B

Appendix B: Siemens ModelSim Tutorial

As mentioned in Chapter 1, the most popular simulators for RTL design are Synopsys VCS, Cadence NC, and Siemens ModelSim. In this book, Siemens ModelSim is used because it provides a student-version license. Running a simulation using ModelSim typically involves eight steps, which can be carried out using either the GUI window or a TCL script. We first introduce the GUI-based operations and then recommend using TCL for simulation, as follows:

1. Step 1: Change the current working directory.
2. Step 2: Create a work library.
3. Step 3: Create a project.
4. Step 4: Add all the Verilog files to the project.
5. Step 5: Compile all the Verilog files.
6. Step 6: Simulate the Verilog design.
7. Step 7: Add the desired signals to the waveform viewer.
8. Step 8: Run the simulation.

We use the ddot design project, which was introduced in Chapter 9, as an example for this tutorial.

B.1   GUI Window

Below are the steps to use the GUI to run Siemens ModelSim.

Step 1: Change directory

To change the working directory to the **sim** folder of the project, follow these steps:

1. Click on File and select Change Directory, as shown in Fig. B.1.
2. Navigate to the **sim** folder of your project and click OK to confirm the change.

The Transcript window will show the same operation with the Linux/Unix command "cd".
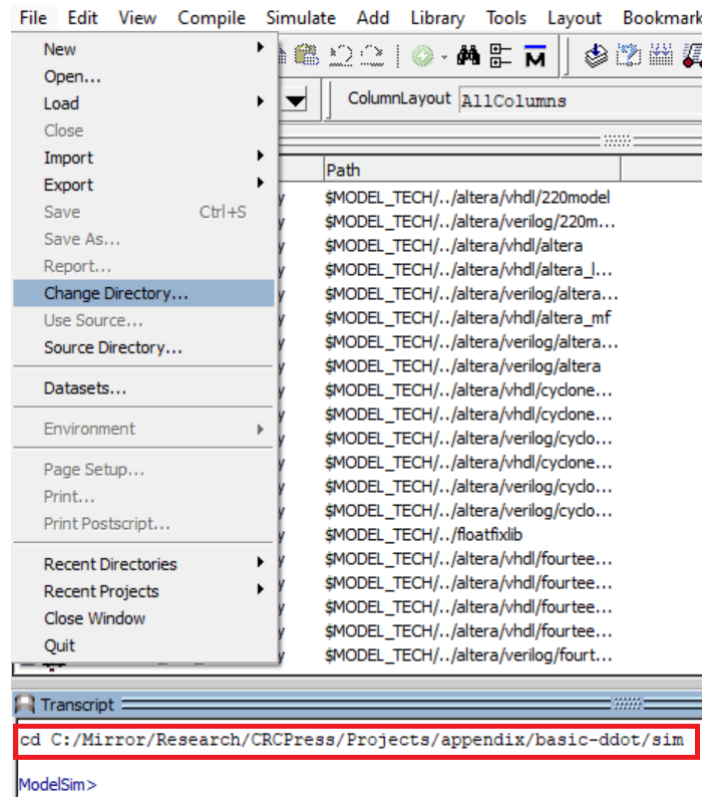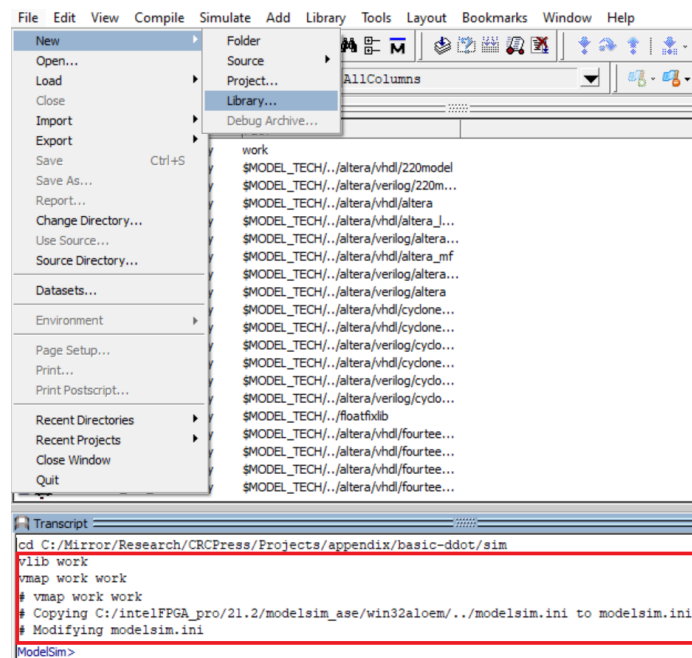

FIGURE B.1
Step 1: Changing the working directory.

