ModelSim Tutorial

By: Larbi Boughaleb, UT ECE

At the beginning of the semester, we will focus on getting acquainted with the VHDL syntax. We will use ModelSim 5.7 to simulate and verify the functionality of our VHDL code. This tutorial teaches the basic capabilities of ModelSim.

I. Starting ModelSim

1. Start ModelSim

   Startà Programsà ModelSim à ModelSim

2. After starting ModelSim, you should see the **Welcome to ModelSim 5.7g dialog**. If the dialog does not show up, you can display by selecting **Helpà Welcome Menu** from the Main window (See Figure 1).

Figure 1: "Welcome to ModelSim 5.7g" dialog

II. Setting up the Project

The first thing to do is to create a project. Projects ease the interaction with ModelSim and are useful for organizing files and simulation settings.

1. Create a new project by clicking on **Jumpstart** (See Figure 1) on the Welcome to ModelSim dialog, and then on **Create a Project.** You can also create a new project without the help of the dialog window by selecting: **File à New à Project** from the Main window.

2. A “Create Project” window pops up (See Figure 2). Select a suitable name for your project; set the Project Location to D:/Temp as shown above, and leave the Default Library Name to work. Hit Ok.

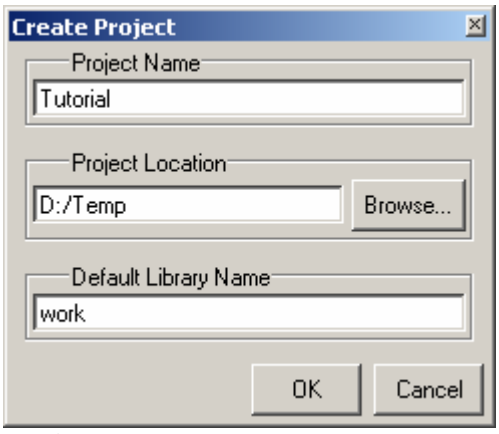


Figure 2: “Create Project” dialog

3. After hitting OK, an **Add items to the Project** dialog pops out (See Figure 3).

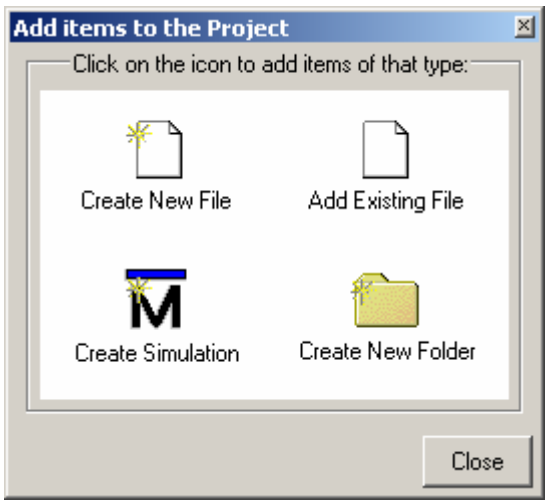


Figure 3: “Add items to the Project” dialog

We now have three options to add files to the project:

- We can create new VHDL files (from scratch) and then add them to the project, or
- We can add already existing files to the project, or
- We can do a combination of the two operations by combining the two above operations.

We will first illustrate the method of adding new files to the project we have just created.

Creating a VHDL File from Scratch

1. From the “Add items to the Project” dialog click on **Create a new file**.
   If you have closed the “Add items to the Project” dialog, then select **File à Add to Project à New File**

2. A Create Project File dialog pops out. Select an appropriate file name for the file you want to add; choose VHDL as the add file as type option and Top level as the Folder option (See Figure 4).

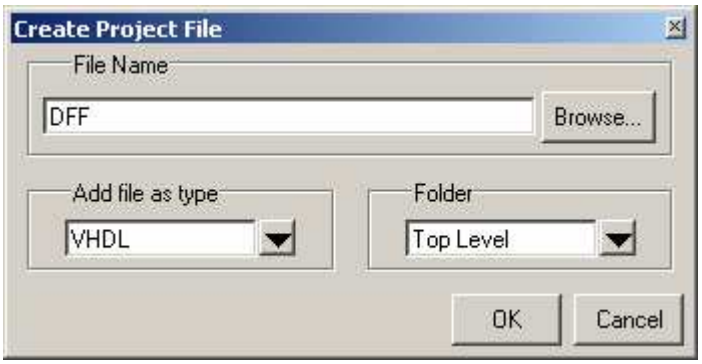


Figure 4: "Create Project File" dialog

3. On the workspace section of the Main Window (See Figure 5), double-click on the file you have just created (DFF.vhd in our case).

