

ECE/CS 5710/6710 - Modelsim Tutorial

1 Objective

This document shows the main steps for performing the simulation of VHDL or Verilog netlists. This step is crucial to ensure that your RTL code will behave correctly, before and after the synthesis and Place&Route steps. The tool used is Modelsim® from Mentor Graphics. Keep in mind that this document simply gives a general overview and presents a basic case (a Verilog 2-input xor module) so you can get an idea of how to use the tool for your project. There are many other possibilities when using the tools that you will probably learn by yourself when working with it.

2 Simulating the RTL Code

3 Starting the Tool

To launch Modelsim, go to your *modelsim* directory and do:

```
vsim
```

4 Creating a New Project

Modelsim® allows you to work with project to organize your files. It is generally a good practice to have a project dedicated for each design. For instance, in this lab, you will create a specific project but you should create a new one for your class project later on. To create a new project:

1. Go to *File -> New -> Project*.
2. Define the following settings:
 - *Project Name*: xor2_rtl
 - *Project Location*:/PROJECTS
 - *Default Library Name*: xor2_rtl (it is a good idea to use the same name as your projectname).
 - Select *Copy Library Mappings*
3. Click OK.

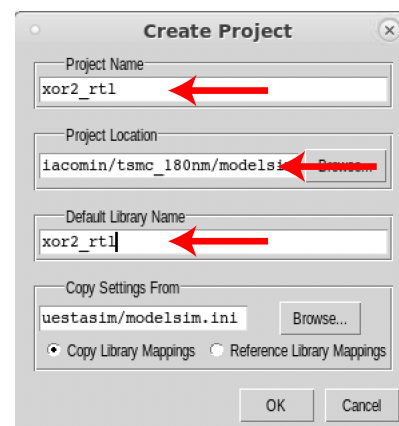


Figure 1: Creating a new project.

5 Adding the RTL Files

A window should pop up. From there, you can either create a new file for your RTL code or import existing ones. For this tutorial, you will import two files: a verilog file describing the RTL code of a 2-input xor function and a verilog file for its testbench. To do so: (note the appropriate reports).

1. Click on *Add Existing File*.
2. A new window opens. Click on *browse* and go to your *HDL/RTL* folder. Select both *xor2.v* and *xor2_tb.v* files.
3. Click on *Open* and then *Ok*.
4. You can then close the *Add items to the Project* Window.
5. You should be able to see the 2 verilog files in the main window.

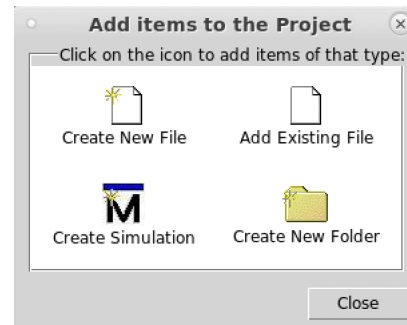


Figure 2: Creating a new project.

6 Compiling the Files

Now, you need to compile the files. This step will check for errors in your verilog code and let you know if there are any.

- R** When working on your class project, it is suggested to change the project settings to allow compilation messages to appear in the **Transcript pane**. To do so, go to: *Project -> Project Settings...*, check the *Display compiler output* box and click *Ok*.

In the *Project* tab, right click and select: *Compile -> Compile All*.

If there is no compilation error, each file status becomes OK (✓). If there are some errors, the status is ✗. Compilation error messages are displayed in red in the console pane. If you double click on those, it will open the editor pane and highlights the line causing the error.

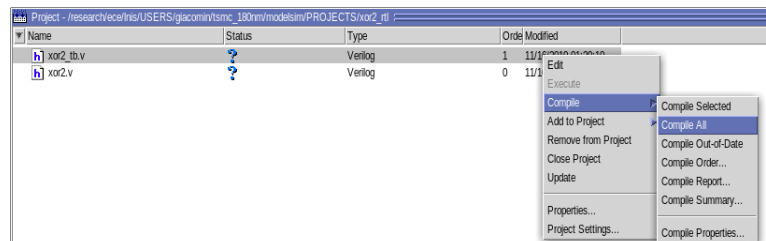


Figure 3: Compiling the files.

