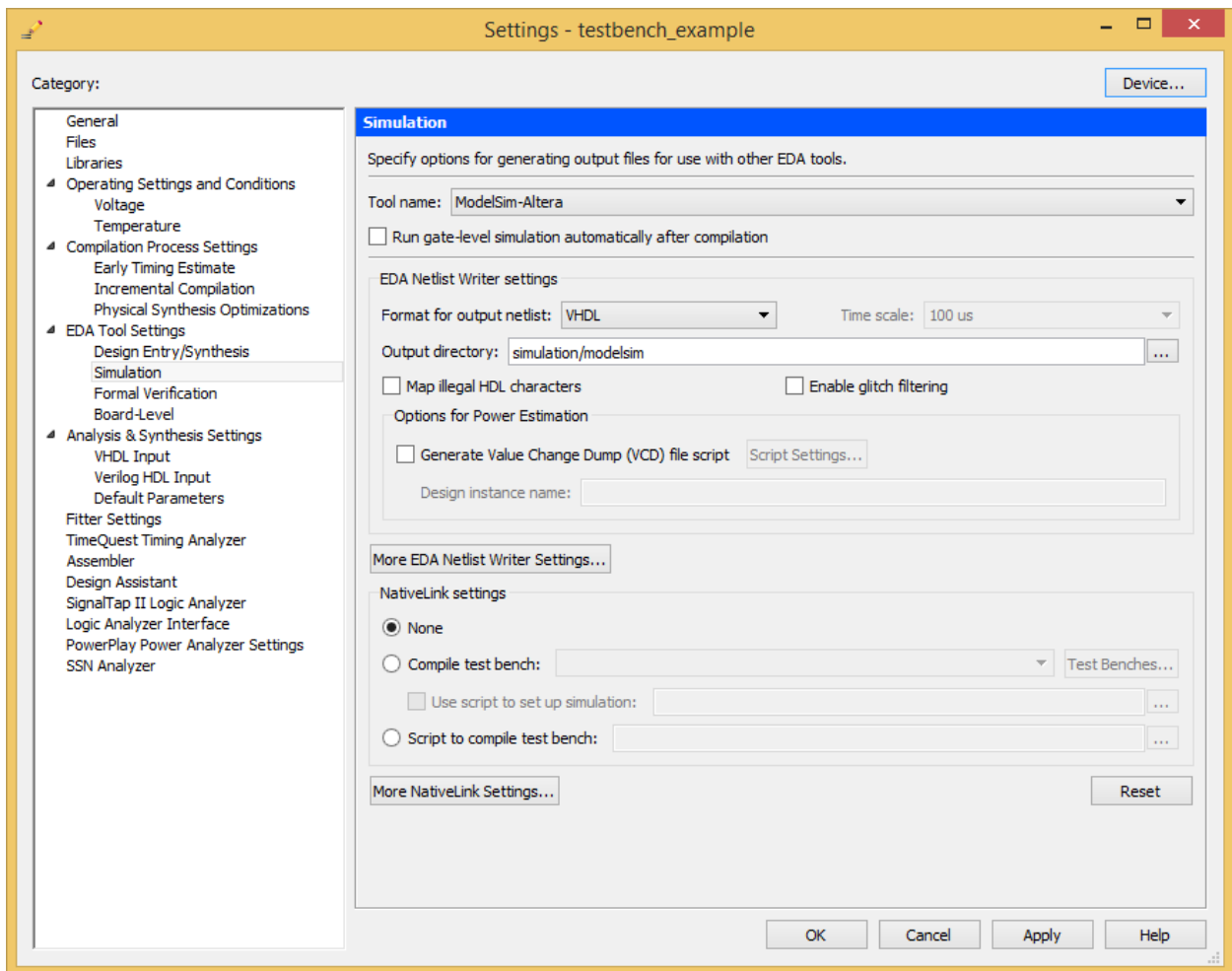


## Quartus II Testbench Tutorial

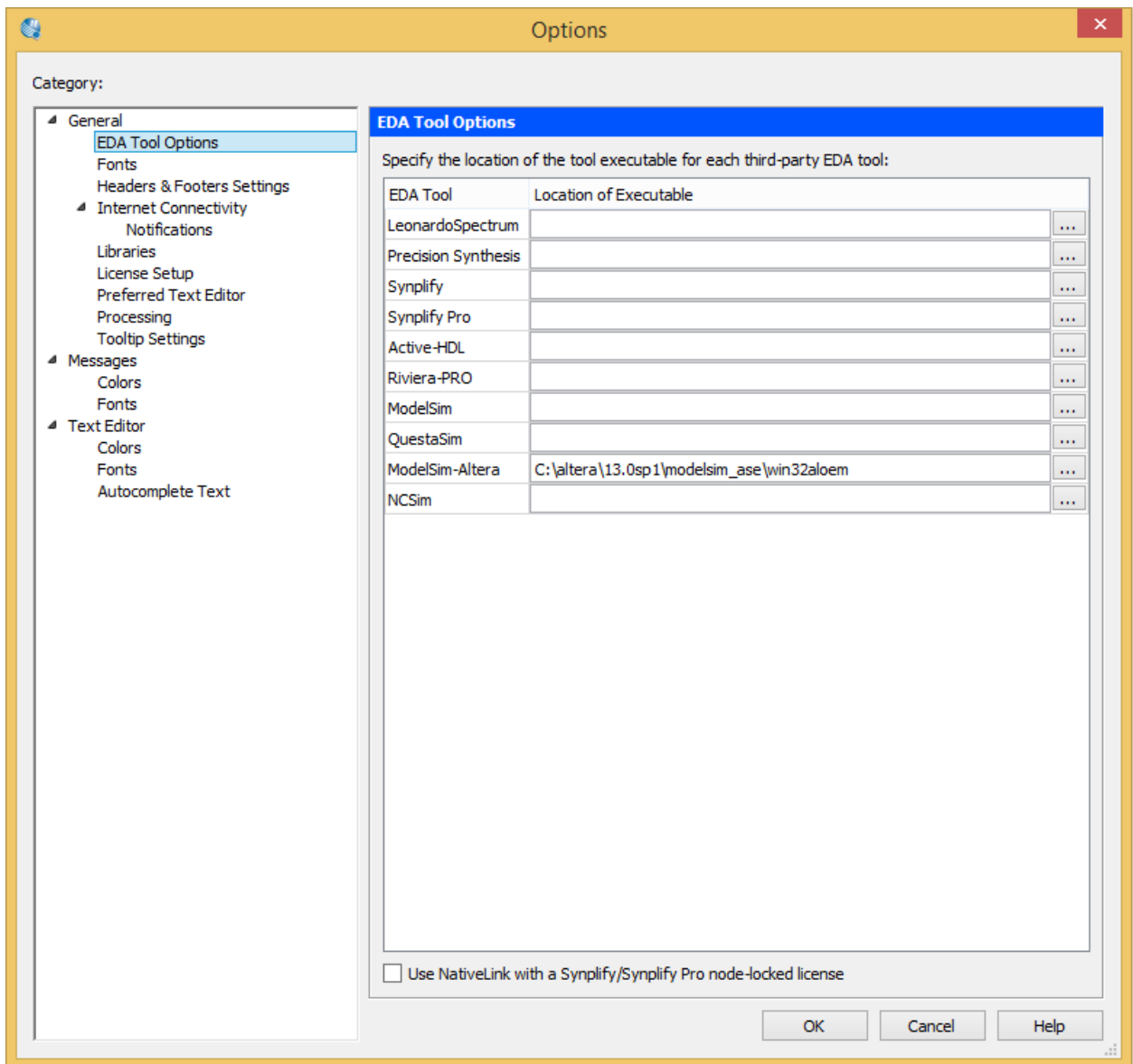
This tutorial will walk you through the steps of creating Verilog modules in Quartus II and simulating them using Altera-Modelsim.

### 1) Create a new Quartus Project & configure it for Altera-Modelsim

To configure Quartus to use Altera-Modelsim as the simulator, first create a new project (or open an existing one) and go to Assignments > Settings > EDA Tool Settings > Simulation. Make sure “Modelsim-Altera” is selected as the tool as shown below (make sure to click *Apply* then *OK* if you made any changes).

