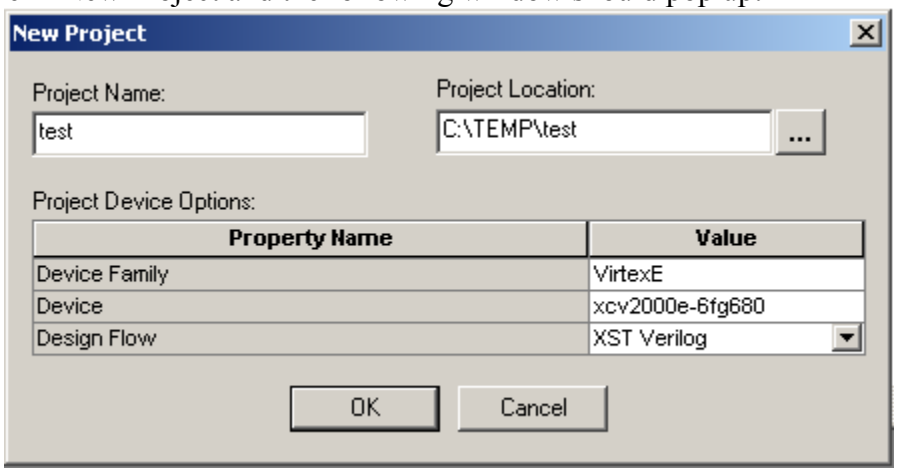
Spring 2003 CS152 BASIC PROJECT TUTORIAL

Topic Index

A) CREATING A PROJECT

1) Load up Xilinx Project Navigator.

2) Go to File→ New Project and the following window should pop up:



3) Set the Project settings as seen above:

Project Name	What you wish to call the project
Project Location	Place in c:\Temp and later move the project directory into your U:\ for storage
Device Family	VirtexE
Device	xcv2000e-6fg680
Design Flow	XST Verilog

B) ADDING NEW MODULES

There are 3 types of modules you will add: Verilog, schematic and testbench modules.

Adding Verilog Modules:

1) Go to Project→New Source and then Select Verilog Module. Under *File Name*, give the Verilog module you want to create an appropriate name. *** Note:** names should start with an alphabetic letter (meaning do NOT begin the name with a digit or special character). For this tutorial, call the module "bit1adderv". Click Next.

2) The next window that pops up allows you to specify the inputs and outputs of your Verilog module. You don't have to specify them all at the beginning; you can add more inputs and outputs directly in the Verilog file. Add inputs a, b, cin and outputs sum and cout. Click next. And then click finish.

3) The Verilog file should pop up and edit it from there. You can find details on how to program in Verilog elsewhere. Don't forget to save when you're done.
Your code for bit1adderv.v should look somewhat like this:

```
module bit1adderv(a,b,cin,sum,cout);
    input a;
    input b;
    input cin;
    output sum;
    output cout;
    wire sum, cout;

   assign {cout, sum} = a + b + cin;

endmodule
```

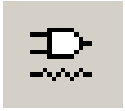
Adding Schematic Modules:

Now you will learn how to make schematics, which basically means you will be creating logic designs down to each single gate and wire.

1) Go to Project→New Source and then Select Schematic Module. Under *File Name*, give the Schematic module you want to create an appropriate name. *** Note:** names should start with an alphabetic letter (meaning do NOT begin the name with a digit or special character). For this tutorial, call the module "bit1adder". Click Next. And then click finish.

2) ECS, Xilinx's schematic program should load. A blank sheet should appear.

To add basic **logic gates** click on the button that looks like the following:



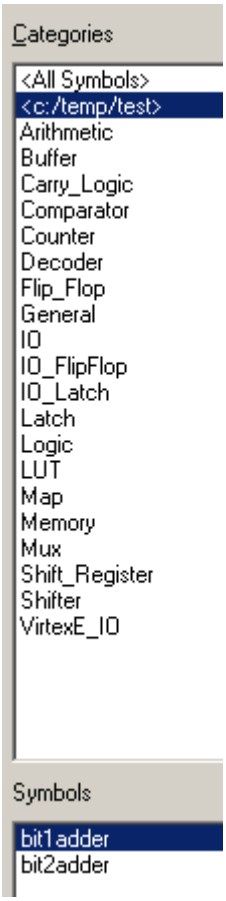
Or you can use the hot-key Ctrl-F

Two windows on the right hand side should appear including all the gates that you can add to the schematic. It should be fairly self explanatory. Press the ESC key when you are done adding symbols/gates.

