ModelSim is a simulation and debugging environment created by Mentor Graphics. ModelSim allows you to check the syntax and verify the functionality of VHDL programs. This tutorial teaches the basic capabilities of ModelSim.

I. Starting ModelSim

To start ModelSim, click **Start → Programs→ ModelSim → ModelSim.**

II. Creating a Project

The first step in using ModelSim is creating a project. Projects ease the interaction with ModelSim and are useful for organizing files and simulation settings.

When you started ModelSim, a welcome dialog may have appeared (see Figure 1). If the welcome Dialog did not appear, you can display it by selecting **Help → Welcome Menu**.

1. Click the "Jumpstart" button.

2. Click "Create a Project."

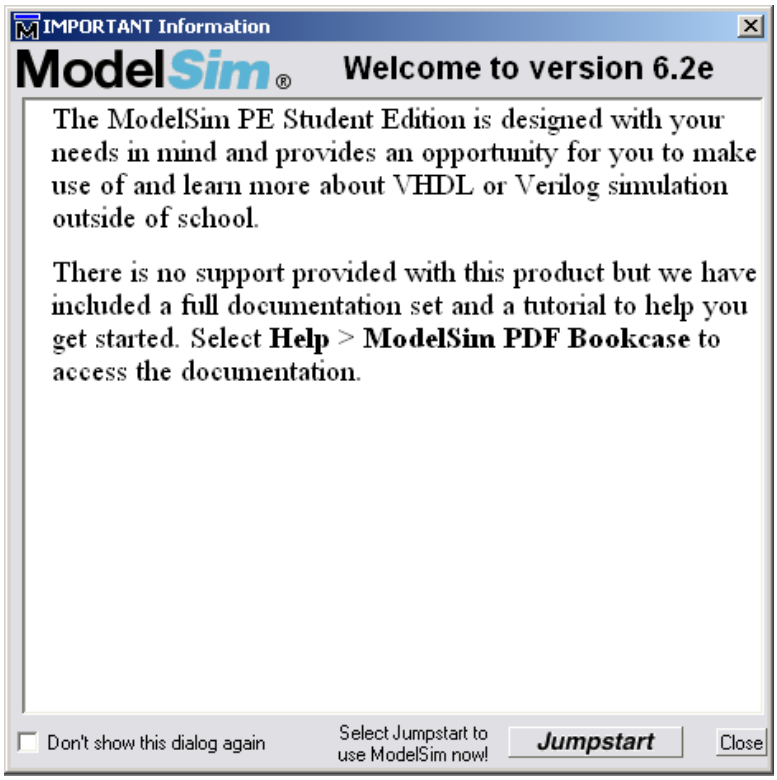

Figure 1: Welcome Menu

Instead of using the welcome menu, you can create a new project by selecting **File → New → Project** from the menu bar in the main window.

3. A "Create Project" window appears (See Figure 2). Enter an appropriate name for your project in the "Project Name" field, set the location where you would like the project to be created, and do not modify any of the other options. For this tutorial, name your project "Tutorial." Hit OK.

