Concise Manual for the Modelsim/Questasim VHDL Simulator

Sabih H. Gerez
University of Twente, Department of Electrical Engineering
s.h.gerez@utwente.nl

Version 9* (August 29, 2022)

This document provides a minimal set of instructions to work with the *Questasim*[1] VHDL simulator produced by *Model Technology*. This is a powerful commercial simulator that can handle both the VHDL and Verilog hardware description languages. The manual is based on software version *Release 2020.10* for Linux (it displays `Version 2020.4 linux_x86_64 Oct 13 2020` in the *Transcript Window* after starting up). The information should be sufficient for the lab exercises related to the course *System-on-Chip Design*.[2]

Contents

1 Preparations **2**

2 Projects **3**

3 Libraries **4**

4 Compiling VHDL Files **4**

5 Preparing for Simulation **5**

6 Selecting the Signals to Be Traced **5**

7 The Actual Simulation and Results Analysis **7**

*Version history: Version 1 was released in 2003, Version 2 in 2004, Version 3 in 2006, Version 4 in 2007, Version 5 in 2011, Version 6 in 2013, Version 7 in 2015, and Version 8 in 2019.

[1] *Questasim* is an extended version of *Modelsim*.

[2] See also: `http://sabihgerez.com/ut/soc/`.

8 Gate-Level Simulations **8**

9 Termination, Restarting and Loading New Designs **9**

10 Printing and Editing VHDL **10**

11 Troubleshooting **10**

11.1 Corrupt Local Library 10

11.2 Small Fonts in Built-in Editor 10

1 Preparations

Unless stated otherwise, the left mouse button should be used for all mouse actions.

It is wise to keep all VHDL files belonging to one exercise in one directory. Go to that particular directory and type `vsim` in the Unix shell (command line).

The graphical user interface of Questasim consists of a "main" *Questasim* window that hosts several of subwindows, called *panes*. A pane can be manipulated by clicking on the controls at its right:

- One can *zoom* (maximize) a pane using the "plus" symbol and restore its original size using the "minus" symbol.
- One can *undock* a pane using the symbol consisting of a square with an arrow pointing outward. This will separate the pane from the main window and make it a window of its own in the host operating system. The reverse operation of *docking* is also possible using a button consisting of a square and an arrow pointing inward.
- One can *hide* (remove) a pane by using the "cross" symbol. To restore a hidden pane, use the view menu. Suppose that you have hidden the *objects* pane, then:

  View → Objects

  or its command-line equivalent:

  ```
  VSIM> view objects
  ```

Questasim offers a lot of freedom to rearrange the relative positions of the panes by making it possible to drag the panes around grabbing them halfway their top bar. It is not recommended to change the initial rearrangement, though, as this makes it harder for an assistant to follow what you are doing.

You can restore the default layout of panes with:

Layout → Reset

The command prompt in the *transcript* pane changes from the `Questasim>` prompt into the `VSIM>` prompt when a design has been loaded for simulation indicating that a simulation can be started. Note: the `VSIM>` prompt contains a sequence number that is incremented after processing a command; this number is omitted in the rest of this manual.

2 Projects

Questasim's mechanism to keep all source files of a design together is called a *project*. Although you can compile and simulate outside projects, it is mandatory that you make use of the project mechanism for all exercises in the *System-on-Chip Design* course.

The use of projects has the major advantage that Questasim keeps track of the order in which files have been compiled as well as the specific compiler options set for that file. It can recompile all source files in the right order and with the right options in case of problems (which saves quite some time compared to manual compilation).

Create a project using the command:

File → New → Project ...

Provide an appropriate project name and leave all other information as is. Then, a window will pop up asking you to add files to the project. If you know what you are doing, you can proceed to add files. Otherwise, you can close this window and add the files later on (see Section 4).

In future sessions, if your project is not yet open at start up, open it using (projects are stored in a text file with extension *mpf* on the file system):

File → Open ...

or:

File → Recent Projects → ...

To recompile all files, use:

Compile → Compile All

If necessary, you can change the compilation order by:

Compile → Compile Order...

Select a file by clicking on it and then use the up and down arrow buttons to modify the order.