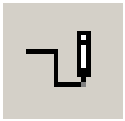
*Note:

schematic symbols you create within your project can be accessed by selecting the directory in which you placed your project.

For example in the right picture, the project was placed in c:/temp/test and two symbols created for that project from Verilog or schematic sources are named “bit1adder” and “bit2adder”.



To add **wires** click on the button that looks like the following:



Or you can use the hot-key Ctrl-W
Put your mouse head over the wire you want to begin the wire until you see a red outline and left click. Then drag your mouse and the wire to the net you want to connect it too until you see a red outline and left click. The wire then should be connected. Press the ESC key when you are done. Use can use the same method to add busses.

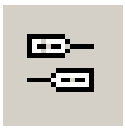
To **name nets/wires** click on the button that looks like the following:

Or you can use the hot-key Ctrl-D
Three new forms should appear on your toolbar as seen below:

Type the net name in the middle text box: For example, if you want to name the net "hello", type hello in the text box as shown below:

Finally you should see the net name at the end of your cursor when you place the cursor over the schematic window. Place the cursor over the wire/net you wish to name and left click. Press the ESC key when you are done.

To add **I/O pins** click on the button that looks like the following:

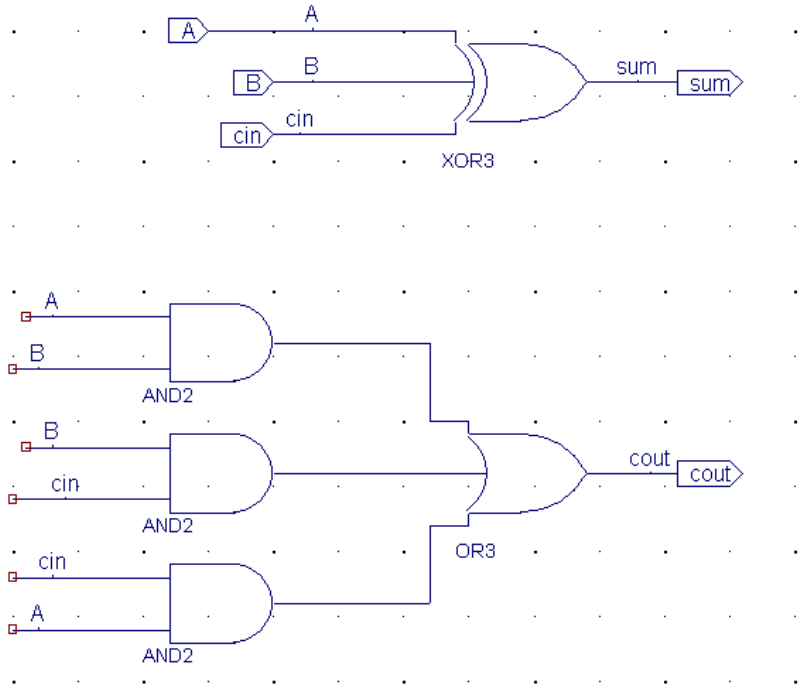


Or you can use the hot-key Ctrl-G
Select the type of pin you wish to add in the new box in the toolbar as seen below:

Finally place the cursor over the wire/net you wish to add the pin to and left click. Press the ESC key when you are done.

This should be enough for you to complete a basic design. If you need more help consult the Xilinx ECS manual.

Now after playing around with the tools, create a 1 bit adder that looks like the following:

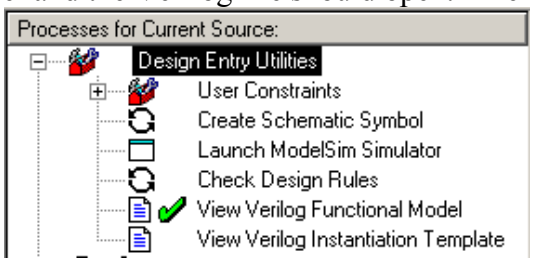


Don't forget to save when you're done.