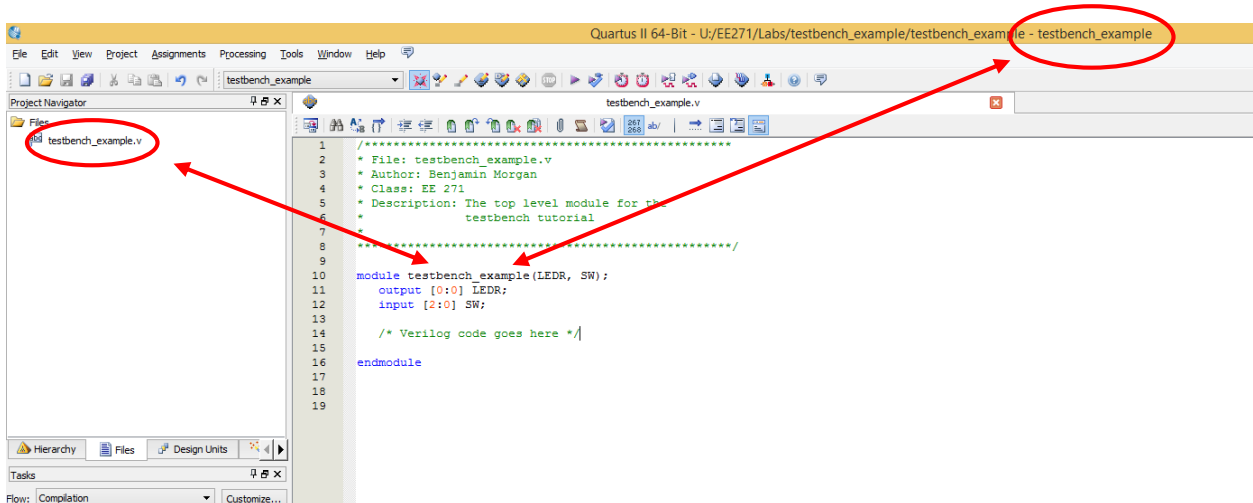


Next go to Tools > Options > General > EDA Tools sure that the path next to Modelsim-Altera is configured correctly. If you installed Quartus II with Altera-Modelsim the path should be similar to the one shown in the figure below (i.e. on Windows: “C:\altera\13.0sp1\mode-sim\_ase\win32aloem”), otherwise you will need to browse to where you installed Altera Modelsim and point it to the *win32aloem* directory.



## 2) Create your Top-Level Module

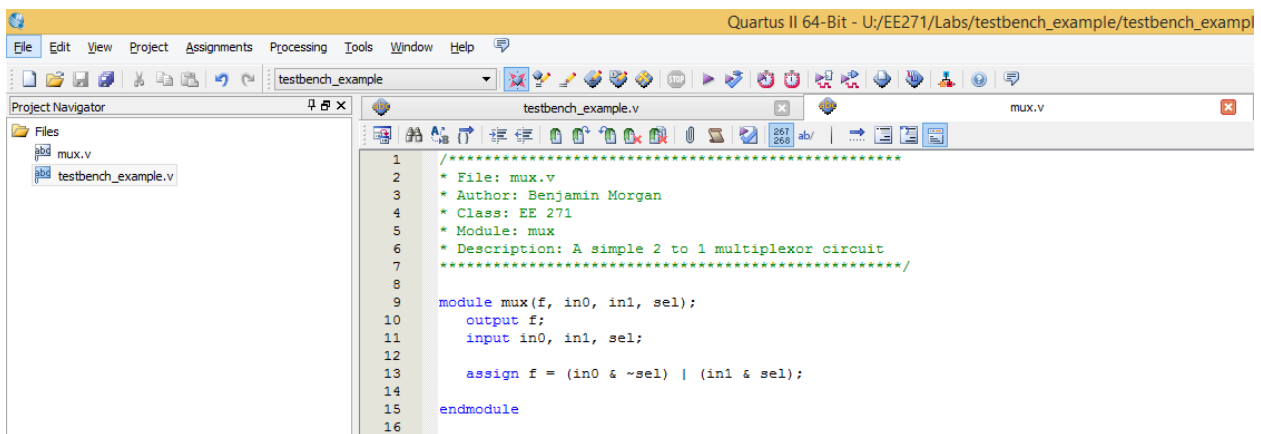
In Quartus, the top level module must have the same name as your project name and must be in a file with the same name and a “.v” extension. To create the top level module go back to the main Quartus window and select File > New... > Design Files > Verilog HDL File and click OK. An example template of a top-level module can be seen below. *Notice how the project is called “testbench\_example” which is the same name as the top level module, “testbench\_example” which exists in the file “testbench\_example.v”.*



