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 **??**, as an example for this tutorial.

B.1 GUI Window

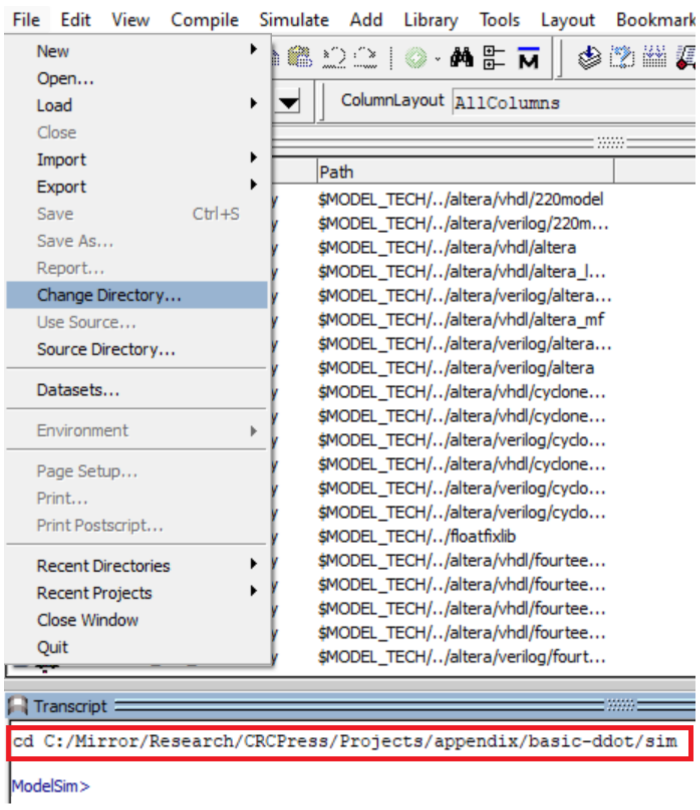
Below are the steps to use the GUI to run Siemens ModelSim.

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".



```
$MODEL_TECH/../altera/vhdl/220model
$MODEL_TECH/../altera/verilog/220m...
$MODEL_TECH/../altera/vhdl/altera
$MODEL_TECH/../altera/vhdl/altera_l...
$MODEL_TECH/../altera/verilog/altera...
$MODEL_TECH/../altera/vhdl/altera_mf
$MODEL_TECH/../altera/verilog/altera...
$MODEL_TECH/../altera/verilog/altera
$MODEL_TECH/../altera/vhdl/cyclone...
$MODEL_TECH/../altera/vhdl/cyclone...
$MODEL_TECH/../altera/verilog/cyclo...
$MODEL_TECH/../altera/vhdl/cyclone...
$MODEL_TECH/../altera/verilog/cyclo...
$MODEL_TECH/../altera/verilog/cyclo...
$MODEL_TECH/../floatfixlib
$MODEL_TECH/../altera/vhdl/fourtee...
$MODEL_TECH/../altera/vhdl/fourtee...
$MODEL_TECH/../altera/vhdl/fourtee...
$MODEL_TECH/../altera/vhdl/fourtee...
$MODEL_TECH/../altera/verilog/fourt...
```

```
cd C:/Mirror/Research/CRCPress/Projects/appendix/basic-ddot/sim

ModelSim>
```

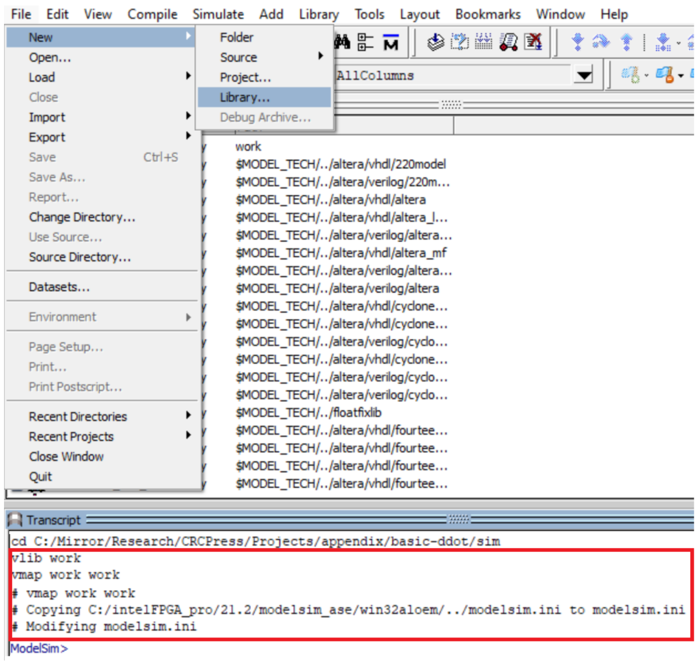
FIGURE B.1
Step 1: Changing the working directory.