The schematic file “bit1adder.sch” should then appear in the *Sources in Project* window in Xilinx Project Navigator.

To **view your schematic in Verilog** HDL

Select your schematic file in the *Sources in Project* window. Then in the *Processes for Current Source* window, expand the Design Entry Utilities box. Double click View Verilog Functional Model and the Verilog file should open. The file is read-only though.



Creating a schematic symbol from a Verilog or Schematic source in your project.

Select the source file in the *Sources in Project* window. Then in *Processes for Current Source* window, expand the Design Entry Utilities box. Double click *Create Schematic Symbol*. This will allow you to use the block you created in the source file in the ECS schematic editor program. This allows you to reuse a block you created in ECS without having to redraw it over and over again.

Combining Verilog and Schematics

You can easily combine schematic and Verilog modules in your project. For this tutorial, we will use both 1 bit adders you created in Verilog and in schematic by “wrapping” both adders in a Verilog file to create a 2 bit adder. Basically in Verilog you can reference each schematic module by just referencing the name of the schematic block.

1. Create a new Verilog module as detailed in the previous sections named “bit2adder”.
2. For the appropriate inputs and outputs, don’t forget some are 2 bit busses.
3. Your code should look somewhat like the following:

```
module bit2adder(a,b,cin,sum,cout);
    input [1:0] a;
    input [1:0] b;
    input cin;
    output [1:0] sum;
    output cout;
    wire ctemp;
   // note that the input output position of both adders are different
   // cout precedes sum out in bitladder though the opposite occurs for bitladderv
   bitladder ba (a[0], b[0], cin, ctemp, sum[0]);
   bitladderv bav (a[1], b[1], ctemp, sum[1], cout);

endmodule
```

4. Don't forget to save the file.

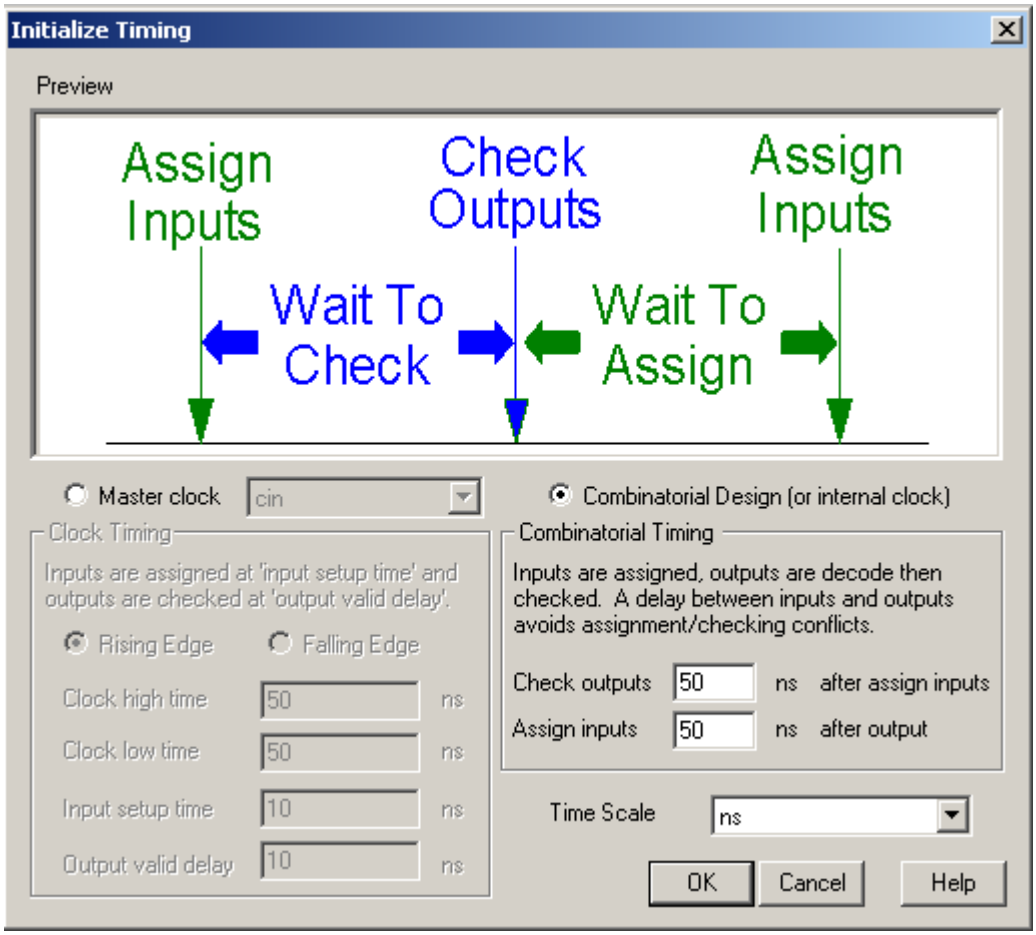
Adding Testbench Modules:

A testbench module is used to test the functionality or get the expected results of your project designs. You will have to create a separate testbench for each individual source you would like to test. For the current project let's create a testbench to make sure bit2adder works correctly. To accomplish this, do the following steps:

1. Select the bit1adder.v source in the Sources in Project window.
2. Select Project →New Source.
3. In the New dialog box, select Test Bench Waveform source type.
4. Type the name "bit2adder_tbw".
5. Click Next.
6. In the Associate with Source window, select bit2adder. This associates the testbench with the bit2adder so the testbench file knows which file you are going to be testing with. Click Next.
7. Click Finish.

HDL Bencher then is launched and ready for timing requirements to be entered.

> You can now specify the timing parameters used during simulation. The clock high time and clock low time together define the clock period for which the design must operate. The Input setup time defines when inputs must be valid. The Output valid delay defines the time after active clock edge when the outputs must be valid.
>
> For this tutorial, you will not change any of the default timing constraints but will use the defaults for Combinatorial Timing.



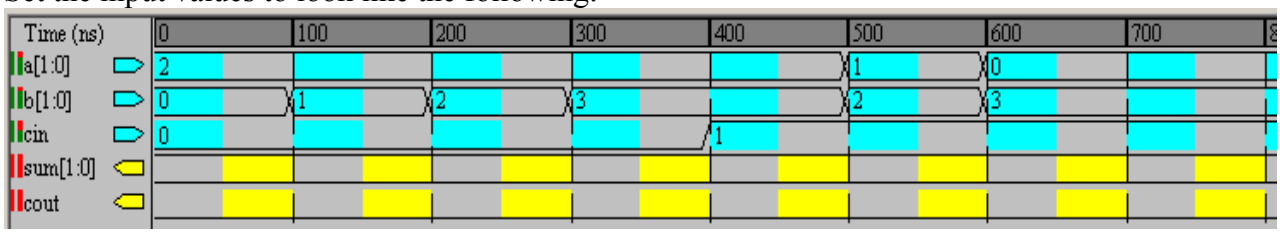
The settings should be set as shown above. Check *Combinatorial Design* and set the *Check outputs* and *Assign inputs* to 50 ns.

Click OK to accept the default timing constraints.

Initializing Inputs

For each signal cell at a specific clock cycle, click on the cell to specify the value you want it to have starting at that clock cycle. You will have to type in the value you would like it to have if the signal is more than 1 bit, else the signal is 1 bit and you can just click it to toggle its value. The signal will hold the value you entered for the rest of the simulation after the indicated clock cycle unless you set its value again in a later cycle. ***NOTE** If you set the output values to the expected correct results, when you generate the expected outputs later on, HDL bencher will check if the actual results by running the inputs through your design will match your inputted outputs. This may be useful for your testing purposes.