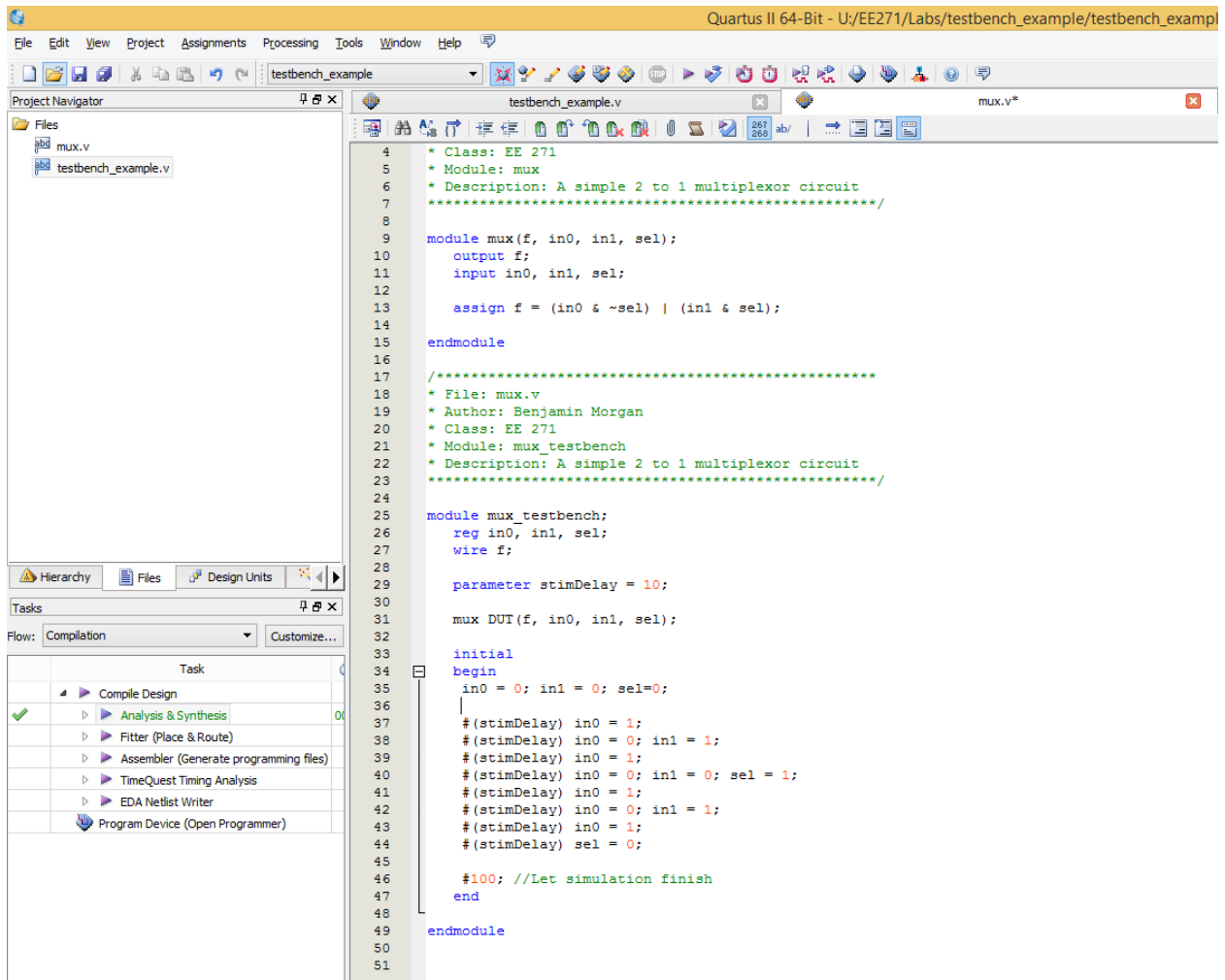
### 3) Create your Unit Under Test & Testbench

Next we will write the Verilog code that we want to test. This code can go in the same file as the top-level, but it is good practice for separate modules to have their own files, so we will do that in this example.

For this tutorial the code that we want to test will be a simple 2 to 1 multiplexor circuit. Create a new file by going File > New... etc. and save it as "mux.v". Next write the code that we want to test as shown below (Note: the example code can all be found in the appendix)



Next we will write the testbench for our multiplexor circuit. While it is not necessary, it is good practice to keep the testbench for a module in the same file as the module itself, so again that is what we will do here. Below is a simple testbench for testing this circuit (again, the code can be found in the appendix).

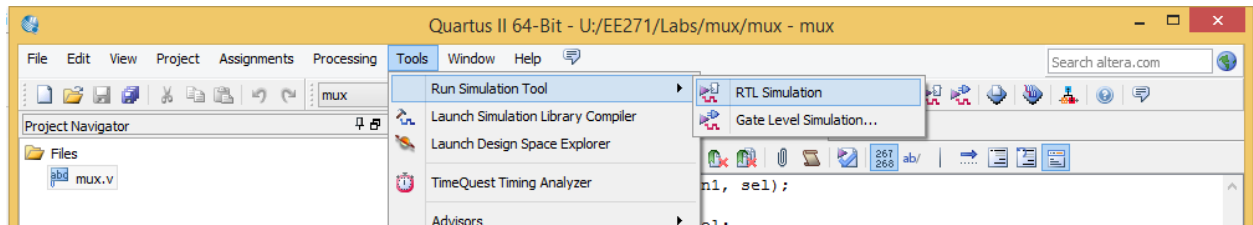


Note: The testbench module does not contain any inputs or outputs, and it instantiates an instance of our mux module that we want to test calling it “DUT”.