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.


File Edit View Compile Simulate Add Library Tools Layout Bookmarks Window Help
New
Open...
Load
Close
Import
Export
Save Ctrl+S
Save As...
Report...
Change Directory...
Use Source...
Source Directory...
Datasets...
Environment
Page Setup...
Print...
Print Postscript...
Recent Directories
Recent Projects
Close Window
Quit
Folder
Source
Project...
Library...
Debug Archive...
AllColumns
work
$MODEL_TECH/../altera/vhdl/220model
$MODEL_TECH/../altera/verilog/220m...
$MODEL_TECH/../altera/vhdl/altera
$MODEL_TECH/../altera/vhdl/altera_l...
$MODEL_TECH/../altera/verilog/altera...
$MODEL_TECH/../altera/vhdl/altera_mf
$MODEL_TECH/../altera/verilog/altera...
$MODEL_TECH/../altera/verilog/altera
$MODEL_TECH/../altera/vhdl/cyclone...
$MODEL_TECH/../altera/vhdl/cyclone...
$MODEL_TECH/../altera/verilog/cyclo...
$MODEL_TECH/../altera/vhdl/cyclone...
$MODEL_TECH/../altera/verilog/cyclo...
$MODEL_TECH/../altera/verilog/cyclo...
$MODEL_TECH/../floatfixlib
$MODEL_TECH/../altera/vhdl/fourtee...
$MODEL_TECH/../altera/vhdl/fourtee...
$MODEL_TECH/../altera/vhdl/fourtee...
$MODEL_TECH/../altera/vhdl/fourtee...
Transcript

```
cd C:/Mirror/Research/CRCPress/Projects/appendix/basic-ddot/sim
vlib work
vmap work work
# vmap work work
# Copying C:/intelFPGA_pro/21.2/modelsim_ase/win32aloem/../modelsim.ini to modelsim.ini
# Modifying modelsim.ini
ModelSim>
```

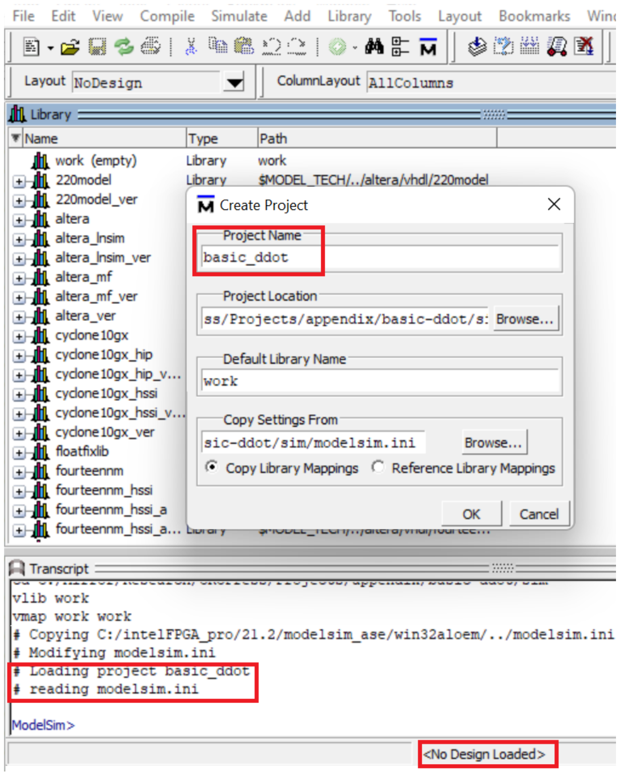
FIGURE B.2
Step 2: Creating a work library and mapping it in the working directory.

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.