Set the input values to look like the following:

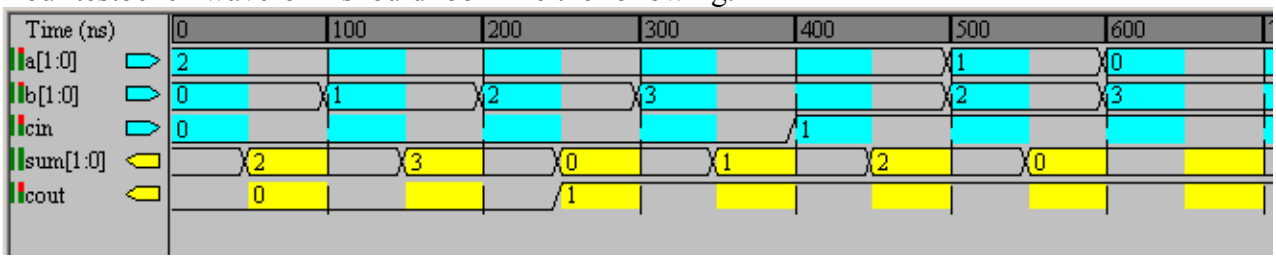


Finally, save your testbench waveform by selecting
File→Save Waveform or by clicking the Save Waveform icon in the toolbar.

Next, HDL Bencher will prompt you to set the number of clock cycles for which you wish to simulate.
Enter the number of cycles you want the testbench to simulate for in the dialog box: 'End the testbench __ cycles after the last input assignment'. The default value is 1. For this tutorial, let's set it to 8. Click OK and proceed to exit the HDL Bencher. The new testbench waveform source (bit2adder_tbw) is automatically added to the project.

Generating the Expected Simulation Output Values
Now you can generate the expected outputs for the bit1adder module based on the initialized inputs you have entered.
1. Select bit2adder_tbw.tbw in the *Sources in Project* window.
2. In the *Processes for Current Source* window, click the + beside ModelSim Simulator to expand the hierarchy.
3. Double-click *Generate Expected Simulation Results*.

This process runs a background simulation using the inputs specified, generating output values which are added to the testbench waveform.
Your testbench waveform should look like the following.



Exit HDL Bencher without saving your waveform.

C) SIMULATING YOUR DESIGN

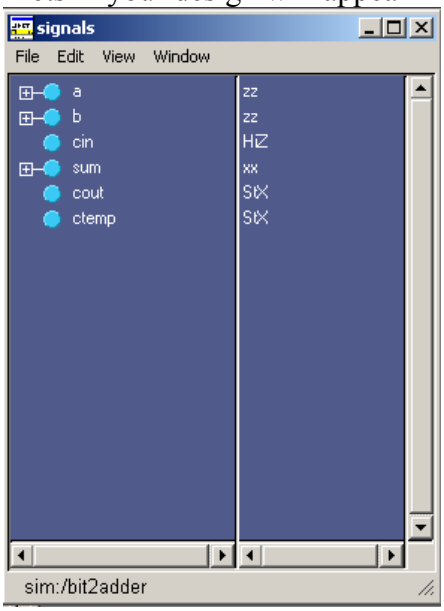
In this course, you can also use ModelSim to simulate your designs. It allows you to assign random values to inputs of your design and it can generate waveforms for you to view your design's outputs.

Launching ModelSim to load your design

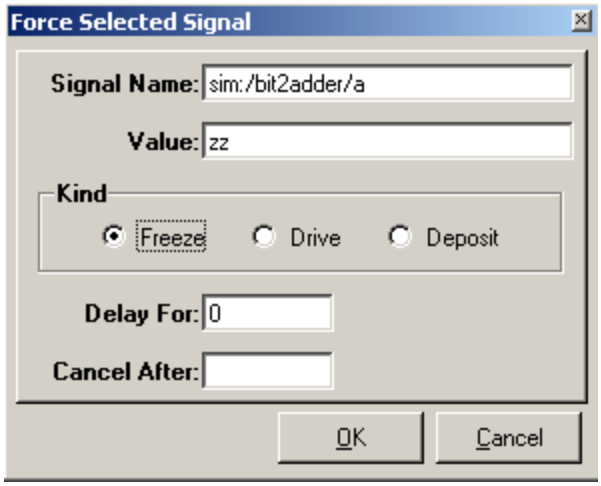
Manually testing your design

Select the source module in the Sources in Project window that you wish to simulate. In our case, select bit2adder.v. Then in the Processes for Current Source window, expand the Design Entry Utilities box. Double click Launch ModelSim Simulator. This should load up ModelSim.