7 Defining the Simulation Configuration

This step will define all the necessary settings required to perform the RTL simulation. To do so:

1. Go to: *Project -> Add to Project -> Simulation Configuration...*
2. Define the following settings:¹

- *Simulation Configuration Name:* rtl
- *Resolution:* ns
- *In the Design Tab:* Select the configuration declaration *xor2_tb* (this is your testbench module) in the *xor2_rtl* (that you defined when creating your project) design library.
- *Optimization Options...* : Click here and select *Apply full visibility to all modules* so all the signals will be available to be displayed when doing the simulation.
- Click on *Save*.

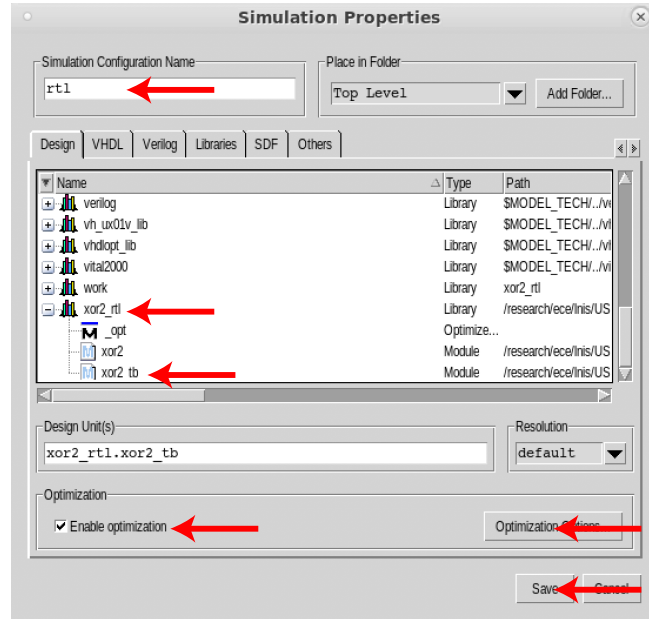




Figure 4: Creating a simulation configuration.

3. You should be able to see the simulation icon () in the *Project* tab.

8 Running the Simulation

To start the simulation, double click on the simulation configuration icon () in the *Project* tab. A new *sim* tab should appear in the Modelsim® window, as shown in Fig. 5. This tab includes all the simulation hierarchy (such as the different modules, the statements, *etc.*).

Now, select the signals to be displayed. In the *Objects* pane, select the signals you want to display, right click and do: *Add Wave*. You can also display all the top level signals of your testbench by going to the *sim* pane, right clicking on the top level module (*xor2_tb* in this case) and do: *Add Wave*. You should now be able to see the signal list in the *wave* pane.

To launch the simulation:

1. Specify a simulation time (which should makes sense for the defined testbench).
2. Click on *Run*.
3. In case you modify your verilog file, you can select *Restart* to clean the simulation output pane and restart the simulation.

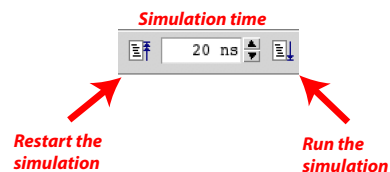


Figure 6: Simulation toolbox.

You can also run the simulation by using the **Transcript pane**.