Layout: NoDesign
ColumnLayout: AllColumns

Name	Type	Path
work (empty)	Library	work
220model	Library	$MODEL_TECH/../altera/vhdl/220model
220model_ver		
altera		
altera_lnsim		
altera_lnsim_ver		
altera_mf		
altera_mf_ver		
altera_ver		
cyclone10gx		
cyclone10gx_hip		
cyclone10gx_hip_v...		
cyclone10gx_hssi		
cyclone10gx_hssi_v...		
cyclone10gx_ver		
floatfixlib		
fourteennm		
fourteennm_hssi		
fourteennm_hssi_a		
fourteennm_hssi_a...	Library	$MODEL_TECH/../altera/vhdl/fourtee...

Create Project
Project Name: basic_ddot
Project Location: ss/Projects/appendix/basic-ddot/s:
Default Library Name: work
Copy Settings From: sic-ddot/sim/modelsim.ini
[✓ Copy Library Mappings] Reference Library Mappings

```
vlib work
vmap work work
# Copying C:/intelFPGA_pro/21.2/modelsim_ase/win32aloem/../modelsim.ini
# Modifying modelsim.ini
# Loading project basic_ddot
# reading modelsim.ini

ModelSim>
```

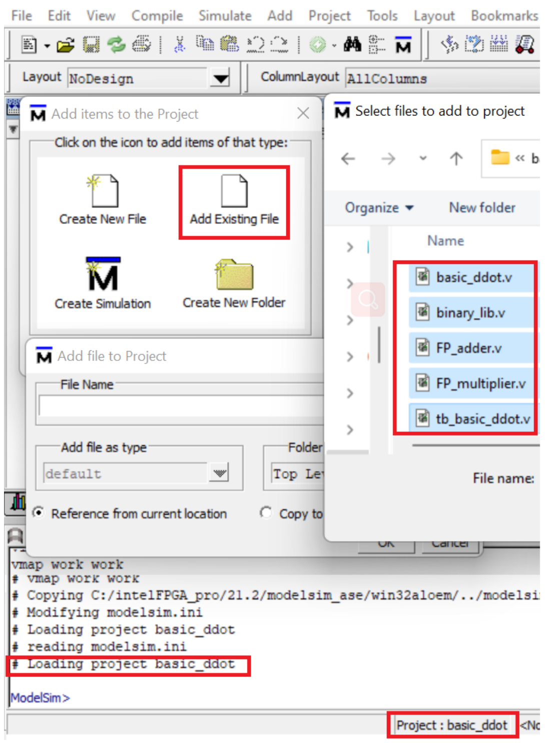
<No Design Loaded>

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.



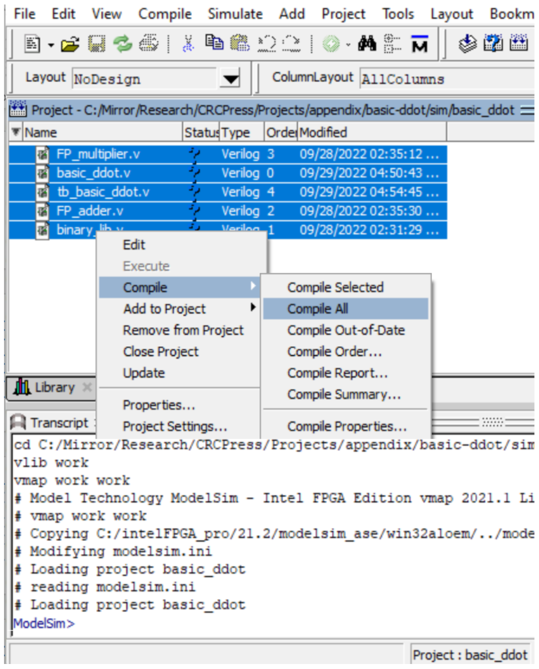
```
vmap work work
# vmap work work
# Copying C:/intelFPGA_pro/21.2/modelsim_ase/win32aloem/../modelsi
# Modifying modelsim.ini
# Loading project basic_ddot
# reading modelsim.ini
# Loading project basic_ddot

ModelSim>
```