Several windows should pop up including the main ModelSim command window along with wave, structure and signals windows.
You'll notice that all logical nets in your design will appear in the signals window.



Originally all the nets that are not already set to a specific value are by default in a high impedance state, denoted by Hiz or z's. You can set the input values (and even the output values, though most likely you would not want to do that) to specific values by first selecting the signal you want to set in the Signals window, and then selecting in the toolbar, signals→Edit→Force. The following window should pop up:



Enter the value you want the net to be set to in the Value field, and you can also specify how long you want the signal to be held at that value in the Delay For field or specify the exact cycle when the signal should stop being forced in the Cancel After field. Click OK.

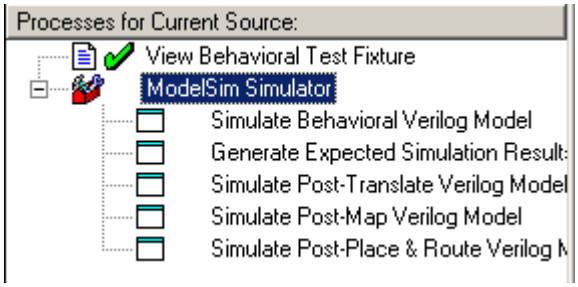
Set all input signals to any test values and then you can begin the simulation.
To simulate, go into the main ModelSim command prompt and type "run" and press the Enter key. This will run the simulation for one cycle. You can have it run for more than one cycle by typing "run" and then followed by the number of cycles you want it to simulate for. The resulting simulated outputs will appear as waveforms in the Wave window. Using this method of testing, you will have to manually test if the output values are correct though. This may certainly be a tedious task, though it does allow you to view the timing delays as well as change input values on the fly.

To exit ModelSim, just close the main command window.

Simulating your design using a testbench module.

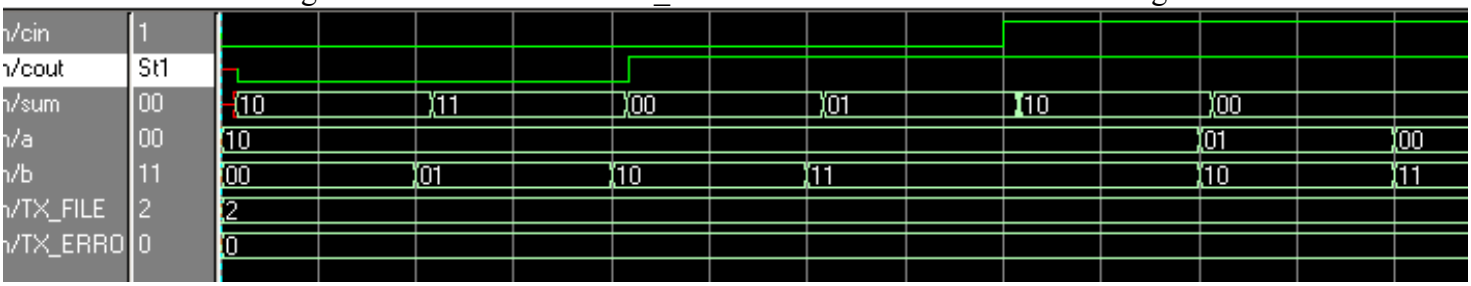
Simulating your testbench module through ModelSim after synthesizing/mapping/implementing your design will allow you to view the actual timing delays and characteristics of your design. Previously, you could only check the functional behavior of your designs (meaning it spits out the right outputs for the right inputs on a high level). To do this:

First select your previously created testbench source in the *Sources in Project* window.



In the *Processes for Current Source* window, expand the ModelSim Simulator box, and double click *Simulate Post-Place & Route Verilog Model.*
This will launch ModelSim which will run your testbench (simulate the input values you set in the testbench). The resulting waveforms will result in the wave window. From here you should be able to identify timing the results of your testbench and also timing delays for your tested module.
The resulting waveforms in the wave window may not be easily seen b/c the time interval between clock cycles may be too long or too short. To remedy this you can either use the zoom out tool (the magnifying glass with the plus sign) to shorten the length between clock cycles and view more of the wave at once, or the zoom in tool to do the opposite.