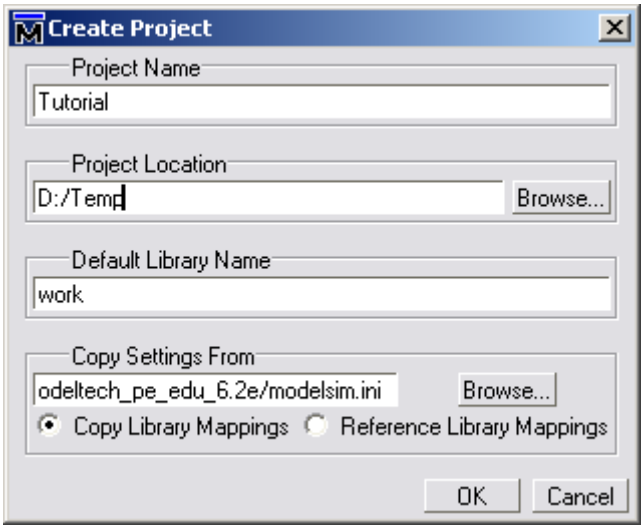

Figure 2: Create Project Dialog

4. After hitting OK, an "Add items to the Project**"** dialog box appears (See Figure 3).

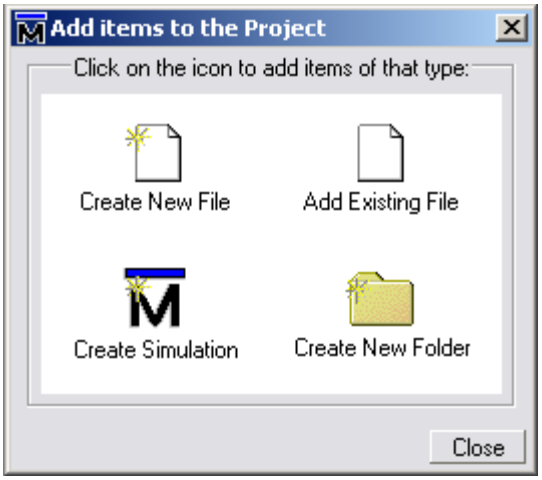

Figure 3: Add Items to the Project Dialog

There are three options to add files to the project:

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

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

Creating new VHDL files

1. In the "Add items to the Project" dialog box click on "Create New File." If you have closed the "Add items to the Project" dialog box, then select **Project → Add to Project → New File**.

2. The "Create Project File" dialog box appears. Enter an appropriate file name for your new file, choose "VHDL" in the "Add file as type" field, and choose "Top Level" in the "Folder" field (See Figure 4). For the tutorial, name your new file "DFF." Hit OK.

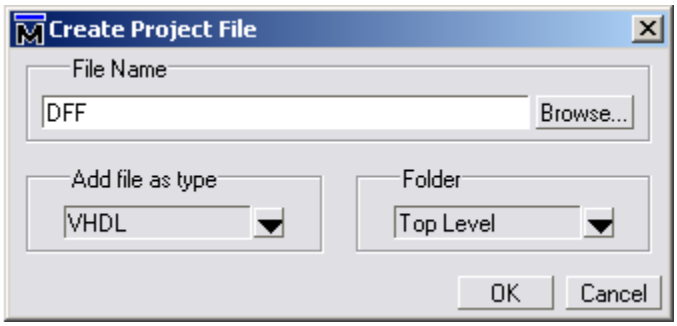

Figure 4: Create Project File Dialog

3. Click "Close" in the "Add items to the Project" dialog.

4. In the workspace window of the main interface (see Figure 5), double-click on the file you have just created (DFF.vhd).

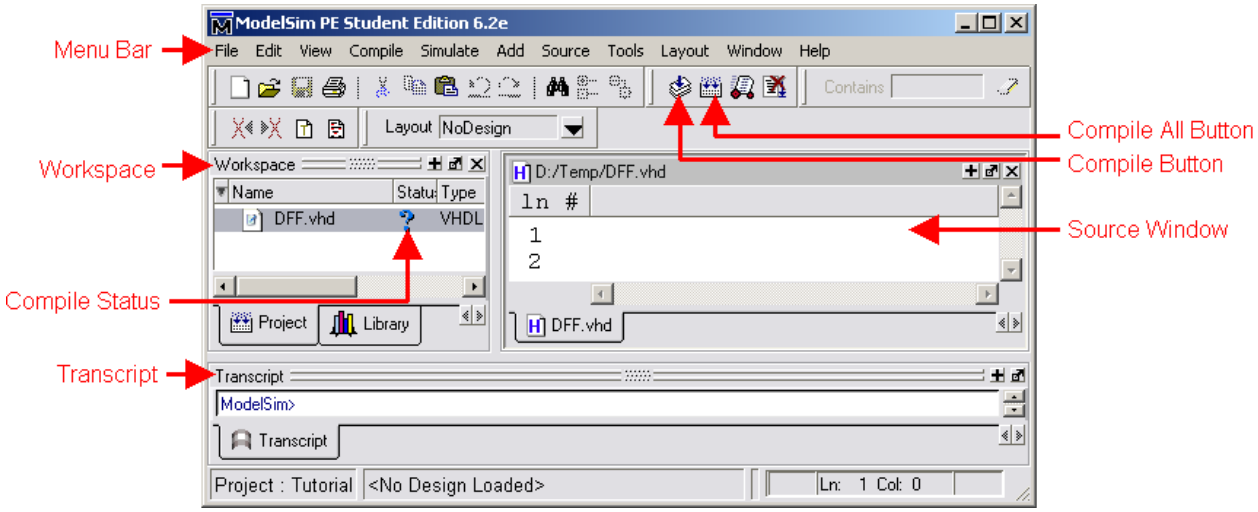

Figure 5: ModelSim's Interface

5. The source code of your file appears in a new source window. In this tutorial, we will use the VHDL code for a simple D flip-flop from the text.