Project : basic_ddot

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

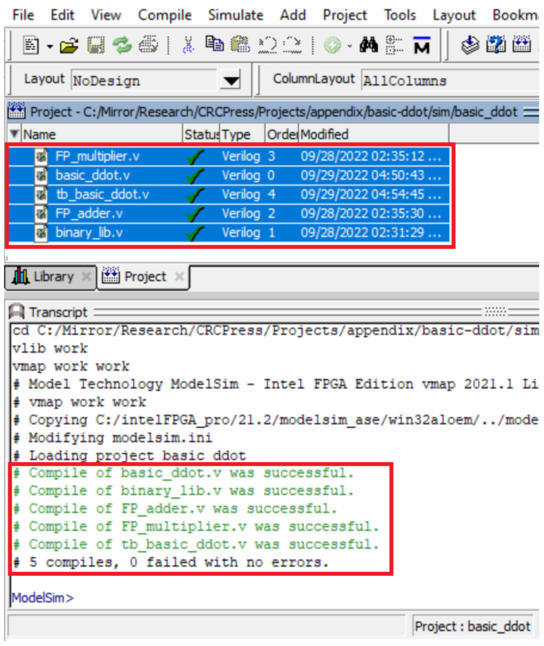


Name	Status	Type	Order	Modified
FP_multiplier.v	?	Verilog	3	09/28/2022 02:35:12 ...
basic_ddot.v	?	Verilog	0	09/29/2022 04:50:43 ...
tb_basic_ddot.v	?	Verilog	4	09/29/2022 04:54:45 ...
FP_adder.v	?	Verilog	2	09/28/2022 02:35:30 ...
binary_lib.v	?	Verilog	1	09/28/2022 02:31:29 ...

```
cd C:/Mirror/Research/CRCPress/Projects/appendix/basic-ddot/sim
vlib work
vmap work work
# Model Technology ModelSim - Intel FPGA Edition vmap 2021.1 Li
# vmap work work
# Copying C:/intelFPGA_pro/21.2/modelsim_ase/win32aloem/../mode
# Modifying modelsim.ini
# Loading project basic_ddot
# reading modelsim.ini
# Loading project basic_ddot
ModelSim>
```

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.”



Project - C:/Mirror/Research/CRCPress/Projects/appendix/basic-ddot/sim/basic_ddot

Name	Status	Type	Order	Modified
FP_multiplier.v	✓	Verilog	3	09/28/2022 02:35:12 ...
basic_ddot.v	✓	Verilog	0	09/29/2022 04:50:43 ...
tb_basic_ddot.v	✓	Verilog	4	09/29/2022 04:54:45 ...
FP_adder.v	✓	Verilog	2	09/28/2022 02:35:30 ...
binary_lib.v	✓	Verilog	1	09/28/2022 02:31:29 ...

```
cd C:/Mirror/Research/CRCPress/Projects/appendix/basic-ddot/sim
vlib work
vmap work work
# Model Technology ModelSim - Intel FPGA Edition vmap 2021.1 Li
# vmap work work
# Copying C:/intelFPGA_pro/21.2/modelsim_ase/win32aloem/../mode
# Modifying modelsim.ini
# Loading project basic ddot
# Compile of basic_ddot.v was successful.
# Compile of binary_lib.v was successful.
# Compile of FP_adder.v was successful.
# Compile of FP_multiplier.v was successful.
# Compile of tb_basic_ddot.v was successful.
# 5 compiles, 0 failed with no errors.

ModelSim>
```

Project : basic_ddot

FIGURE B.6
Step 5.2: Compilation of Verilog files.