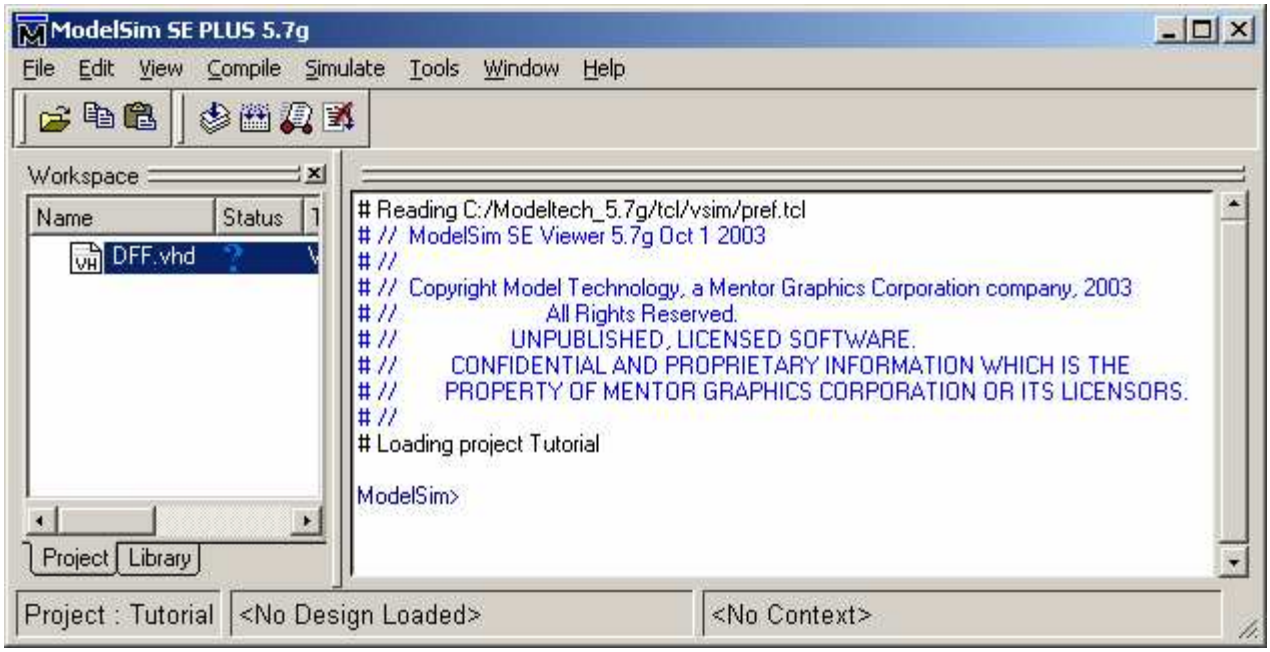


Figure 5: "ModelSim's Main Window"

4. Type in your code in the new window. For our tutorial, we will use a simple D flip-flop code from our textbook "**Digital Systems Design Using VHDL":**

```
entity DFF is
port (D, CLK: in bit;
Q: out bit; QN: out bit := '1');
end DFF;

architecture SIMPLE of DFF is
begin
```

```
process (CLK)
begin
if CLK = '1' then
Q <= D after 10 ns;
QN <= not D after 10 ns;
end if;
end process;
end SIMPLE;
```

5. Type (simply paste) the above code in the new window; however, leave out the last semicolon of this code. This will illustrate error correction using the ModelSim compiler.

6. Save your code (**File à Save).**

Adding Files to the Project

1. Select **File à Add to Project à Existing File**…

2. An "add file to Project" dialog pops up. Select the file that you want to add to the project. Also, make sure that you select VHDL from "the Add file as type" menu. Hit ok.