   Before you begin entering code, make sure read-only mode is turned off by right-clicking in the source window and making sure "Read Only" in the menu that appears does not have a checkmark by it.

6. Type (or copy and paste) the following code in the source window; however, leave out the last semicolon of this code. This will illustrate error correction using the ModelSim compiler.

```
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'event and CLK = '1' then
      Q <= D after 10 ns;
      QN <= not D after 10 ns;
    end if;
  end process;
end SIMPLE;
```

7. Save your code (**File → Save).**

Adding files to the project

Instead of starting with a new file, you can add an existing file to the project. We do not need to add any more files to the project for this tutorial; however, the steps below can be used if you need to add a file in the future.

1. Click anywhere in the workspace window. Select **Project → Add to Project → 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 in the "Add file as type" field, and click on "Copy to project directory." Hit OK.

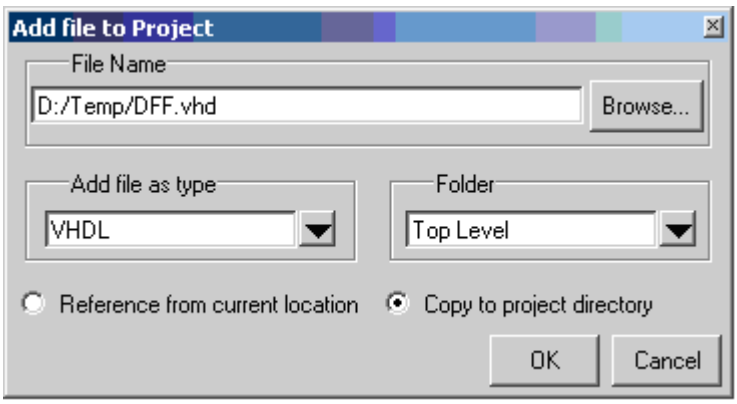

Figure 6: "Add file to Project" Window

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

III. Compiling and Debugging Project Files

1. Select **Compile → Compile All**.

2. The compilation result is shown in the transcript window. A red message indicates that there is an error in the code.

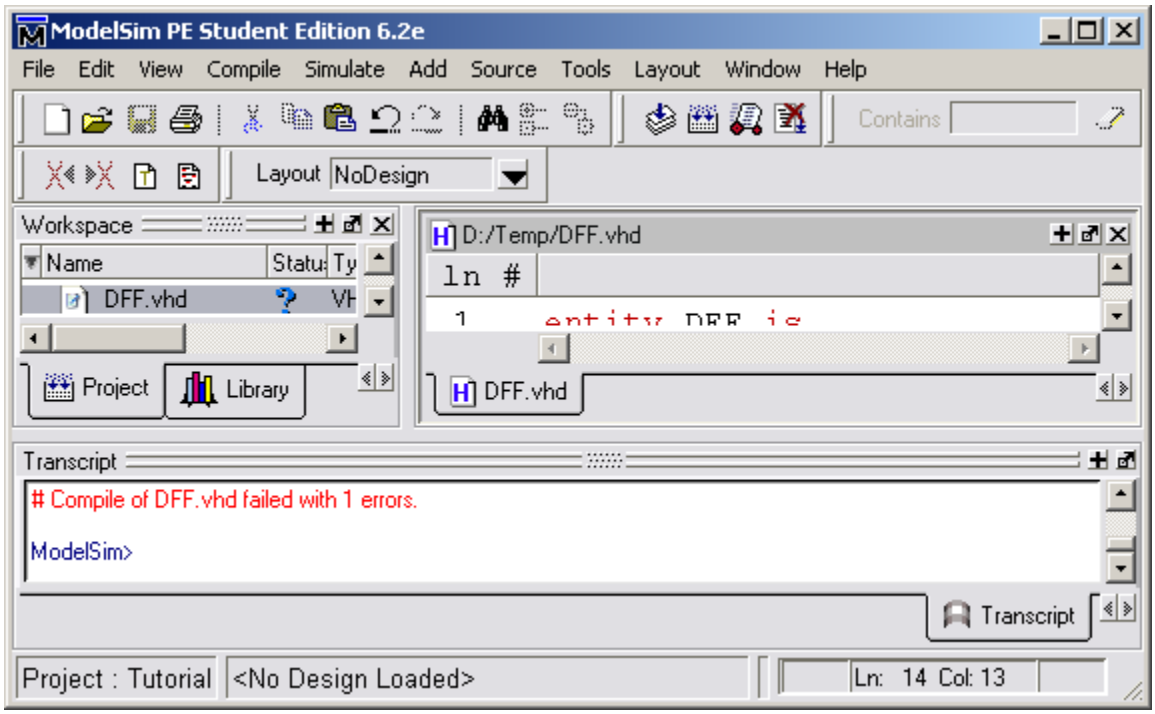

Figure 7: Compilation Result in the Transcript Window

3. Double-click on the compilation results in the transcript window. This will open a new window that describes the nature of the error. In our case, the error message is as follows:

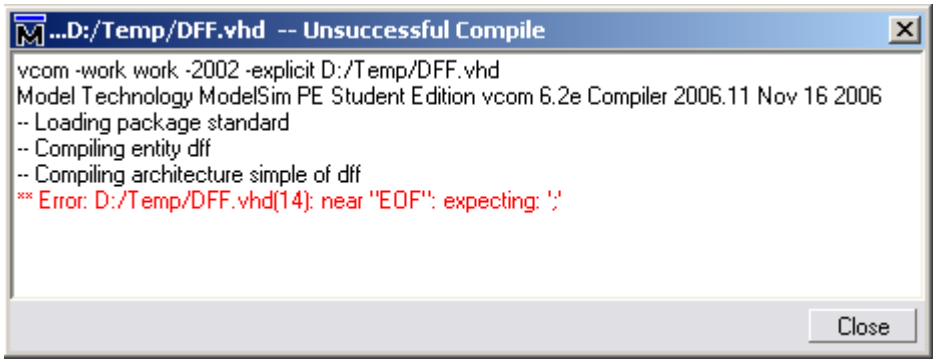

Figure 8: Error

4. Correct the above error by adding the semicolon after the "`end SIMPLE`" statement. Hit save, and then recompile the file. Repeat steps 2-4 until the code compiles with no errors.