Once you have finished writing everything, including your testbench, and everything compiles correctly you can move on to part 4. (Note: you might need to put some dummy inputs and outputs in your top-level module to get Quartus to stop complaining, but don't worry these can be changed later).

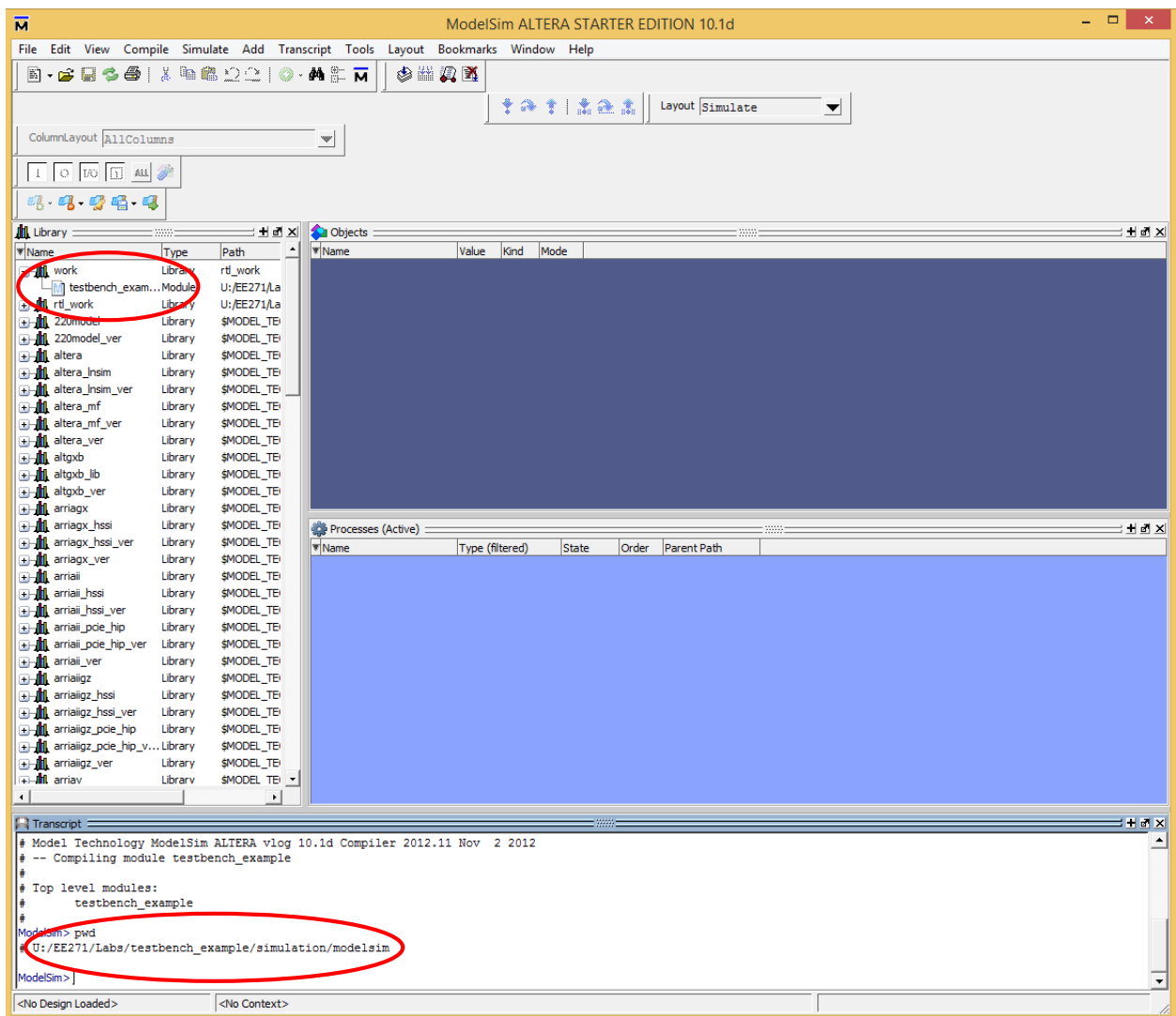
#### 4) Open Altera-Modelsim

To open altera-modelsim go to Tools > Simulation Tool > RTL Simulation. This should open up Altera-Modelsim.