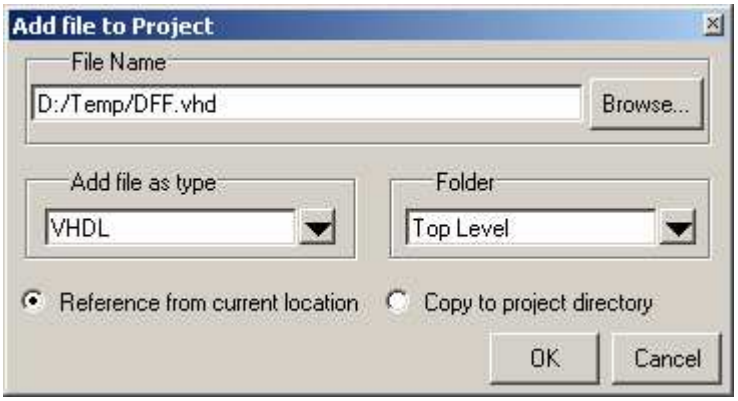


Figure 6: "Add file to Project" window

3. You should now see the file that you have just added in the workspace section of ModelSim's Main window.

Compiling / Debugging Project Files

1. Select **Compile à Compile All**.

2. The compilation result is shown on the main window. A red message indicates that there is an error in our code. Steps 3 through 7 will illustrate how to correct this error.

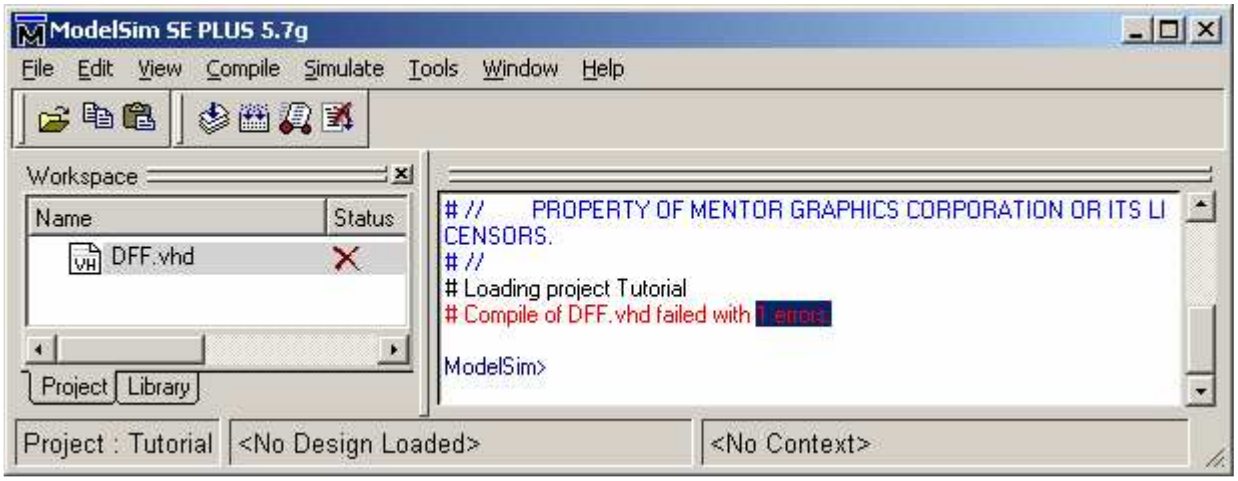
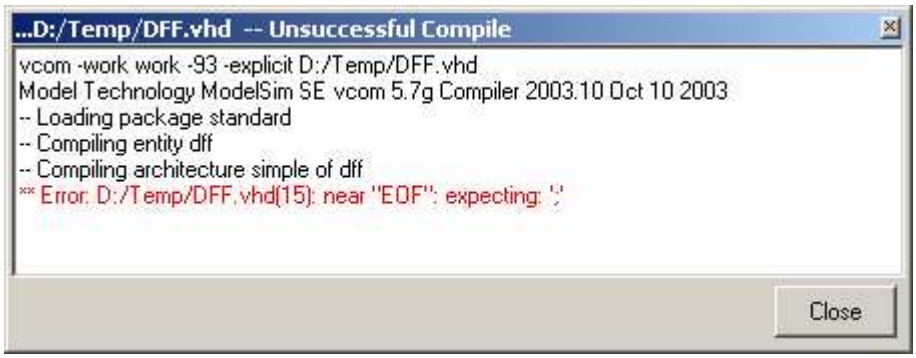


Figure 7: "The error is indicated in red on the main window"

3. Double-click on the error (shown in Red) on the main window. This will open a new window that describes the nature of the error. In our case the error message is as follows:



4. Double-click on the Error message. The error is highlighted in the source window:

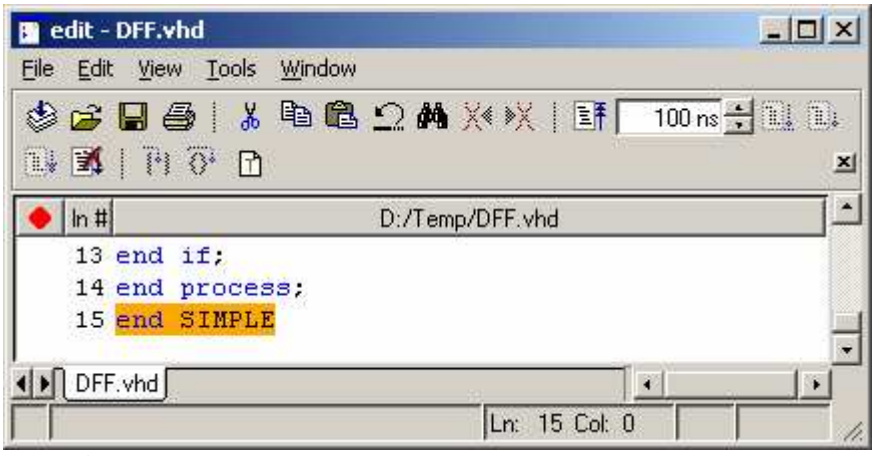


Figure 8: "The error is highlighted in the source window"

5. Correct the above error by adding the semicolon after the "end SIMPLE" statement. Hit save, and then recompile the file again. Repeat steps 1-5 until the code compiles with no errors.

Simulating the Design

This section covers the basics for simulating a design using ModelSim.

1. Click on the Library tab of the main window and then click on the (+) sign next to the work library. You should see the name of the entity of the code that we have just compiled "DFF". (See Figure 9)

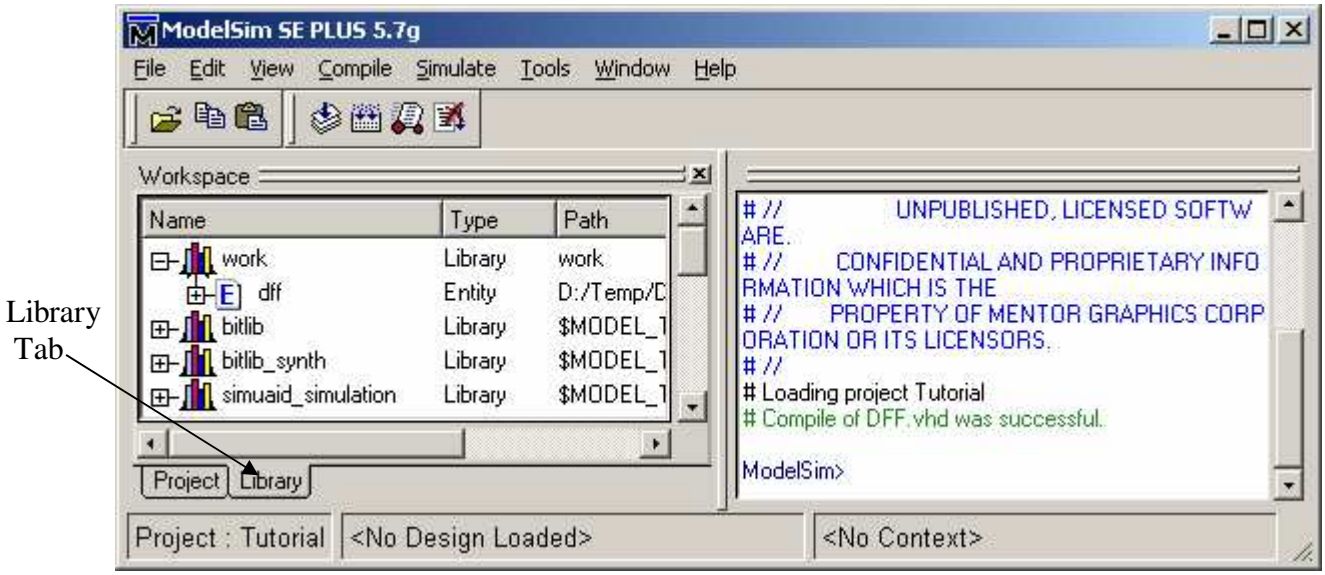


Figure 9: ModelSim's Main Window

2. Double-click on dff to load the file. This should open a third tab "sim" in the main window.

3. Now select **view à All Windows** from the main window to open all ModelSim windows.

4. Locate the signals window and select the signals that you want to monitor for simulation purposes. For this tutorial, select all signals as shown below.