IV. Simulating the Design

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

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

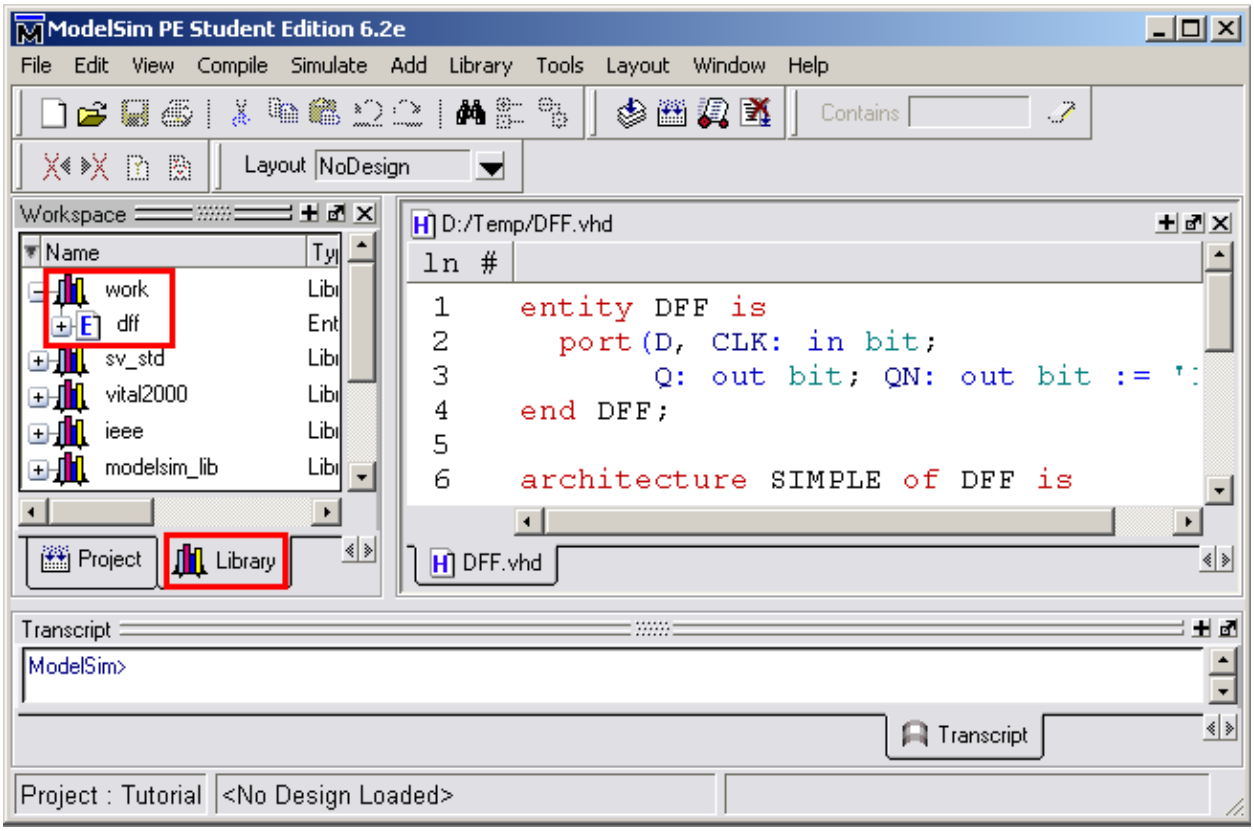

Figure 9: Library Tab

2. Double-click on "dff" to load the simulator. This should switch the workspace view to a third tab, named "sim." An objects window and additional buttons will also appear.

3. Select **View → List** and **View → Wave** from the menu bar. This will open two commonly used debugging windows.