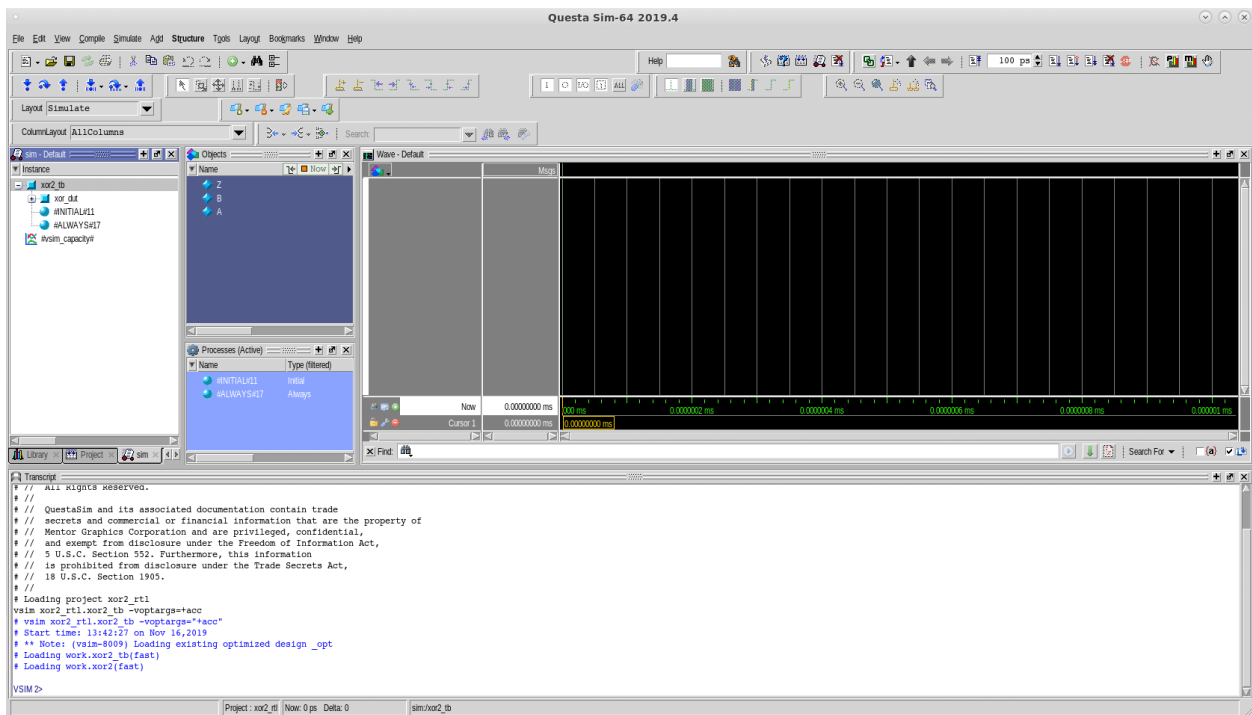


Figure 5: Modelsim® window after doing the simulation.

run 10ns

To restart the simulation, you can also use:

restart -f

Now, you should be able to see the displayed results. In some cases, you might need to change the radix of a variable (to decimal, hexadecimal, signed, *etc.*). To do so: In the *Wave* window, click on the signal or the variable and select *Radix -> radix name*.

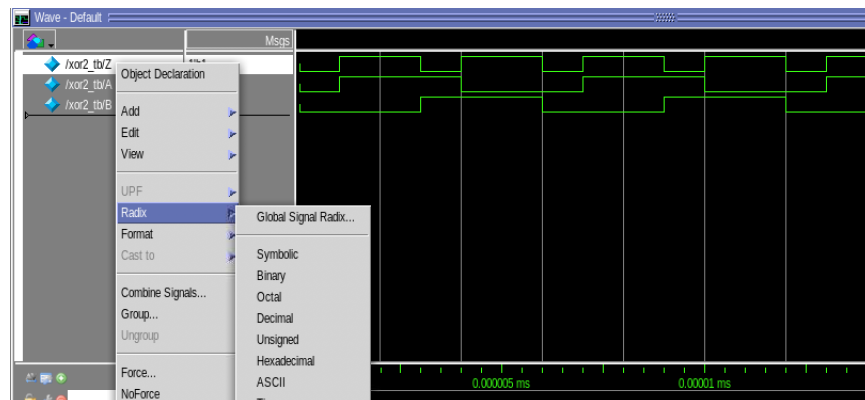





Figure 7: Change the radix of a variable.

9 Modifying the RTL files

For this tutorial, the two provided files show the behavior of a XOR so you do not have to modify those. However, for your project, you probably will have to write some verilog code yourself and modify it many times, after realizing that the simulation does not behave as intended. To flow is generally the following:

1. You can modify any verilog files which are in your current project by double clicking on it. A new window will open where you can make your changes. Once you are done, save the file.
2. You then need to re-compile the changed files, as explained previously.
3. Then you need to launch the simulation. If a simulation was already opened, you do not need to double click on the the simulation icon () in the *Project* tab to relaunch it. You simply need to restart () your simulation, and re-run () it again.
4. Verify if the simulation behaves as expected and reiterate the previous steps if necessary.
5. For your testbenches, those one have to be carefully written. As you did in Virtuoso® for your electrical simulations, you have to make sure that you test all (or as much as possible) the possible input cases to ensure that your system behaves correctly. For instance, in this tutorial, you should check on the XOR output curves that the 4 input cases (00, 01, 10, 11) are all simulated and shows a correct XOR output.