You also have the option of asking Questasim to find out the compile order by pushing Auto Generate. *Beware:* the order generated by Questasim may not be correct, even when it reports that automatic ordering was successful.

3 Libraries

VHDL stores all compiled code in a *library*. Unless you explicitly require the use of another library, your files will be compiled into the library `work`. Questasim creates this library automatically when you create a project (see Section 2). You can verify this by opening the *library* pane. It lists all libraries that are available.

You will not need to create new libraries for the exercises. If it is necessary for some unforseen reason, use:

File → New → Library ...

In the window that pops up, you can type a library name: `work` or any other name that is appropriate. The *library name* and the *directory name* in the file system to which it is mapped, are kept equal by default. Do not change this. You are now ready for compiling VHDL files, also in future invocations of Questasim. You should not create new libraries unless you are explicitly asked to do so.

Note: during exercises you will also make use of other libraries, such as `ieee`. They are located in system directories to be accessed by all users. The global Questasim configuration should allow you to access them without problems.

4 Compiling VHDL Files

Make first sure that the *Project* pane is selected. The source files belonging to your project are visible in this tab. If you do not see it, execute:

View → Project

When you want to compile a file, you should first add it to your project using:

Project → Add to Project → Existing File ...

This command is also available under the right mouse button when your cursor is in the left part of the main window and the *Project* tab is active. After the file has been successfully added, it will show up in the list of files in the left part of the main window (*Project* tab).

Compiling a VHDL file can be achieved by selecting the file in the *Project* tab and issuing the command:

Compile → Compile Selected

The command is also available under the right mouse button.

Note: if the file is to be compiled in another library than `work` (not necessary in the System-on-Chip Design course), change the library by selecting the *Properties* option under the right mouse button.

The *Project* pane has a *Status* column for each file that makes use of small icons. A "question mark" indicates that the file has not yet been compiled since the last time it was edited. A "green check" is

used for successful compilation while a "red cross" is the result of failures during compilation. A short summary of the compilation process is also printed in the *Transcript* pane. In case of failure, one can select a file and issue:

Compile → Compile Report ...

to see what went wrong (the command is also available through the right mouse button).

In VHDL, it is important to compile (and recompile) files in the right order. Files containing definitions should be compiled before files that make use of these definitions. Respect this order when adding files to your project or correct this order (see also Section 2).

Warning messages of the type "No default binding ..." issued by the compiler can often be ignored. They are related to the fact that the compiler has more than a single choice for an architecture or configuration of an entity. The reason that it can be ignored is that a configuration declaration to be compiled at a later moment specifies the choice to be made.

5 Preparing for Simulation

Although Questasim supports the simulation of VHDL entity-architecture combinations, the neat way of specifying what is to be simulated is by means of *configurations*. Execute:

Simulate → Start Simulation ...

A multitab window will pop up. The *Design* tab shows all libraries. By manipulating the plus and minus signs like in a file browser, one unveils or hides the design units compiled in a library.

Select the configuration that you want to simulate. It will most likely be in library `work`. Click on OK (for gate-level simulations, you will need to do something more, see Section 8). You will be positioned in a new pane, called *sim*. This pane contains a browser to navigate through the structure of the design and the packages used.

Questasim makes use of many panes/windows. The windows that are necessary for the exercises, are *Objects* (for seeing the signals belonging to the selected component) and *Wave* (for visualizing signals changing over time). If you don't already see them, open these using:

View → Objects
View → Wave

6 Selecting the Signals to Be Traced

The main goal of a simulation is to verify a design. The simulation results in signal transitions through time. For the purposes of the exercises, it is often sufficient to perform the verification by displaying the transitions along a time scale and zooming in at specific parts of the *waveforms* displayed. Waveforms are displayed in the *Wave* window.

To get signals displayed, you should first choose the design unit in which the signal occurs in the *sim* tab of the main window, which behaves like a browser in which you can navigate through the hierarchy of your design. A design unit is simply selected by clicking on it. Parts of the hierarchy can be hidden or displayed by clicking on the little squares containing a '-' or a '+' respectively. All signals of the selected design unit are listed in the *Objects* window. Using the *Show* command under the right mouse button, one can hide/unhide *inputs*, *outputs*, *internal signals*, etc.