4. Locate the objects window and select the signals that you want to monitor for simulation purposes. For this tutorial, select all of the signals by holding down the shift key and clicking on them.

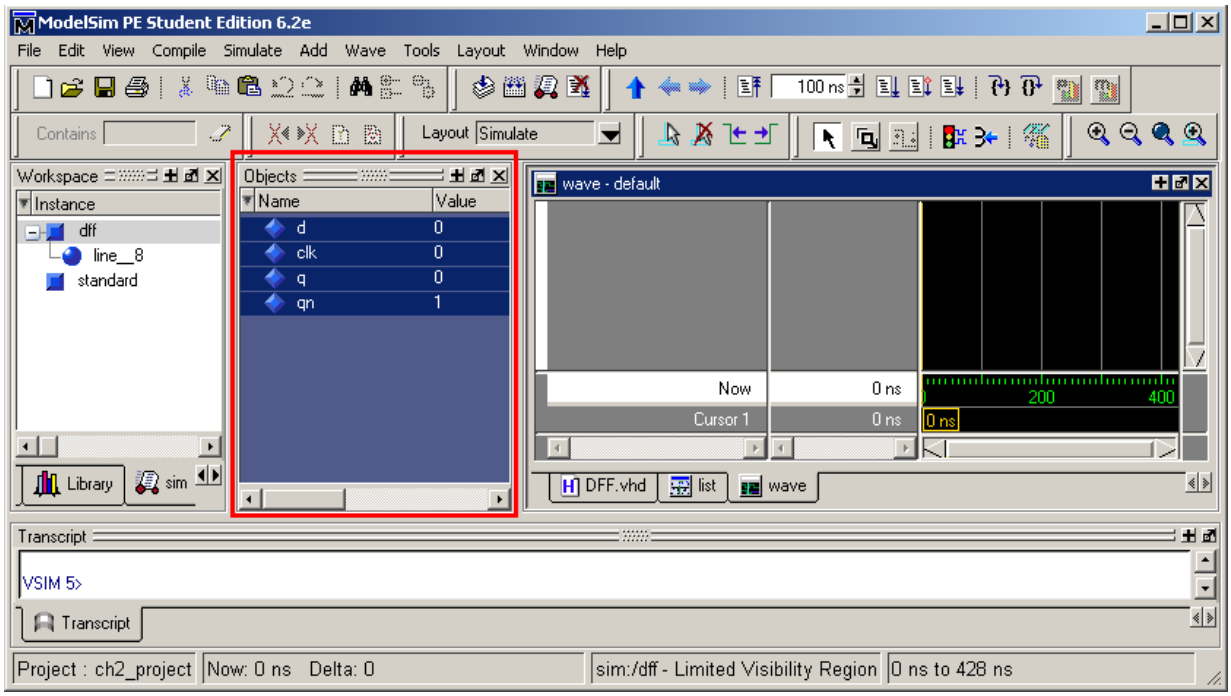

Figure 10: The Objects Window

5. Click and drag the signals into the wave window, using the left mouse button. You can also use the menu and click on **Add → Wave → Selected signals** when the objects window is selected.