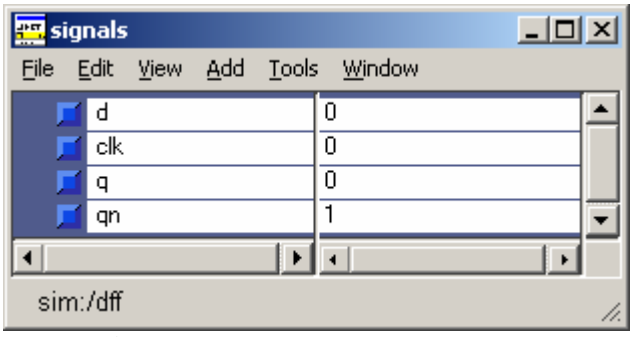


Figure 10: "The Signals Window"

5. Drag the above signals using the left button of the mouse into the wave window. You can also use: **add à wave à Selected signals**.

6. Do the same as in step 5 with the list window (i.e. drag selected signals into the list window, or use **add à list à Selected signals**).

7. We are now ready to simulate our design. For this purpose, we will need to type in simulation commands on the main window of the simulator. Refer to the document "common MXE commands", located on the course website, for some useful ModelSim commands.

8. On the main window type in: **force clk 0 0 ns, 1 10 ns -repeat 20 ns** and then hit enter. This statement forces the clock to take the value of 0 at 0ns, 1 at 10 ns and to repeat the forcing of these values every 20 ns.

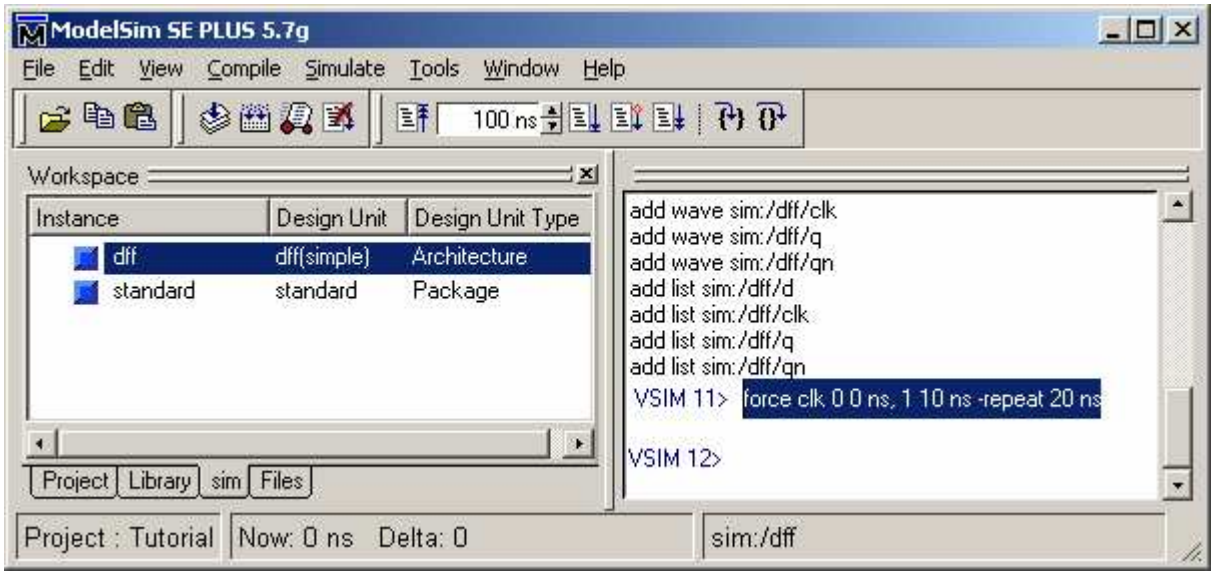


Figure 11: "Input the commands to the simulator as shown above"

9. Next, type **run 40 ns** on the main window and then hit enter. This will run the simulation for 40 ns. You can see the changes in the both the wave and list windows.