A single signal is selected by clicking on it in the *Objects* window. A range of signals is selected by selecting the first and then pressing the Shift button on the keyboard in combination with the left mouse button after positioning the cursor on the last signal in the range. Adding signals to a given selection can be done using the Ctrl button on the keyboard while clicking with the left mouse button.

Individual components of composite signals (e.g. of type array) can be displayed by clicking on the little squares at the left of the signal name. This allows e.g. the selection of individual parts for display in the *Wave* window.

Selected signals are transferred to the *Wave* window with the command:

Add: To Wave → Selected signals

If you are sure that you want to trace all signals of the currently selected design unit, use:

Add: To Wave → Wave → Signals in region

In the *SoC Design* course, you are supposed to be critical on which signals to display. Simply displaying all signals may affect your mark.

The above top-menu commands have equivalents that can be controlled with the right mouse button. Yet an alternative way of transferring selected signals to the *Wave* window is to use *drag & drop*. Select first the signals as described above and put the mouse cursor in the *Objects* window. Then click on the left mouse button and drag the selected objects to the *Wave* window without releasing the button. Release the button when the cursor has arrived at the desired location. This *drag & drop* mechanism also works within *Wave* itself e.g. to reorder the signals.

Once all signals to be traced have been notified to the *Wave* window, the actual simulation can start. You may, however, want to modify the display properties of the signals. If you want e.g. to display a binary vector as a decimal number, you should first select the signal in the *Wave* window and then execute:

Wave → Format → Radix → Decimal

The radix can also be chosen via the right mouse button. In most cases, you will not need to modify the default radix of a signal.

Sometimes, it is interesting to display a sequence of digital values as a continuous waveform. The easiest way to achieve this is by issuing the command:

Wave → Format → Format → Analog (automatic)

Signal names will be displayed with their full instantiation path. To reduce the length displayed execute:

Tools → Window Preferences...

in the undocked *Wave* pane or

Wave → Wave Preferences...

in the docked *Wave* pane.

Fill a small integer number unequal to zero in the field "Display signal path".

When displaying a larger number of signals in the *Wave* pane, it is a good idea to structure the presentation of the signals by inserting one or more *dividers*. To insert a divider, execute:

Add → Divider...

in the undocked *Wave* pane or

Add → To Wave → Divider...

in the docked *Wave* pane.

If you use dividers, choose a meaning label for the divider by modifying its *properties* accessible via the right mouse button.

7 The Actual Simulation and Results Analysis

Because in all exercises the control of all simulation is done by VHDL itself, performing a simulation only requires instructing VSIM to run for an indefinite length of time. The simulation will stop after an assertion in VHDL. Running for an indefinite length of time is accomplished by:

Simulate → Run → Run -All

If nothing goes wrong during the simulation, the simulation will terminate at the desired moment and waveforms of the selected signals will be displayed in the *Wave* window. In addition, a new window, the *Source* window, will show up. This window shows the VHDL code belonging to the design unit being simulated. It not only displays the source code, but can serve as a text editor as well (see Section 10).

Once the simulation has stopped, the waveforms produced will be displayed in the *Wave* window. It is wise to first look at the full time interval of the simulation by:

Wave → Zoom → Zoom Full

(a keyboard shortcut 'f' has the same effect).

In almost any simulation, it will be necessary to zoom in at specific part of the time scale to inspect specific details. The other options of the Zoom pull-down menu, that are quite self-explanatory, can be

used for this purpose; they are also available under the right mouse button in the *Wave* window. Another possibility, which can be advised, is to drag with the middle mouse button (or Ctrl + left mouse button) to manually select a specific range.

In some situations, it is necessary to derive accurate timing data from the waveforms. The *time cursors* available in the *Wave* window are very useful in this respect. These are vertical lines that can be moved through the waveforms and should not be confused with other types of cursors that e.g. follow the mouse movement on the screen. The *Wave* window has one time cursor by default; it can be dragged by clicking (and not releasing) the left mouse button in the waveform window (this is the right part of the *Wave* window). The time cursor will *snap* on signal transitions on which the *mouse cursor* is located. One can also put a time cursor on a transition by pushing the *Find Next Transition* and *Find Previous Transition* icons on the toolbar.