Step 6: Simulate the Verilog design

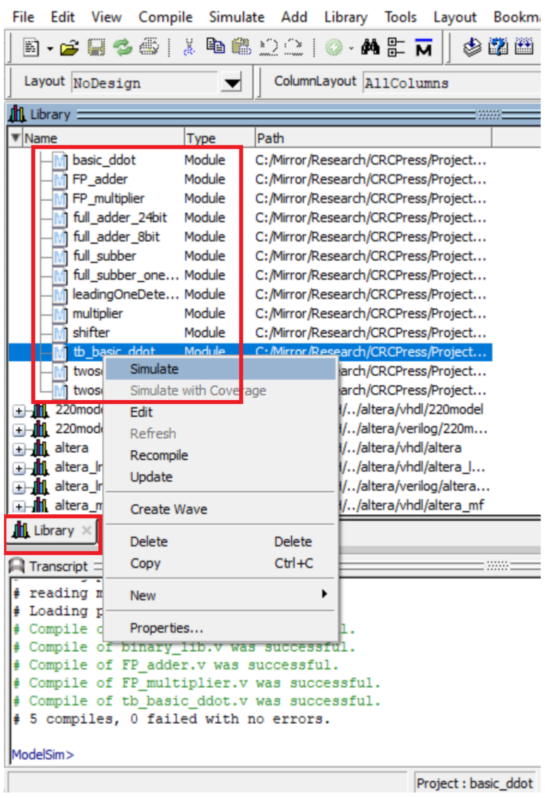
After the compilation of Verilog files, all the necessary operations in the project panel are finished. The next step is to simulate the design in the Library panel.

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.

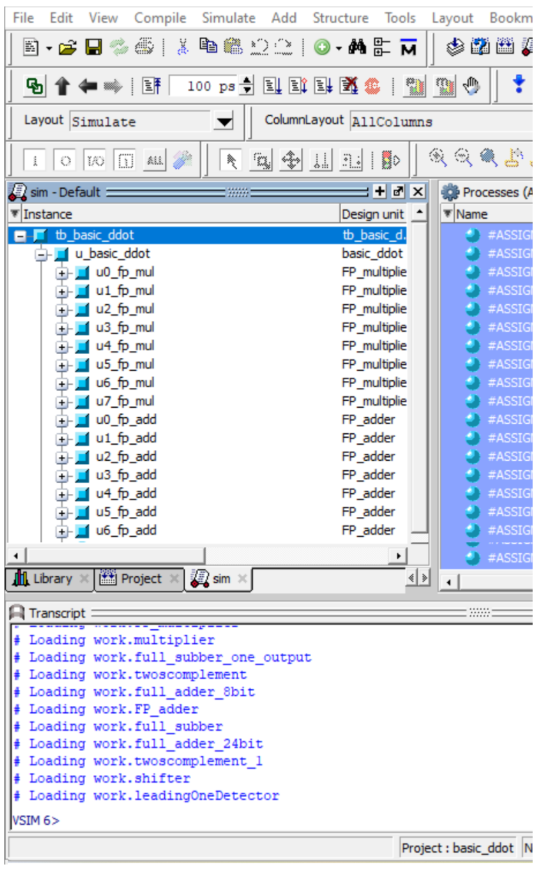
In total, the ddot design instantiates eight FP multipliers (“u0_fp_mul”, “u1_fp_mul”, until “u7_fp_mul”) and seven FP adders (“u0_fp_add”,



Name	Type	Path
basic_ddot	Module	C:/Mirror/Research/CRCPress/Project...
FP_adder	Module	C:/Mirror/Research/CRCPress/Project...
FP_multiplier	Module	C:/Mirror/Research/CRCPress/Project...
full_adder_24bit	Module	C:/Mirror/Research/CRCPress/Project...
full_adder_8bit	Module	C:/Mirror/Research/CRCPress/Project...
full_subber	Module	C:/Mirror/Research/CRCPress/Project...
full_subber_one...	Module	C:/Mirror/Research/CRCPress/Project...
leadingOneDete...	Module	C:/Mirror/Research/CRCPress/Project...
multiplier	Module	C:/Mirror/Research/CRCPress/Project...
shifter	Module	C:/Mirror/Research/CRCPress/Project...
tb_basic_ddot	Module	C:/Mirror/Research/CRCPress/Project...
twos		arch/CRCPress/Project...
twos		arch/CRCPress/Project...
220mod		/../altera/vhdl/220model
220mod		/../altera/verilog/220m...
altera		/../altera/vhdl/altera
altera_l		/../altera/vhdl/altera_l...
altera_l		/../altera/verilog/altera...
altera_m		/../altera/vhdl/altera_mf

```
# reading m
# Loading p
# Compile c                                  l.
# Compile of binary_lib.v was successful.
# Compile of FP_adder.v was successful.
# Compile of FP_multiplier.v was successful.
# Compile of tb_basic_ddot.v was successful.
# 5 compiles, 0 failed with no errors.

ModelSim>
```

FIGURE B.7
Step 6: Simulating the Verilog design.