Step 2: Create a work library

To create a new work library and map it in your working directory, follow these steps:

1. Click on File, select New, and then Library, as shown in Fig. B.2.
2. Enter the default library name "work" and click OK.

The Transcript window will show the same operations with the "vlib" and "vmap" commands.
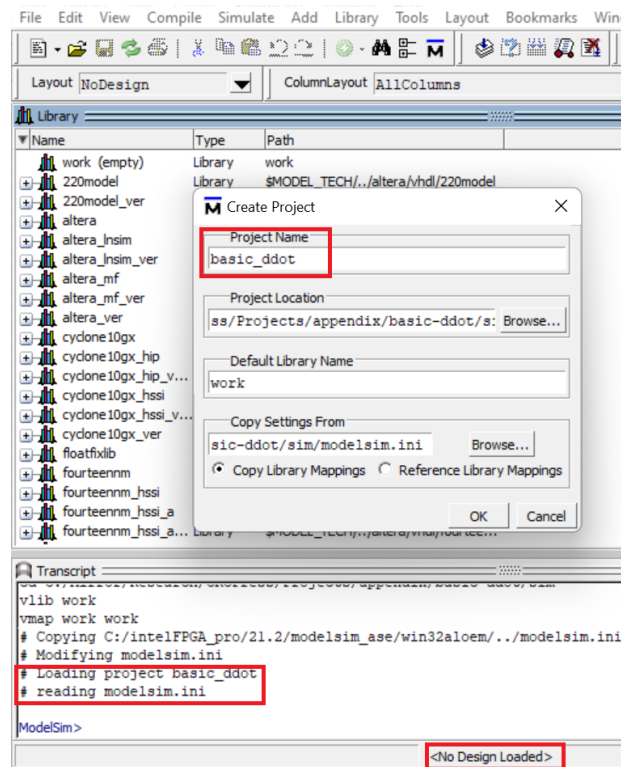
FIGURE B.2
Step 2: Creating a work library and mapping it in the working directory.

Step 3: Create a project

To create a new project, follow these steps:

1. Click on File, select New, and then Project, as shown in Fig. B.3.
2. Enter the project name "basic_ddot" in the Project Name box and click OK.

The Transcript window will show "loading the project basic_ddot". However, the message "No Design Loaded" will appear at the bottom of the window since no Verilog code has been added yet.
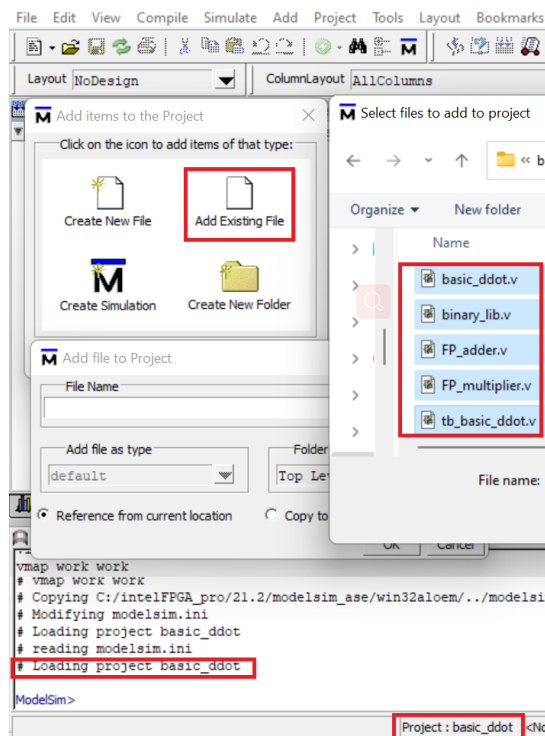

FIGURE B.3
Step 3: Creating a new project.

Step 4: Add all the Verilog files

After creating the project, you will see the "Add items to the Project" window as shown in Fig. B.4. Here, you can add the Verilog code to the project. If you do not have any Verilog code, you can click on "Create New File" to start

coding. However, since we already have the design code, click on "Add Existing File" and locate all the Verilog files, including the design code and testbench. It is important to note that all the files must be compiled in the next step. Additionally, ensure that the design files are located in the **dut** folder and the testbench is located in the **testbench** folder.