10 Post-Synthesis and P&R Functional Verification

After the design has been synthesized, you should perform another simulation to ensure that it still behave correctly. Below are some general guidelines:

1. You will need to create a new project. The RTL and post-synthesis simulations each need to be separate. As such, you can for instance replace *xor2_rtl* by *xor2_mapped* for the project name, library name, etc.
2. The verilog files to include are different: you will need to include the mapped netlist you obtained after running Design Compiler (in the HDL/GATE folder). You also need to include the Verilog standard cell library which describe the behavior of the standard cells you used to synthesize your design. The file path is:
`/research/ece/lnis-teaching/Designkits/tsmc180nm/full_custom_lib/verilog/sclib_tsmc180.v.`
 Finally, you need to include your testbench, which should be the same as for your RTL simulation.
3. When creating the simulation configuration, you also need to link the SDF file. This file will back annotate your mapped Verilog netlist in order to represent the delays from the logic gates of your designs. To do so:
 - (a) As before, create a new simulation configuration and choose its name. Select the design unit as shown in the picture from the appropriate library. Don't forget to apply the full visibility to all modules as before.
 - (b) Go to the SDF tab, and click on *Add..*
 - (c) Browse to select the SDF file created by the synthesis step (located in the *design_compiler/SDF folder*).
 - (d) The *Apply to Region* field should be *xor2_tb/xor2_dut* where *xor2_tb* is the name of your top verilog testbench module *xor2_dut* is the name of the instantiated module in your testbench.
 - (e) Click on *OK* and *Save*.

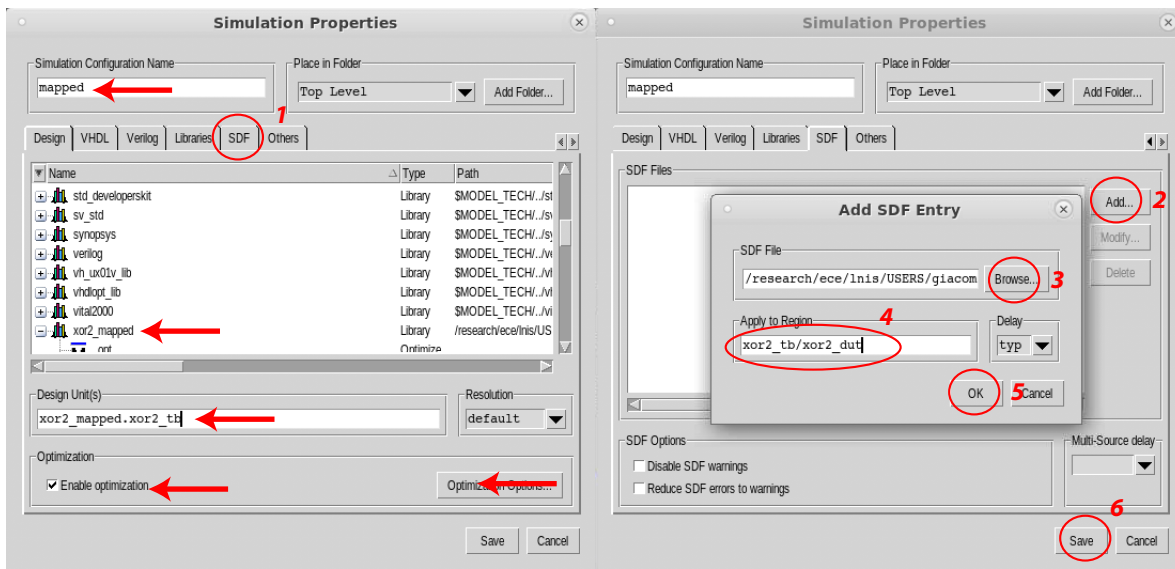


Figure 8: Post-synthesis simulation configuration.

4. For the post-P&R simulation, you will need to repeat this step. Don't forget to create another project. When adding the verilog files to your project, also add the pad library verilog file:
/research/ece/inis-teaching/Designkits/tsmc180nm/full_custom_lib/verilog/padlib_tsmc180.v.

R When doing post-synthesis and P&R simulations, don't forget to specify in your testbench the same period that you specified to constrain your design when synthesizing.