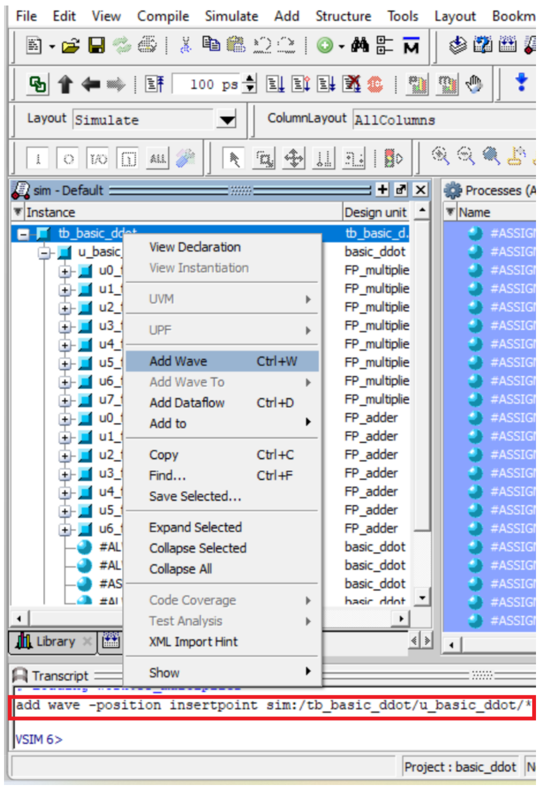
Instance	Design unit
tb_basic_ddot	tb_basic_d
u_basic_ddot	basic_ddot
u0_fp_mul	FP_multiplie
u1_fp_mul	FP_multiplie
u2_fp_mul	FP_multiplie
u3_fp_mul	FP_multiplie
u4_fp_mul	FP_multiplie
u5_fp_mul	FP_multiplie
u6_fp_mul	FP_multiplie
u7_fp_mul	FP_multiplie
u0_fp_add	FP_adder
u1_fp_add	FP_adder
u2_fp_add	FP_adder
u3_fp_add	FP_adder
u4_fp_add	FP_adder
u5_fp_add	FP_adder
u6_fp_add	FP_adder

```
# Loading work.multiplier
# Loading work.full_subber_one_output
# Loading work.twoscomplement
# Loading work.full_adder_8bit
# Loading work.FP_adder
# Loading work.full_subber
# Loading work.full_adder_24bit
# Loading work.twoscomplement_1
# Loading work.shifter
# Loading work.leadingOneDetector

VSIM 6>
```

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.



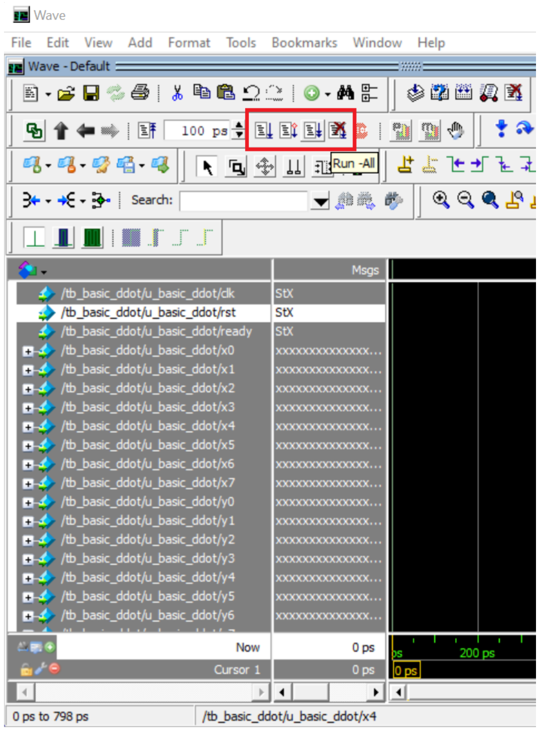
```
add wave -position insertpoint sim:/tb_basic_ddot/u_basic_ddot/*

VSIM 6>
```