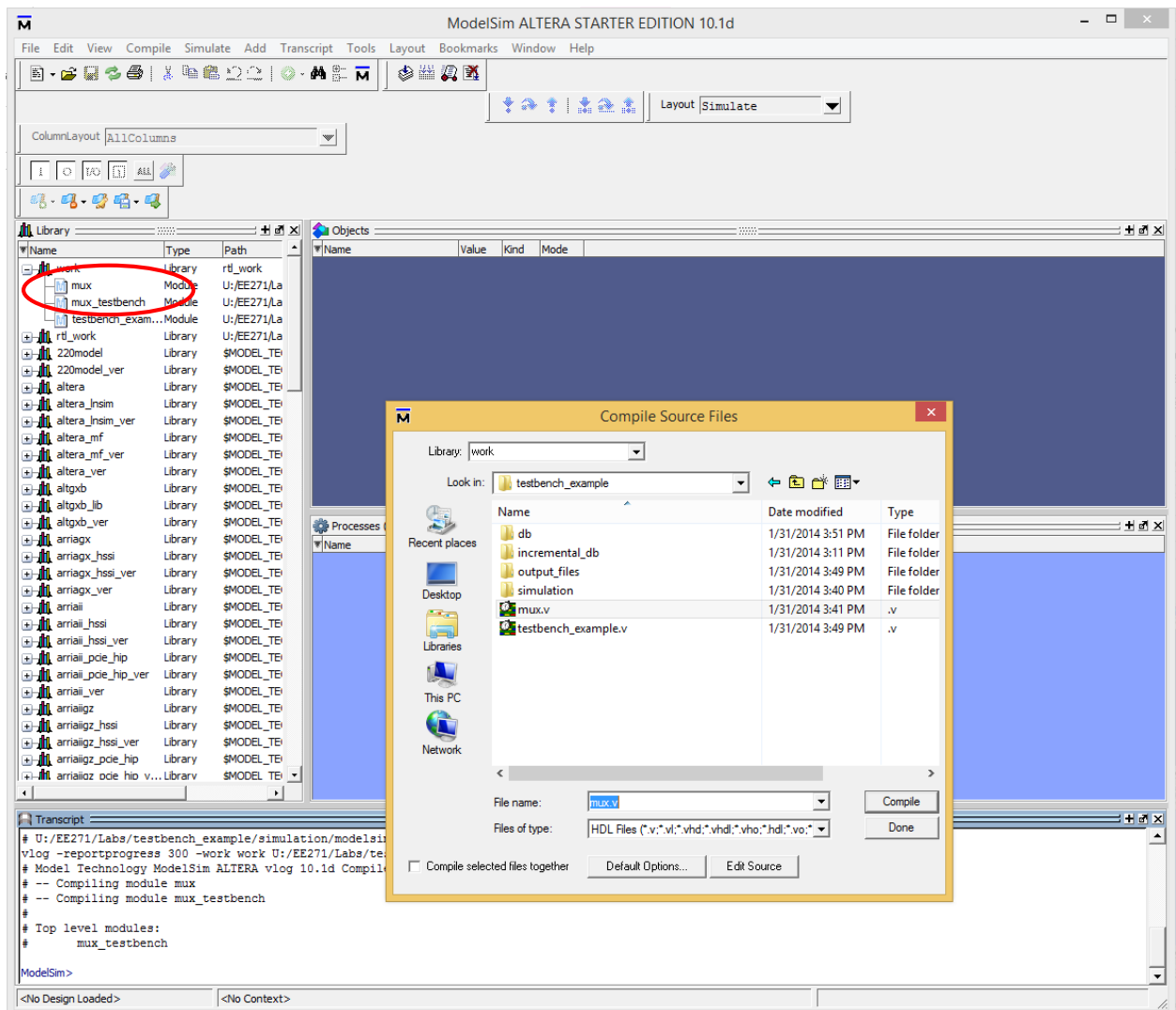


Modelsim creates a *WORK* library and puts your compiled Verilog modules from Quartus in it. It also sets the working directory to be the modelsim directory of your project (this can be changed in the settings in step one, but is not recommended).

