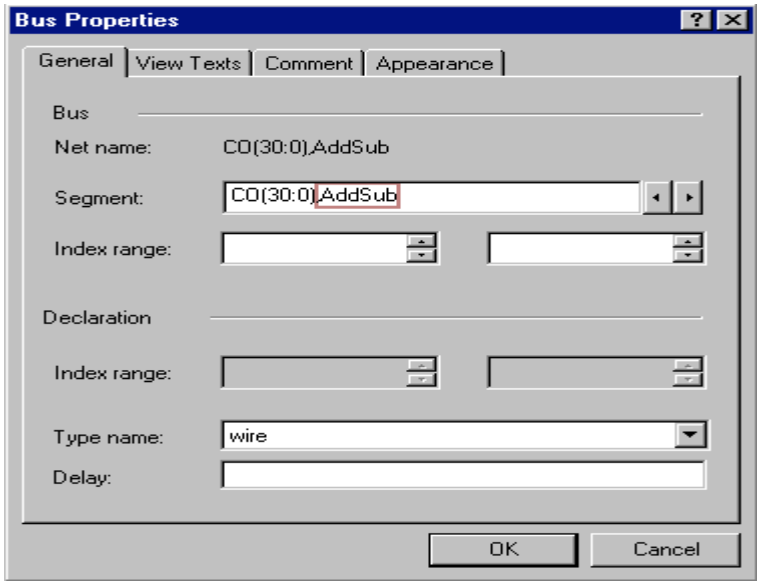

Figure 13

9. Use wires to connect the xor gate to the proper carryout bits, and to connect the AddSub signal to the multiplexer.
10. Save the schematic, run the check diagram tool, and compile the schematic.
11. Using the steps outlined in this tutorial, place a block symbol for the 32-bit adder/subtractor component into a new schematic. For this part of the tutorial, we will provide you with a "real" test fixture, addsub32_tf.v. This test fixture will test 100 random cases and print the results. The printed results simply show whether the component passed or not. If not, it will print the number of errors.
12. Add the test fixture to the project, place the block symbol for this test fixture into the schematic, connect the two components together, and run a simulation as you did for the full adder and 4-bit adder/subtractor components. Why is this not a good test fixture?

Concluding Remarks

You should now understand bottom-up designing and hierarchy. If you took the time to do the optional portion of this tutorial (we highly recommend you do), then you also know how to make easy to read schematic designs even when the design contains many components. Continue to experiment with Active-HDL, and begin to read and try to understand the Verilog code we provided you with for this tutorial.