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-

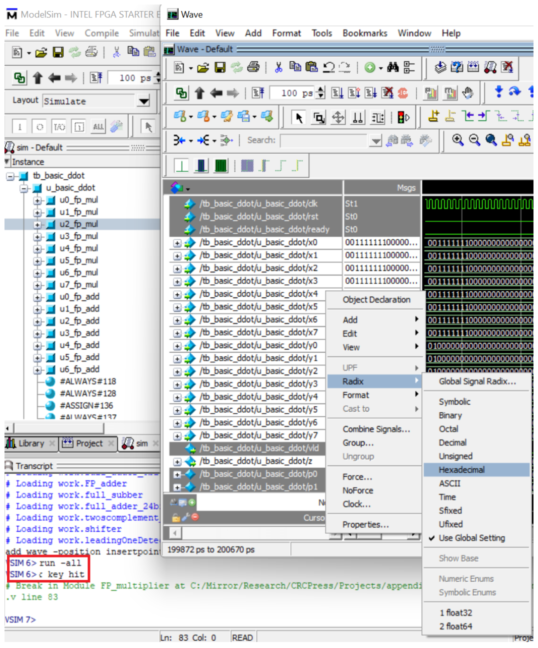


	Msgs
/tb_basic_ddot/u_basic_ddot/clk	StX
/tb_basic_ddot/u_basic_ddot/rst	StX
/tb_basic_ddot/u_basic_ddot/ready	StX
/tb_basic_ddot/u_basic_ddot/x0	xxxxxxxxxxxxxx...
/tb_basic_ddot/u_basic_ddot/x1	xxxxxxxxxxxxxx...
/tb_basic_ddot/u_basic_ddot/x2	xxxxxxxxxxxxxx...
/tb_basic_ddot/u_basic_ddot/x3	xxxxxxxxxxxxxx...
/tb_basic_ddot/u_basic_ddot/x4	xxxxxxxxxxxxxx...
/tb_basic_ddot/u_basic_ddot/x5	xxxxxxxxxxxxxx...
/tb_basic_ddot/u_basic_ddot/x6	xxxxxxxxxxxxxx...
/tb_basic_ddot/u_basic_ddot/x7	xxxxxxxxxxxxxx...
/tb_basic_ddot/u_basic_ddot/y0	xxxxxxxxxxxxxx...
/tb_basic_ddot/u_basic_ddot/y1	xxxxxxxxxxxxxx...
/tb_basic_ddot/u_basic_ddot/y2	xxxxxxxxxxxxxx...
/tb_basic_ddot/u_basic_ddot/y3	xxxxxxxxxxxxxx...
/tb_basic_ddot/u_basic_ddot/y4	xxxxxxxxxxxxxx...
/tb_basic_ddot/u_basic_ddot/y5	xxxxxxxxxxxxxx...
/tb_basic_ddot/u_basic_ddot/y6	xxxxxxxxxxxxxx...
Now	0 ps
Cursor 1	0 ps

```
0 ps to 798 ps    /tb_basic_ddot/u_basic_ddot/x4
```

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.

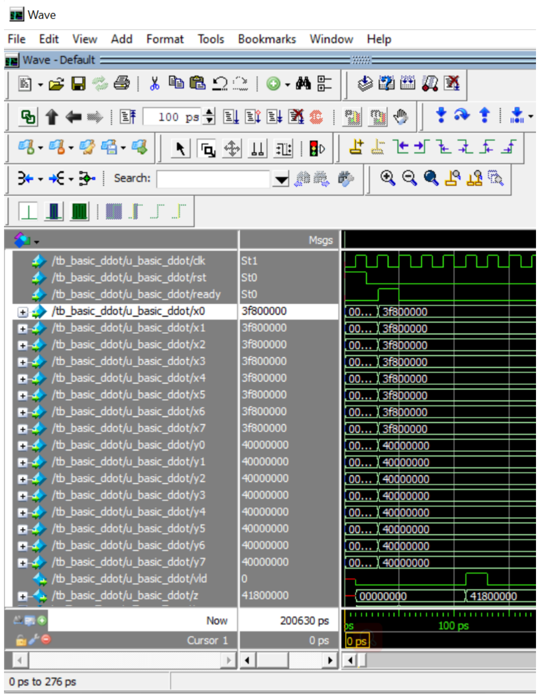


```
# Loading work.FP_adder
# Loading work.full_subber
# Loading work.full_adder_24b
# Loading work.twoscomplement
# Loading work.shifter
# Loading work.leadingOneDete
add wave -position insertpoin
VSIM 6> run -all
VSIM 6> ; key hit
# Break in Module FP_multiplier at C:/Mirror/Research/CRCPress/Projects/appendi
.v line 83

VSIM 7>
```

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 complicated 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.



