Part III

IC Design and Simulation Tutorials

A

Appendix A: Project Directory and Vim Editor

In Chapter 5, we introduce the use of Verilog to design a basic testbench, which includes a bus functional model, design-under-test instantiation, and a monitor. This tutorial focuses on the introduction of an efficient project directory, basic Unix/Linux commands for using the simulation environment, and some important commands for using the Vim editor. We use the ddot design project, which was introduced in Chapter 9, as an example for this tutorial.

A.1 Project Directory

To promote consistency across projects and development teams, it is recommended to create a standardized project directory. This allows for an organized view of all design files, simulation testbenches, scripts, synthesis results, filelists, golden files, and other relevant materials, making it easier for designers to manage tasks, programs, and processes in a single dynamic platform.

As an example, Fig. A.1 shows the project directory structure for the "ddot" design project. The top-level folder is named "basic-ddot", under which seven subfolders are created for managing different files. The **constraint** folder contains the constraint file (in this case, "Nexys-4-Master.xdc") used for the FPGA demonstration. The **dut** folder contains all the design files (".v" files), including the top module – "basic_ddot.v", the submodules – "FP_adder.v" and "FP_multipiler.v", and the "binary_lib.v" file which contains all the binary submodules for constructing FP adders and multipliers. The **filelist** folder contains the "filelist.f" file, which maintains all design files and testbenches to be compiled using the simulator. The **sim** folder contains the TCL script used for running the simulation, while the **syn** folder is used for directing all synthesis and implementation-related projects and files. The **tb** folder contains the testbench, and the **golden** folder stores all input data and golden output results.

To track and manage changes to the project, a version control system (VCS) such as Git or SVN (subversion) is typically used by IC design teams.

VCS platforms provide an efficient way to collaborate, share, and maintain the project within a development group. This is particularly useful as design projects grow in size and complexity, but even simple projects can benefit from tracking code changes with a VCS. SVN has been commonly used in IC design companies for years, while Git has recently become popular due to its use by developers collaborating on open-source projects.

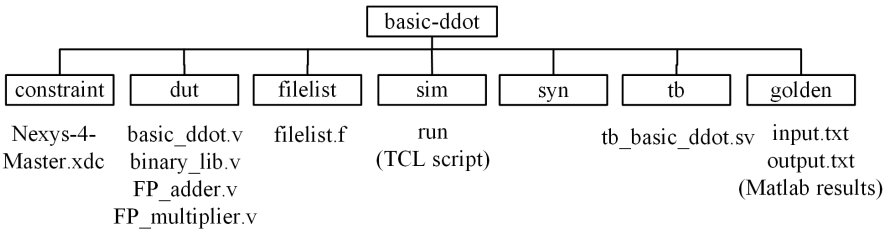

FIGURE A.1
A Typical Project Directory

A.2 Basic Unix/Linux Commands

In the field of IC design, the majority of EDA vendors such as Siemens EDA, Synopsys, and Cadence offer a comprehensive portfolio of tools for IC design, verification, and manufacturing on Unix/Linux operating systems, but not for Windows OS. In addition, the software in Windows OS typically supports Unix/Linux commands. Therefore, it is necessary for IC designers to have knowledge of basic Unix/Linux commands. Although it may seem tedious at first, learning these commands can greatly improve efficiency. This section introduces only the most basic commands as examples, including:

- . : represents the current directory.
- .. : represents the parent directory.
- cd : changes the current directory.
- pwd: prints the current working directory.
- ls : lists all the folders and files in the current directory.

Fig. A.2 shows an example of a project directory hierarchy in Unix.

```
xiaokunyang@~/basic-ddot$ ll
total 36
drwxrwxr-x 9 xiaokunyang xiaokunyang 4096 Dec 10 18:29 ./
drwxr-xr-x 5 xiaokunyang xiaokunyang 4096 Dec 10 16:13 ../
drwxrwxr-x 3 xiaokunyang xiaokunyang 4096 Dec 10 16:13 constraint/
drwxrwxr-x 2 xiaokunyang xiaokunyang 4096 Dec 10 16:48 dut/
drwxrwxr-x 2 xiaokunyang xiaokunyang 4096 Dec 10 16:13 filelist/
drwxrwxr-x 2 xiaokunyang xiaokunyang 4096 Dec 10 18:29 golden/
drwxrwxr-x 3 xiaokunyang xiaokunyang 4096 Dec 10 16:13 sim/
drwxrwxr-x 4 xiaokunyang xiaokunyang 4096 Dec 10 18:28 syn/
drwxrwxr-x 2 xiaokunyang xiaokunyang 4096 Dec 10 16:13 tb/
```

FIGURE A.2
Project Directory in Unix

A.3 Vim Editor

When building a simulation environment, various files such as scripts, Verilog codes and SystemVerilog testbenches, golden models/results, etc., need to be worked on using a text editor. Vim, one of the most popular editors in industry, is widely used for several reasons.