## 5) Compile the UUT & Testbench



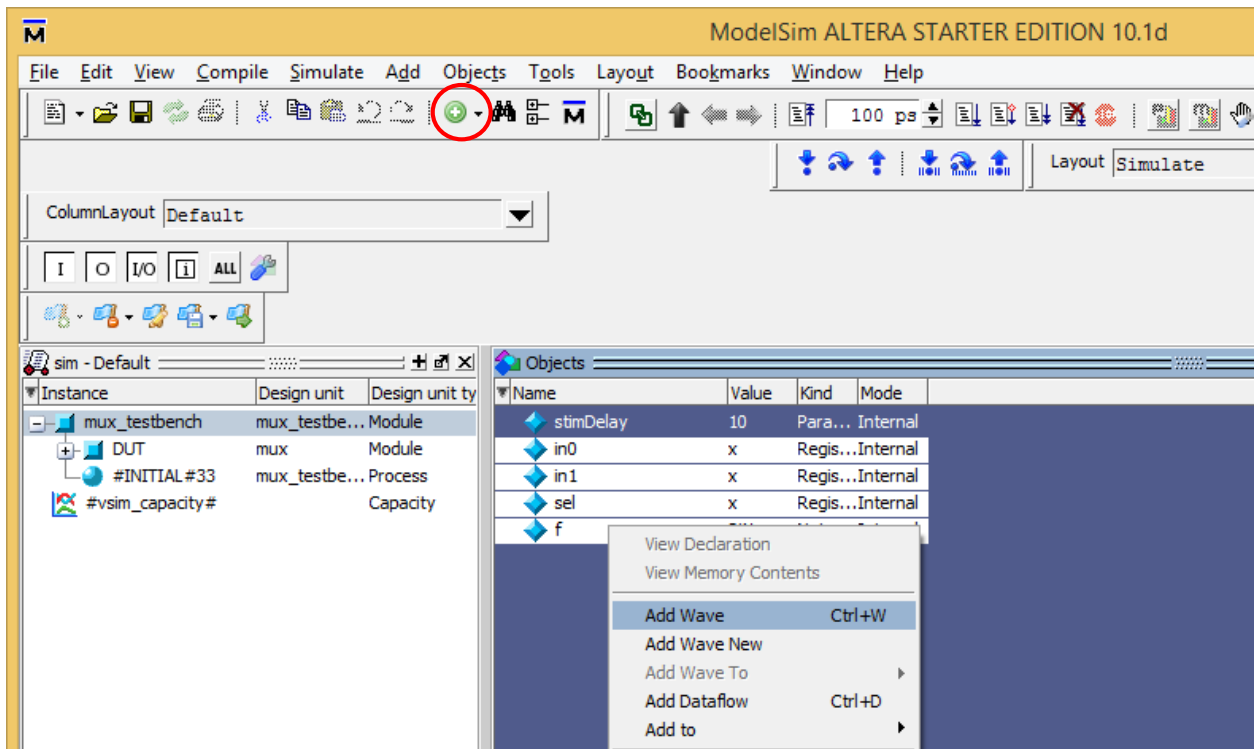
To compile your Testbench go to **Compile > Compile..** and select the file that contains your Verilog code that you want to test and its testbench. If you left the default settings for modelsim's working directory you will probably have to browse up a few folders to find the file you want (in this case mux.v). Once you have selected the file click *Compile*, then after it has finished compiling click *Done*.



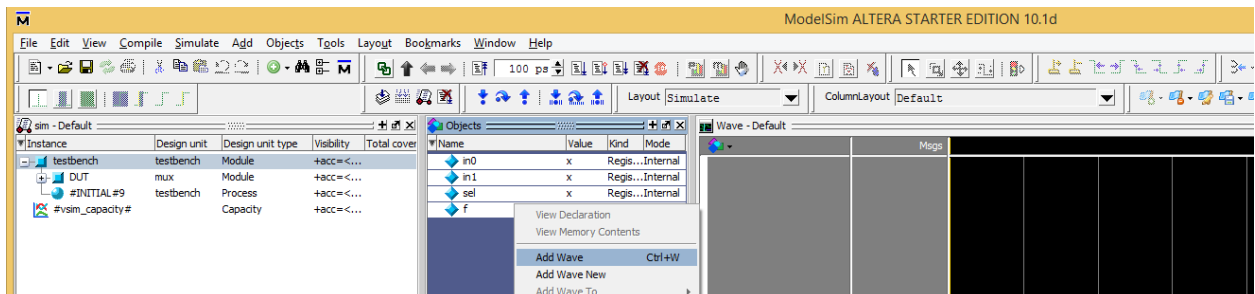
Notice two new modules have been added to your *WORK* library. This process can also be done by declaring an instance of your UUT module in your top-level module in Quartus. However, if you have a module that you do not plan to use directly in your top-level module you can still use the above instructions to compile it in Modelsim. (Note: You can in fact open Modelsim on its own, browse to the where the file that contains the code you want to test and its testbench lives, and compile it there. In some cases this may make a bit more sense than going through Quartus.)