	Msgs
/tb_basic_ddot/u_basic_ddot/clk	St1
/tb_basic_ddot/u_basic_ddot/rst	St0
/tb_basic_ddot/u_basic_ddot/ready	St0
/tb_basic_ddot/u_basic_ddot/x0	3f800000
/tb_basic_ddot/u_basic_ddot/x1	3f800000
/tb_basic_ddot/u_basic_ddot/x2	3f800000
/tb_basic_ddot/u_basic_ddot/x3	3f800000
/tb_basic_ddot/u_basic_ddot/x4	3f800000
/tb_basic_ddot/u_basic_ddot/x5	3f800000
/tb_basic_ddot/u_basic_ddot/x6	3f800000
/tb_basic_ddot/u_basic_ddot/x7	3f800000
/tb_basic_ddot/u_basic_ddot/y0	40000000
/tb_basic_ddot/u_basic_ddot/y1	40000000
/tb_basic_ddot/u_basic_ddot/y2	40000000
/tb_basic_ddot/u_basic_ddot/y3	40000000
/tb_basic_ddot/u_basic_ddot/y4	40000000
/tb_basic_ddot/u_basic_ddot/y5	40000000
/tb_basic_ddot/u_basic_ddot/y6	40000000
/tb_basic_ddot/u_basic_ddot/y7	40000000
/tb_basic_ddot/u_basic_ddot/vld	0
/tb_basic_ddot/u_basic_ddot/z	41800000
Now	200630 ps
Cursor 1	0 ps

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:

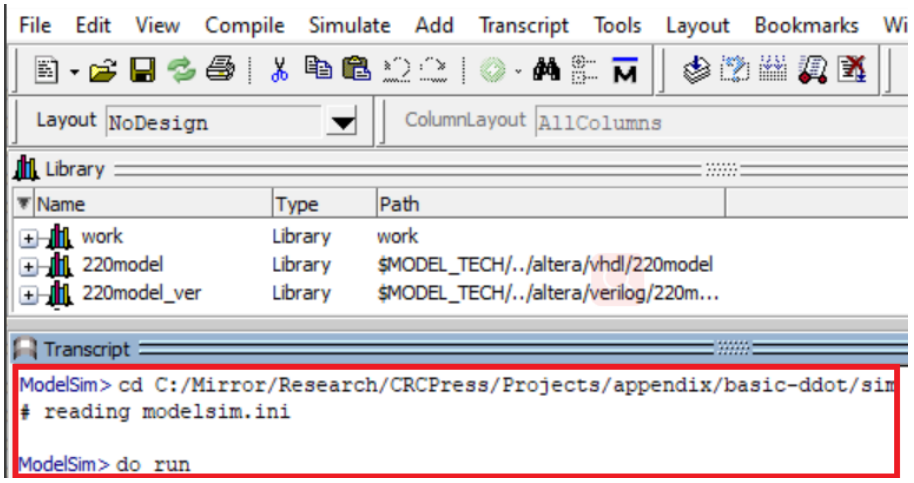
1. Step 1 : changing directory
2. Steps 2-3: creating and mapping a library (lines 1-2)
3. Steps 4-5: compiling all the files listed in the filelist.f (line 3)
4. Step 6 : starting simulation (line 4)
5. Step 7 : adding signals to the waveform (lines 5-13)
6. Step 8 : running the simulation for 1 us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


File Edit View Compile Simulate Add Transcript Tools Layout Bookmarks Wi
Layout NoDesign
ColumnLayout AllColumns
Library
Transcript

Name	Type	Path
work	Library	work
220model	Library	$MODEL_TECH/../altera/vhdl/220model
220model_ver	Library	$MODEL_TECH/../altera/verilog/220m...

```
ModelSim> cd C:/Mirror/Research/CRCPress/Projects/appendix/basic-ddot/sim
# reading modelsim.ini

ModelSim> do run
```

FIGURE B.13
Using run script: 1) change directory to **sim** folder; 2) do run