Firstly, Vim is an open-source text editor created in 1976. Although it is popular in the Unix/Linux world, it is also available on Mac OS and Windows OS. It is very fast and lightweight, even when editing large files and/or source code. Secondly, Vim supports several programming languages and file formats, including ".v" and ".sv" files. It can detect file extensions and recognize specific text contents and syntax. Thirdly, Vim is configurable using the ".vimrc" source file. For example, the following configuration sets the font to Consolas and the font size to 11.

```
1 set guifont=Consolas:h11
```

One of the most significant advantages of using Vim is the ability to use many useful commands to increase programming efficiency. Although it has a bit of a learning curve, once you learn the commands, you will find it helpful and productive to use the editor. Many Vim commands are available, but in this book, only the most frequently used ones are introduced below.

Two Modes Using Vim

Vim has two modes: command mode and insert mode. In the command mode, users can execute various commands such as navigating the file, copying, changing, or deleting code, and more. In the insert mode, users can type text into the file. To switch from the command mode to insert mode, users

need to type "i". To switch from insert mode back to command mode, users can type "Esc" or the "escape key". Below are the steps for writing and saving code with Vim:

- Step 1: Open or create a file with a chosen filename.
- Step 2: Type "i" to switch to the insert mode.
- Step 3: Enter your code in the file.
- Step 4: Hit "Esc" key to switch back to the command mode.
- Step 5: Type ":w!" to write the file, or "wq!" to write the file and exit Vim.

The write and exit commands also have the following variations:

- :w : Write changes to the original file.
- :w! : Force write changes (no warning is given).
- :wq : Write changes to the file and quit the editor.
- :q : Quit the editor (a warning is printed if a modified file has not been saved).
- :q! : Quit the editor without saving any changes (no warning is given).

Navigating within a File

Vim's highly effective key-bindings allow you to write code without lifting your fingers from the keyboard. Instead of using arrow keys to navigate within a file, Vim provides the "h", "l", "k", and "j" keys in the command mode for navigation. Here's what each key does:

- h: Moves the cursor one character to the left.
- l: Moves the cursor one character to the right.
- k: Moves the cursor up one line.
- j: Moves the cursor down one line.

While moving fingers on the keyboard and using a mouse may only take seconds, using key-bindings can save hours and keep your fingers in the center of the keyboard. This can help keep programmers in flow-states and ultimately help them focus on writing code. Additionally, there are several additional keyboard shortcuts to navigate within a file:

- :set nu : Shows the number of lines in the current file.
- :N : Move cursor to the specified line (for example, :10 goes to line 10).

- 0 : Move cursor to the beginning of the current line.
- Shift+$: Move cursor to the end of the current line.
- 1 followed by Shift+G: Jump to the first line in the file.
- Shift+G : Jump to the last line in the file.

Commands for Copy, Paste and Delete

Here are some useful commands for copying, pasting, and deleting text in Vim. Each command can be repeated a certain number of times by typing a number before the command, and using the "." command can repeat the previous action. Additionally, the "u" command is essential for undoing changes made to the text.

- x : Deletes the character where the cursor is.
- dw : Deletes the word from where the cursor is.
- dd : Deletes the line where the cursor is.
- Ndd: Deletes N lines from where the cursor is.
- yy : Copies the current line where the cursor is.
- Nyy: Copies N lines starting from the current line.
- p : Pastes the copied or deleted text below the current line where the cursor is.
- u : Undoes the last change made to the text.

By knowing these commands, you can easily manipulate text in Vim for increased efficiency.

Visual Block: Editing Text by Block/Row/Column

One of the main benefits of using Vim is the ability to edit text in block/row/-column in the visual block. In Unix/Linux, typing "Ctrl-v" enters block-wise visual block mode to highlight rows and columns. In Windows, the command is "Ctrl-q".

Once in visual block mode, the keyboard navigation keys "h-j-k-l" can be used to select and highlight text by blocks. Then, batch commands can be used to manipulate the selected text, such as delete ("d"), yank ("y"), or paste ("p"). To exit visual block mode, press "Esc".