10. Next, change the value of D to 1 by typing: **force d 1** and then hit enter.
The change in d will take place at 40 ns (the current time)

11. Again, type **run 40 ns** on the main window to simulate the code for another 40ns.

12. Now, select the wave window, and click on "**zoom full**" (see below). Your simulation should look as follows.

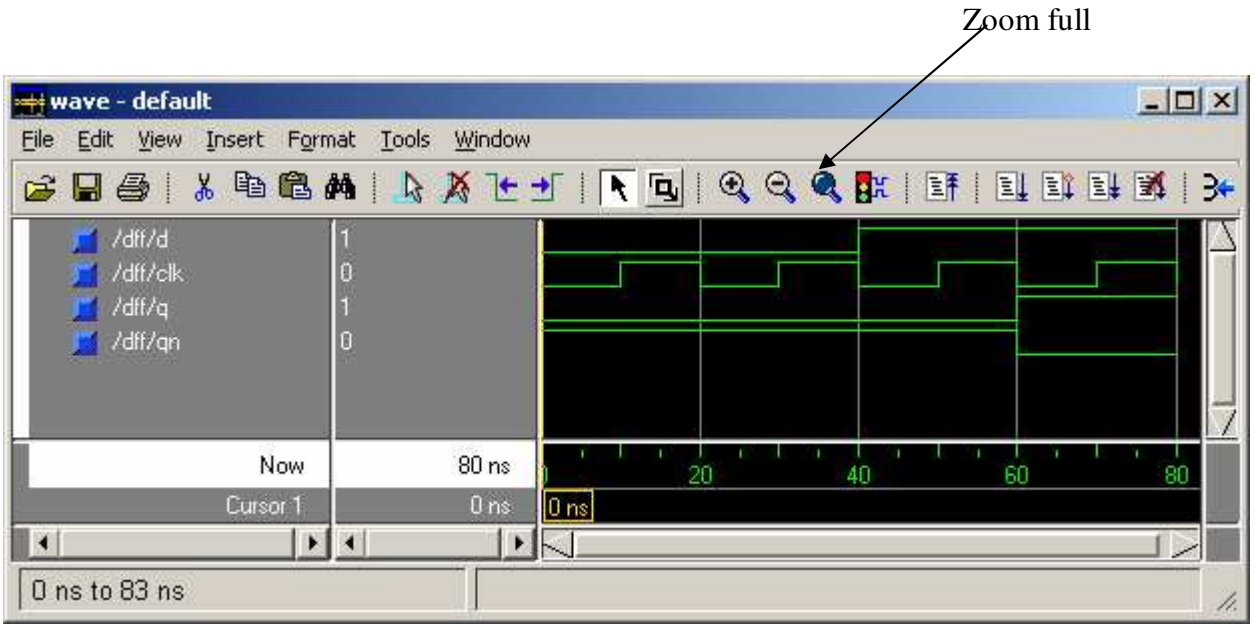


Figure 9: "the Wave window"

The above waveform shows that q follows d 10 Ns after the rising edge of the clock. The same result is confirmed using the list window.

Printing the Results:

- Printing the list window:
  1. In order to print the list window, select **File à write list à Tabular** from the list window and then save it as simulation.lst
  2. Start Notepad and open the file that you saved above. You can then proceed to printing it.

- Printing the wave window:
  Printing from the wave window is simple as you only need to select: **File à Print** from the wave window. However, make sure that your printed waveform can be read (Numbers may be printed too small if you try to print a long simulation waveform to one page).

Notes:

- You may find some differences in ModelSim behavior when using it in different labs. For example, let's assume that you have an output that you want to initialize to 0 at the beginning of the simulation. Assuming also that you have not initialized this output to 0 in your code, you may simply type: ***force Z 0 0 ns.*** You may note after running the simulation that Z never changes. This is because the options of some simulators were not set up properly during installation. To overcome this problem, change the above

statement to: ***force –deposit Z 0 0 ns***. The deposit will simply deposit the value of 0 to Z at 0ns instead of freezing it at 0.

- You can create a file of simulator commands (such as force, run…) and save it with a ".do" extension. You can then simply click on **Tools**à **Execute Macro** and then select your commands file to execute these commands. Note that this process is very convenient when you are debugging your code and you have to execute the same commands again and again.