6. Click and drag the signals into the list window, using the left mouse button, or click **Add → List → Selected signals** in the menu bar when the objects window is selected.

7. We are now ready to simulate our design. At the "VSIM" prompt in the transcript window, type "**force clk 0 0 ns, 1 10 ns -repeat 20 ns"** and then hit enter. This statement forces the *clk* signal to take the value of 0 at 0ns and 1 at 10 ns. The statement repeats itself every 20 ns, creating a clock signal with a 20 ns period.

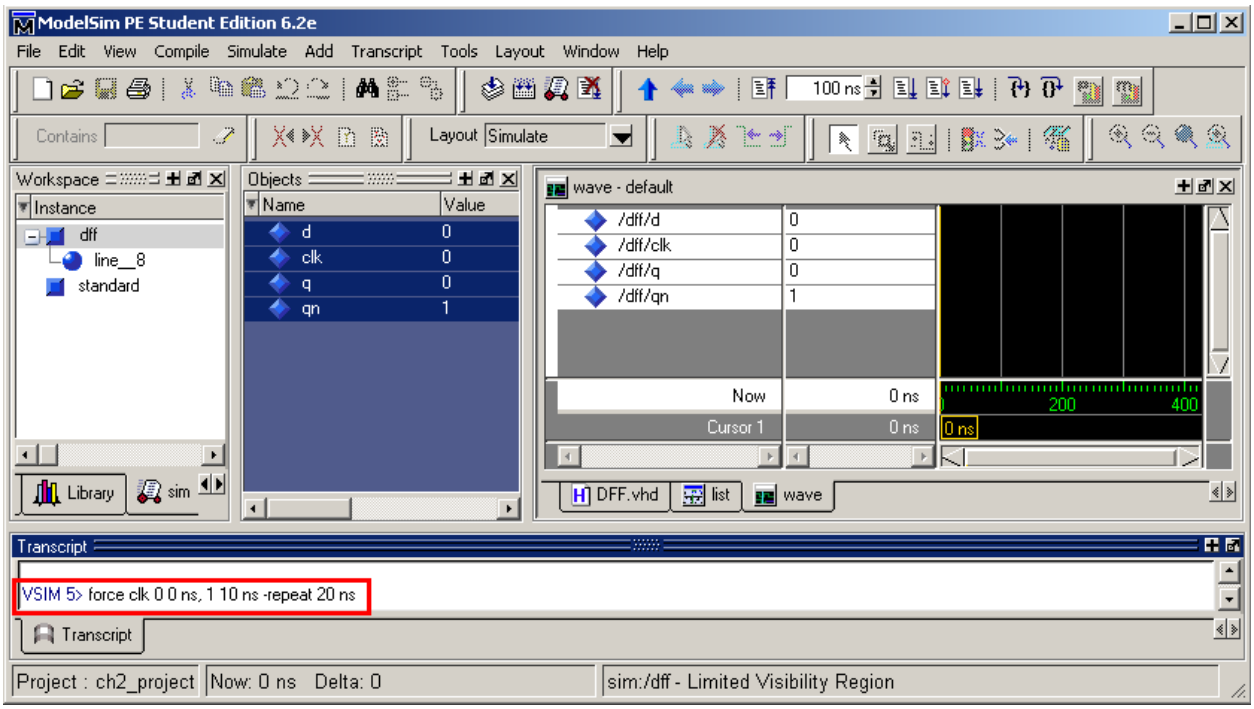

Figure 11: VSIM Prompt

8. Type "**run 40 ns"** at the "VSIM" prompt in the transcript window and then hit enter. This command will run the simulation for 40 ns. You can see changes in signal values in the wave and list windows.

9. Next, change the value of the signal *d* to 1 by typing "**force d 1"** and then hit enter. The change in *d* will take place at the current simulation time, 40 ns.

10. Type "**run 40 ns"** and then hit enter to simulate for another 40 ns. The simulator will continue from the current simulation time of 40 ns and finish at 80 ns.

11. Select the wave window, and click on the "zoom full" button (See Figure 12). Your simulation output should look similar to Figure 12.