Once all the Verilog files have been added, you will find "Project: basic_ddot" at the bottom of the window.

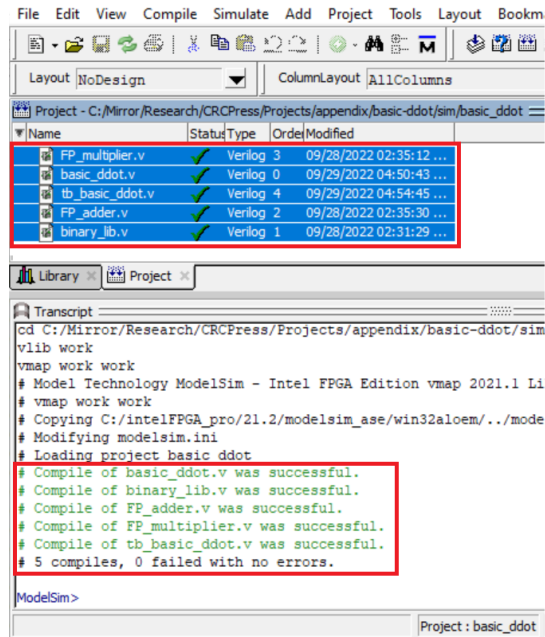**FIGURE B.4**
Step 4: Adding Verilog files to the project.

Step 5: Compile all the Verilog files

After adding all the Verilog code, you will see all the files added with question marks in the status column. This is because all the Verilog files have not yet been compiled. As shown in Fig. B.5, select all the ".v" files, right-click and select "Compile-Compile All". After clicking on "Compile All", all the Verilog files will be compiled. If the compilation is successful, the question marks in the status column will change into check marks. In the Transcript window, as
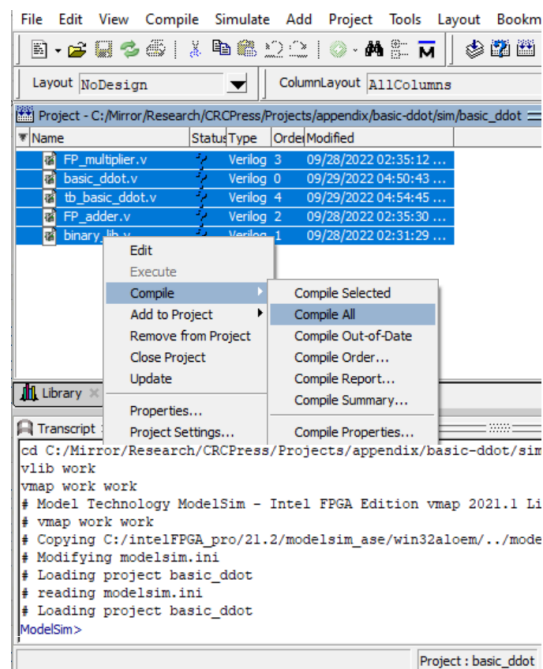
FIGURE B.5
Step 5.1: Compile all the Verilog files.

shown in Fig. B.6, you will find the same text for the compile operations. If no errors occur during compilation, it will show "0 failed with no errors."

FIGURE B.6
Step 5.2: Compilation of Verilog files.
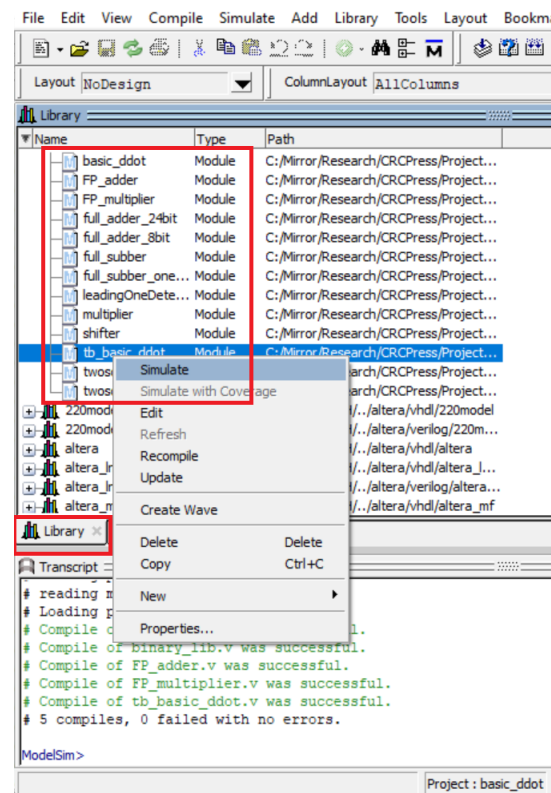
Step 6: Simulate the Verilog design

After the compilation of Verilog files, all the necessary operations in the project panel are finished. The next step is to simulate the design in the Library panel.