The resulting waveform from bit2adder_tbw.tbw should look like the following:



From the output waveforms, you can calculate the timing delays between when an input value changes and the output value responds. This though somewhat tedious can help you with timing analysis of your design.

Writing Testbench fixtures

Creating testbenches in Xilinx Testbencher does limit the types of tests as compared to writing your own Verilog test fixtures. To create test fixtures that can utilize Verilog functions such as $random and $display and Verilog's recursive functions, you can create Verilog test fixtures for the sources in your project by doing the following:

Adding Verilog Modules:

1) Go to Project→New Source and then Select Verilog Test Fixture. Under *File Name*, give the Verilog Tesst Fixture you want to create an appropriate name. *** Note:** names should start with an alphabetic letter (meaning do NOT begin the name with a digit or special character). For this tutorial, call the module "bit2adder_tf". Click Next.

2) The next window that pops up asks you to specify the source file to associate the test fixture with. Select the module "bit2adder". Click next. Click finish.

3) The Verilog file should pop up and edit it from there. Code that declares the module you want to test and its initial inputs should already be set for you. All that is left for you to write is what and when to set values to.
Under the autoinitialize inputs definition is where you should start your code. Most likely you will want to use an always block that looks like the following:

```
module testbench( );
....
//initialize inputs
….
   ‘endif
//in this section you start your code.
//initialize inputs and other temp variables
reg [7:0] someTempReg;
//initial values to inputs
initial
        a = 0
        b = 0
always
#100 //time to wait between input changes
begin
        a = $random;
        b = 2;
        cin = $random;
        #100 //wait for outputs to settle
        $display(“just like c printf %d, %d, %d”, a, b, cin);
end
endmodule
```

Running the testfixture

Select the testfixture source in the *Sources in Project* window that you wish to simulate. In our case, select bit2adder_tf.tf. Then in the *Processes for Current Source* window, expand the *ModelSim Simulator* box. Double click *Simulate Post-Place and Route Verilog Module*. This should load up ModelSim and the testfixture results. ModelSim by default runs the testfixture for 1 ns.

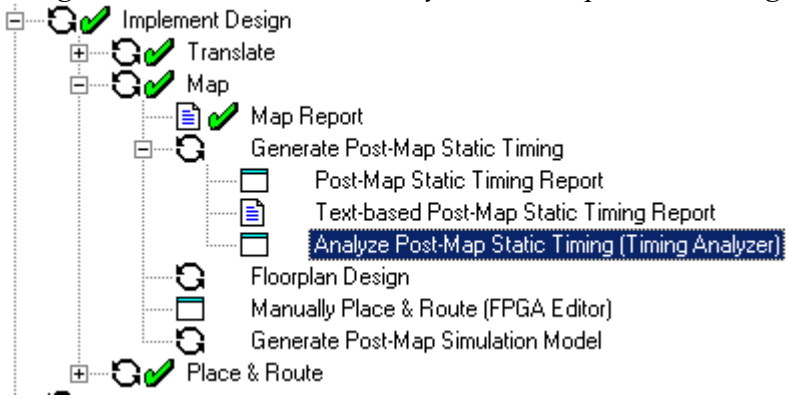
Any calls to **$display** will print into ModelSim’s command prompt window.

Timing Analysis (Using Timing Analyzer)

Often you will want to analyze the timing delays and have more detailed timing descriptions of your design. In this case you can run the Xilinx Timing Analyzer program to analyze a source of your project.

First select the source in the *Sources in Project* window that you wish to analyze. In our case, select bit2adder.v. Then in the *Processes for Current Source* window, expand the

Implement Design box, expand the *Map* box, expand *the Generate Post-Map Static Timing* and then double-click *Analyze Post-Map Static Timing*(Timing Analyzer).



This should open up Xilinx Timing Analyzer.
In the toolbar, click on the *Analyze against Autogenerated Design Constraints* button:

This will open up a detailed timing report including path delays and maximum wire delays for your design.