## 6) Simulate

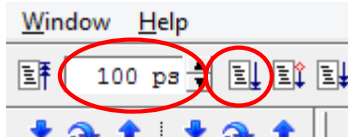
To run the simulation right click on the testbench module that you want to simulate, in this case “mux\_testbench”, i.e. right click mux\_testbench > simulate, or simply double click on it. (Note: you can have multiple testbenches compiled at the same time under your WORK library and you can then select which one you want to simulate).



Next, add the waves by selecting the signals you want and right click > add wave (or simply use the hot key CTRL + W). This can also be done by selecting the signals you want and clicking the icon of a green circle with a + in it at the top.

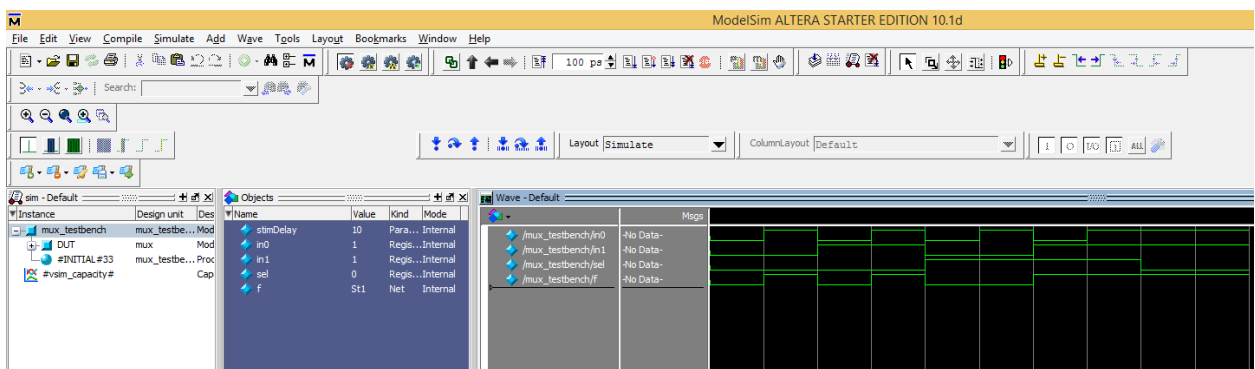


To run the simulation click the icon that looks like a white page with a down-arrow directly to the right of the box that says 100 ps.



To adjust the simulation time simply edit the time in this box (for this multiplexor example 100 ps is more than enough time, but for more complex circuits you may need to simulate longer). This can also be achieved by typing: *run 100 ps* in the command prompt at the bottom.

Observe your simulation waveforms (you may have to adjust the window zoom/scroll location to see your waveforms)



\*Note if you make a change to either your Unit Under Test or your testbench, you must first recompile whatever you changed before re-running the simulation. Also make sure that you click the *Restart* button directly to the right of the simulation time window (or simply type “restart” in the command prompt at the bottom) otherwise you will not see your changes.



## Appendix: Source Code

```

/*****
* File: testbench_example.v
* Author: Benjamin Morgan
* Class: EE 271
* Module: testbench_example
* Description: The top level module for the
*               testbench tutorial
*
*****/

module testbench_example(LED_R, SW);
    output [0:0] LED_R;
    input [2:0] SW;

    //mux my_mux(.f(LED_R[0]), .in0(SW[0]), .in1(SW[1]), .sel(SW[2]));

endmodule

/*****
* File: mux.v
* Author: Benjamin Morgan
* Class: EE 271
* Module: mux
* Description: A simple 2 to 1 multiplexor circuit
*****/

module mux(f, in0, in1, sel);
    output f;
    input in0, in1, sel;

    assign f = (in0 & ~sel) | (in1 & sel);

endmodule
```

```
/******  
* File: mux.v  
* Author: Benjamin Morgan  
* Class: EE 271  
* Module: mux_testbench  
* Description: A simple 2 to 1 multiplexor circuit  
*****/
```

```
module mux_testbench;  
  reg in0, in1, sel;  
  wire f;  
  
  parameter stimDelay = 10;  
  
  mux DUT(f, in0, in1, sel);  
  
  initial  
  begin  
    in0 = 0; in1 = 0; sel=0;  
  
    #(stimDelay) in0 = 1;  
    #(stimDelay) in0 = 0; in1 = 1;  
    #(stimDelay) in0 = 1;  
    #(stimDelay) in0 = 0; in1 = 0; sel = 1;  
    #(stimDelay) in0 = 1;  
    #(stimDelay) in0 = 0; in1 = 1;  
    #(stimDelay) in0 = 1;  
    #(stimDelay) sel = 0;  
  
    #100; //Let simulation finish  
  end  
  
endmodule
```