For example, Fig.A.3 shows an example of copying and pasting a visual block (all the IO ports of the design) into parentheses for the design instantiation and connection. First, enter visual block mode by typing "Ctrl-v" (Unix/Linux) or "Ctrl-q" (Windows) at the top left corner of the visual block.

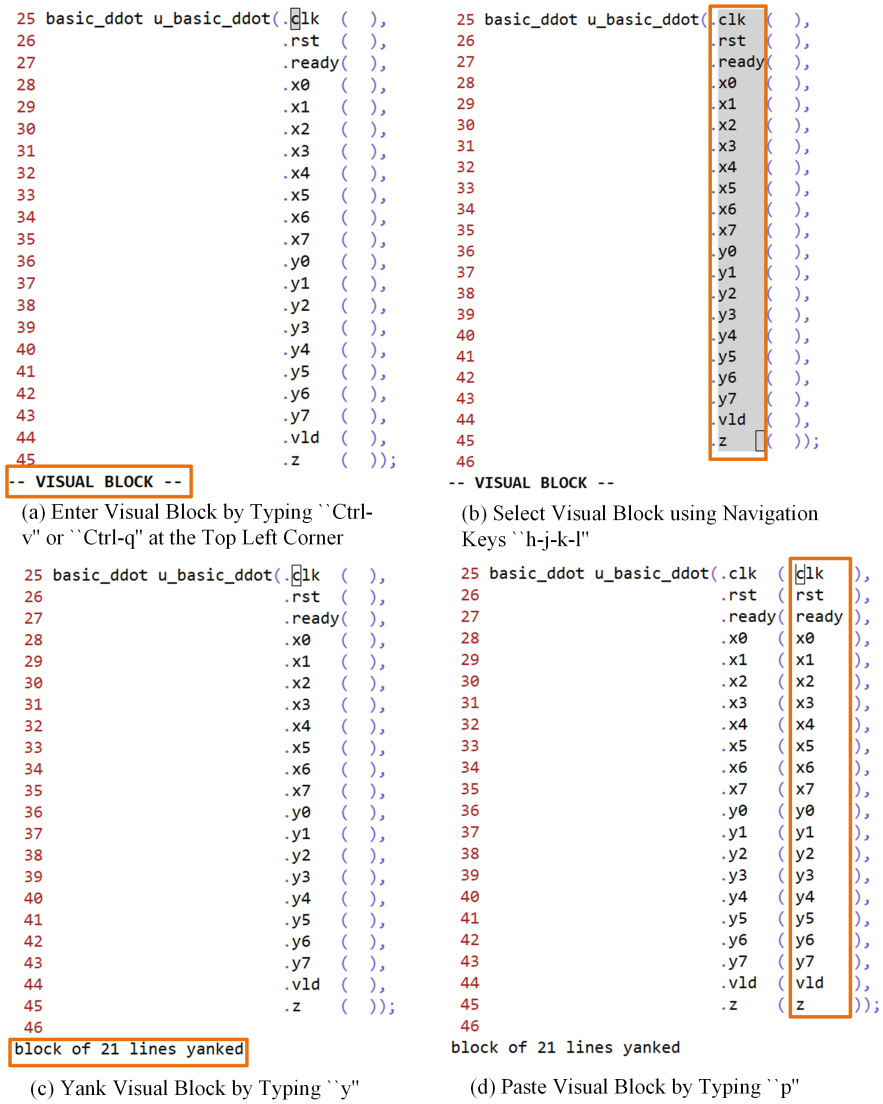

FIGURE A.3
Copy and Paste a Visual Block (all the IO ports of the design) into the Parenthesis (IO ports connection for the design instantiation and connection)

Fig.A.3(a) shows that the first letter "c" of the "clk" signal is selected with a gray background. The Vim editor also displays "VISUAL BLOCK" in the bottom left corner of the screen. Next, select the visual block by moving the navigation keys. As shown in Fig.A.3(b), all the ports in lines 25-45 are selected as the visual block with a gray background. To copy the selected visual block, type "y", and Vim will display "block of 21 lines yanked" in the bottom left corner, as shown in Fig.A.3(c). After the visual block is copied, move the cursor to the top left position of the block where the copy will be pasted. As shown in Fig. A.3(d), the cursor is moved to line 25 and into the parentheses, which is the top left corner of the block. Finally, type "p" to paste the copied visual block into the parentheses in lines 25-45.

The visual block feature in Vim is not limited to copying and pasting. It can be used with any other Vim command. Therefore, vertical alignment for writing Verilog is strongly recommended as it can make programming very productive and improve the readability of the code.