As shown in Fig. B.7, all the Verilog files are displayed in the Library panel. Notice that the simulation starts with the testbench, so select and right-click on the "tb_basic_ddot" file, then click on Simulate.

Step 7: Add signals to the waveform

After clicking Simulate, the simulation panel (or sim in Fig. B.8) will show the project hierarchy, with the top file being the testbench ("tb_basic_ddot"), followed by the top design module ("u_basic_ddot"). Note that the name of the top design module is the unique name instantiated in the testbench, and under the top design, all the FP adders and FP multipliers are instantiated.

In total, the ddot design instantiates eight FP multipliers ("u0_fp_mul", "u1_fp_mul", until "u7_fp_mul") and seven FP adders ("u0_fp_add",
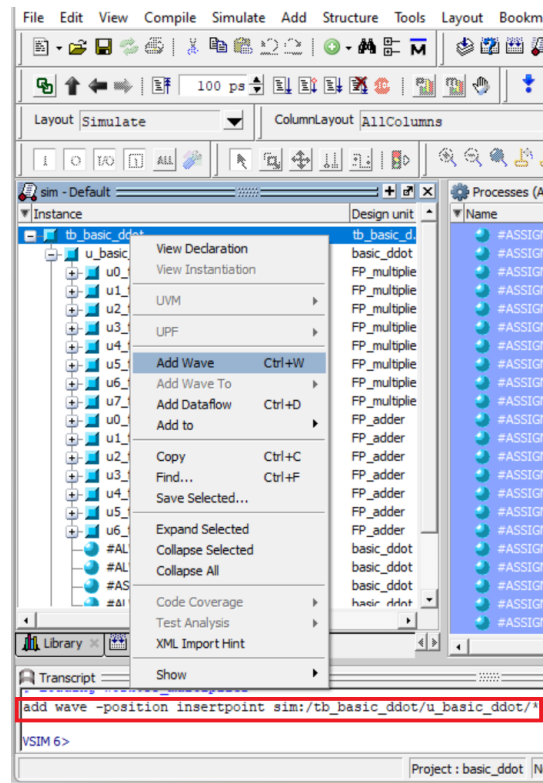
FIGURE B.7
Step 6: Simulating the Verilog design.
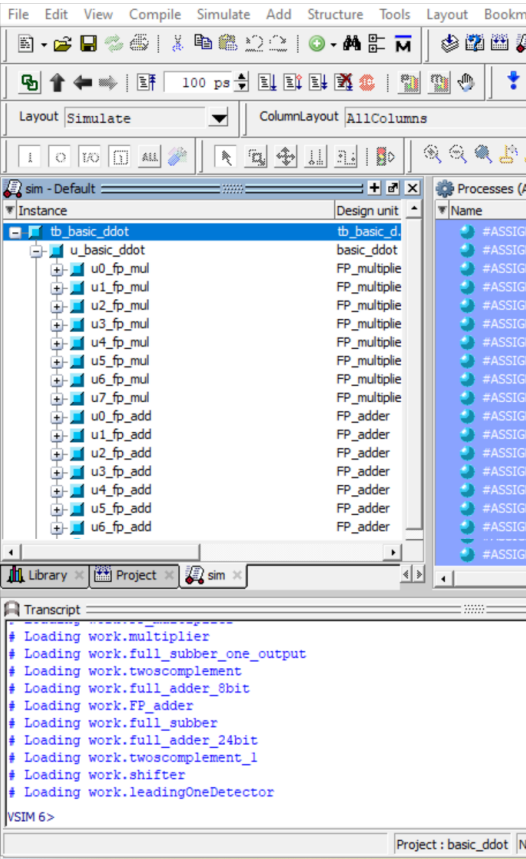
FIGURE B.8
Step 7.1: In the simulation panel, the project hierarchy is shown.

"u1_fp_add", until "u6_fp_add"). The names of the FP operators are the unique names instantiated in the top design module.