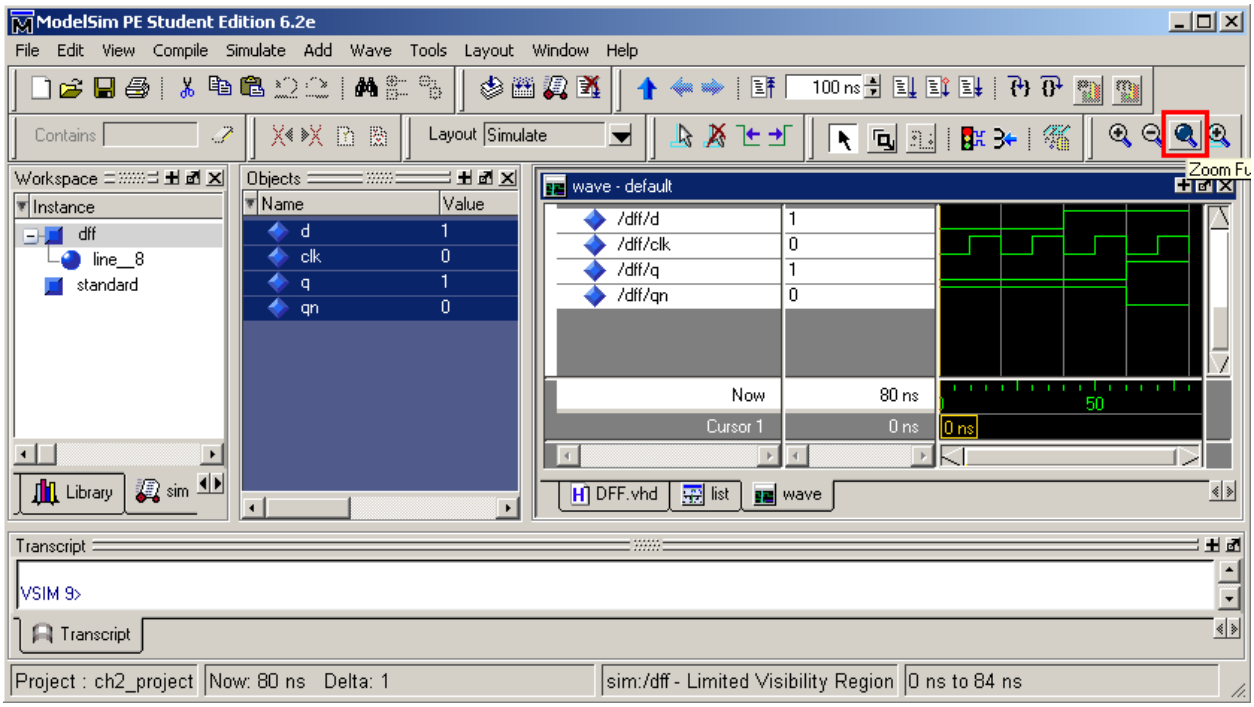

Figure 12: Simulation Results

The above waveforms show that *q* changes to the value of *d* 10 ns after the rising edge of the clock, as expected. The same result is confirmed using the list window.

Printing results

- Printing the list window
  1. In order to print the list window, first select it, and then select **File → Export → Tabular list** from the menu bar, and then save the file as "simulation.txt" to a convenient location.
  2. Start Notepad and open the file that you saved above. You can then print the file.

- Printing the wave window
  To print the wave window, simply select **File → Print** from the menu bar when the wave window is selected. Make sure that your printed waveform can be read. (Numbers may be printed too small if you try to print a long simulation waveform on one page.)

V. Notes

- When you use the **force** command, statements in your VHDL code that modify any signal you have forced will no longer have any effect. For example, if your code has a signal *Z* that you wish to initialize to 0 at the beginning of the simulation, you might use the command "**force Z 0 0 ns**." If you do this, *Z* will never change during simulation due to assignments in your program, even if you use the statement "`Z <= '1';`" as your first line of code. To overcome this problem, change the above statement to: "**force –deposit Z 0 0 ns**." The "deposit" option will set the value of *Z* to 0 at 0 ns and allow it to be changed by statements within your code.

- You can create a text file of simulator commands (such as **force**, **run**…) and save it with a ".do" extension. You can then simply click on **Tools → TCL → Execute Macro**, and then select your ".do" file to execute all of the commands in the file. Macros are extremely useful when you debug a large program and need to execute a large number of simulator commands each time you fix a bug in the program and wish to retest your code.