In the Transcript window, it shows that all the Verilog files in the work library are loaded. Before running the simulation, signals and IOs should be added into the waveform window. In Fig. B.9, all the signals in the "tb_basic_ddot" are added to the waveform by right-clicking the "tb_basic_ddot" and then clicking Add Wave. The command "add wave" is shown in the Transcript window.
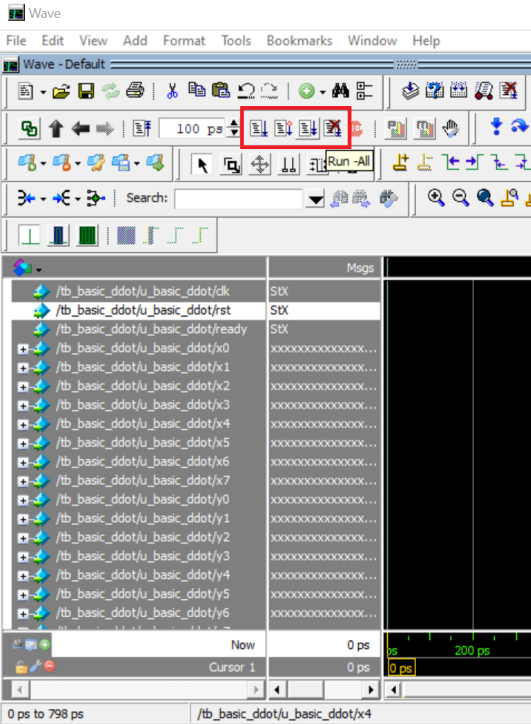

FIGURE B.9
Step 7.2: Adding signals to the waveform

Step 8: Run the simulation

After adding the signals, the waveform window will pop up with all the added signals shown in Fig. B.10. Click the Run-All button and then click Break to stop the running.

In Fig. B.11, all the values of the signals are shown in the waveform. The default data format is binary, but if you want to convert signals to hexadec-
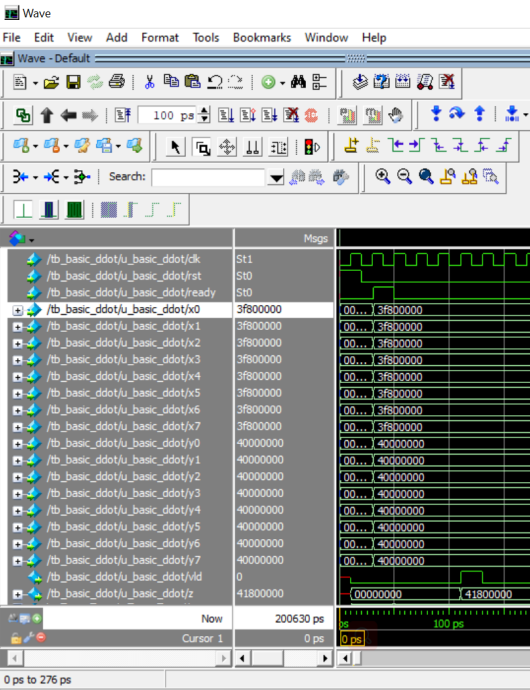
FIGURE B.10
Step 8.1: Run simulation and generate waveforms

imal, select all the signals you want to convert, right-click, and then click
Radix-Hexadecimal to convert the data format into hexadecimal.

FIGURE B.11
Step 8.2: Run simulation and generate waveforms

Finally, in Fig. B.12, it shows the final simulation waveform. All the inputs
"x0-x7" are single precision floating-point 1.0, so the IEEE 754 format will be
32'h3f800000. Similarly, the input "y0-y7" are floating-point 2.0, so the IEEE
754 format will be 32'h40000000. The ddot result "z" is floating-point 16.0
which is in IEEE 754 format 32'h41800000. By comparing the simulation result
with our expected timing diagram designs shown in Fig. ?? in Chapter ??,
we can ensure that our design features has been correctly verified.

If you make any changes to the Verilog code, you will need to recompile
the files, and then repeat steps 5-8. This process can be tedious for compli-
cated projects due to many rounds of debugging and simulation. Therefore,
an efficient way to simulate Verilog files is presented in the following section
using a TCL script.

FIGURE B.12
Step 8.3: Run simulation and generate waveforms

B.2   TCL Script

In IC design, simulation is a repetitive process for designing and debugging HDL code. To automate these activities, TCL scripts are commonly used in the industry. TCL is a scripting language that is extensively used by simulators built on it, including Siemens Modelsim.

As mentioned earlier, the Transcript window displays the command for each step in the simulation process. Rather than using GUI windows, a TCL script can be utilized to automatically conduct the simulation flow step by step. Below is an example TCL script named "run":

```
1  vlib work
2  vmap work work
3  vlog -f ../filelist/filelist.f
4  vsim work.tb_basic_ddot -t 1ns
5  add wave -radix hexadecimal tb_basic_ddot/u_basic_ddot/clk
6  add wave -radix hexadecimal tb_basic_ddot/u_basic_ddot/rst
7  add wave -radix hexadecimal tb_basic_ddot/u_basic_ddot/ready
8  add wave -radix hexadecimal tb_basic_ddot/u_basic_ddot/x0
9  add wave -radix hexadecimal tb_basic_ddot/u_basic_ddot/x7
10 add wave -radix hexadecimal tb_basic_ddot/u_basic_ddot/y0
11 add wave -radix hexadecimal tb_basic_ddot/u_basic_ddot/y7
12 add wave -radix hexadecimal tb_basic_ddot/u_basic_ddot/vld
13 add wave -radix hexadecimal tb_basic_ddot/u_basic_ddot/z
14 run 1us
```

The simulation process involves eight steps:

1. Step 1 : changing directory
2. Steps 2-3: creating and mapping a library (lines 1-2)
3. Steps 4-5: compiling all the files listed in the filelist.f (line 3)
4. Step 6 : starting simulation (line 4)
5. Step 7 : adding signals to the waveform (lines 5-13)
6. Step 8 : running the simulation for 1 us (line 14).

The "filelist.f" is located in the **filelist** folder. To find it, navigate to the parent directory ("../"), enter the **filelist** folder (../filelist/"), and locate the "filelist.f" file ("../filelist/filelist.f"). In the "filelist.f" file, all the Verilog files are listed. Similarly, to find all the ".v" files in the **dut** folder, navigate to the parent directory (".."), enter the **dut** folder ("../dut/"), and locate all the ".v" files ("../dut/ddot_basic.v", "../dut/FP_multiplier.v", etc.).

The "run" script is located in the **sim** folder. To use the script, first change the directory to the **sim** folder, where all the generated files and libraries will be directed. Then, type "do run" in the Transcript window to automatically carry out all seven simulation steps (except for the first step "changing directory"). Note that "changing directory" is only necessary once. After that, simply type the "do run" command whenever re-compilation and re-simulation are needed. To read and debug more signals in the Verilog code, add more "add wave" lines to the run TCL script. Note that adding signals must follow the design hierarchy. For example, the added "clk" signal in line 5 is from the design module "u_basic_ddot". If the "clk" signal from the testbench is added, the hierarchy should be "tb_basic_ddot/clk".

Refer to Fig. B.13 for a visual representation of using the run script:

1. Change directory to the **sim** folder.

2. Type "do run" to carry out all simulation steps.
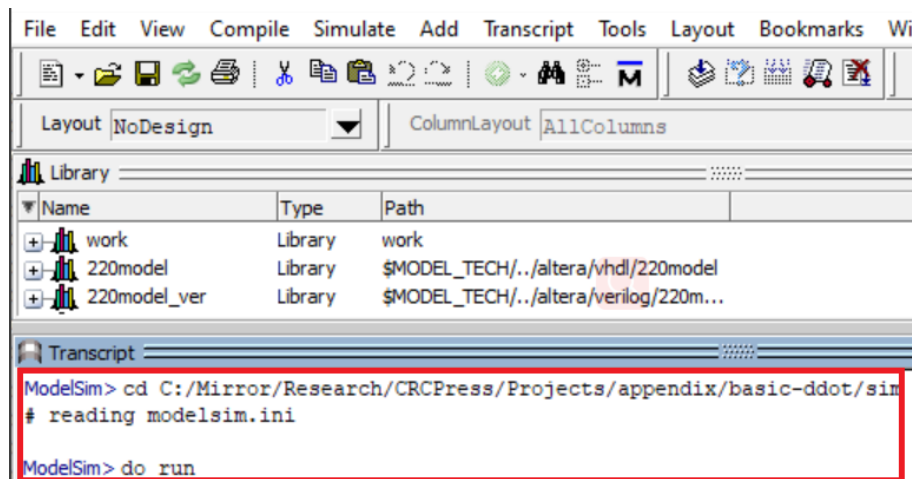
FIGURE B.13
Using run script: 1) change directory to **sim** folder; 2) do run