A useful feature is the availability of multiple time cursors. To create a new cursor, execute:

Add → To Wave → Cursor

in the docked *Wave* pane or

Add → Cursor

in the undocked *Wave* pane.

When multiple time cursors are present, the time difference between each pair of cursors will automatically be displayed in the *Wave* window. Together with the possibility to snap on signal transitions, this gives the possibility of accurate timing measurements.

Once you have selected a suitable range in which a specific behavior becomes visible, you might want to have a hardcopy of the waveforms or save the waveforms in a graphic file format. Use:

File → Print PostScript ...

for this purpose. In the window that pops up, select the option to print to a file (not to a printer). You can then also set some properties of the page, especially which signals to plot and the time range. Click on Setup... to display a *Page Setup* window in which you can configure the layout of the page to be printed. Pay special attention to choosing *A4* paper size and a *Scaling* that fits to 1 page. Close the windows by clicking on OK. You can convert PostScript to PDF with Linux command `ps2pdf -sPAPERSIZE=a4`. You can view PDF documents with the program `evince`.

Instead of using the print command, you can, of course, also make sure that the waveforms display well on your screen and then make a screenshot.

8 Gate-Level Simulations

The procedure for performing a gate-level simulation, starts in the same way as described in Section 5. However, after selecting a configuration, one should specify the SDF file to be used. SDF stands for *Standard Delay Format*, a standardized text-file format that specifies delays in interconnections and gates. An SDF file can be generated by the Synopsys Design Compiler (this is the case for your

exercises). In practice, more accurate delay data are available after layout design with placement and routing; then, the SDF file is generated by the layout tools.

In order to link one or more SDF files to your design, select the *SDF* tab in the the multitab window that pops up after:

Simulate → Start Simulation ...

You should, of course, select first the configuration to be simulated in the *Design* tab. Click on Add in the *SDF* tab. A new window pops up. Type the name of the SDF file to be used or select it by browsing. Note that the SDF file contains information on a synthesized block of hardware. The simulation model normally instantiates this block in a larger context, e.g. as a subblock of a testbench. In the second line indicated by *Apply to region*, you need to indicate the part of the design for which the SDF is meant. This is indicated by a *directory style* path. The path "`/`" is the *root* or top-level design. Subdesigns are selected by the instance names as used in VHDL structural descriptions (in the `siso_gen` example, the hardware has instance name `duv` and is part of the testbench instance `tg`; therefore the path to be used for SDF is `/tg/duv`).

Once the configuration and SDF has been correctly specified click on OK twice. The *Transcript* pane should state "SDF Backannotation Successfully Completed". From here on, the simulation can be carried out using the same approach as already described above.

9 Termination, Restarting and Loading New Designs

If you want to terminate the *Questasim* session, you simply give the command:

File → Quit

and confirm your wish in the window that pops up. It may, however, be wise to keep the set of signals selected for the *Wave* window for later use, rather than to have to repeat the selection procedure in a future simulation. The signals can be saved with:

File → Save Format ...

A window will pop up in which you have the possibility of changing the default file name `wave.do`.

If you had previously saved a set of signals to be traced in e.g. a file `wave.do`, in the undocked window, you can restore the settings using:

File → Load ...

and entering the file name in the window that pops up. In the docked window:

File → Load → Macro File ...

An alternative way is to use the command-line interface of the *Questasim* window:

```
VSIM> do wave.do
```

Suppose that you have modified and recompiled the VHDL source files related to the entity or configuration that you have just been simulating or have added new signals to the *Wave* window, you can repeat your simulation using:

Simulate → Restart ...

This will erase the waveforms but keep the selected signals in the *Wave* window. You can now rerun the simulation and analyze the results as explained in Section 7.

You may also wish to run a simulation on a different configuration than the one you were just simulating. You can simply execute:

Simulate → Start Simulation ...

10 Printing and Editing VHDL

In the environment set up for the *System-on-Chip Design* course, you can use the command `vhd2pdf` to generate PDF files of VHDL source files. At the Linux shell prompt you type `vhd2pdf` followed by one or more file names with extension `.vhd` located in the current directory.

In order to use the built-in editor of Questasim, select the VHDL file to be edited in the *Project* tab and use the *Edit* option of the pop-up menu that is invoked by the right mouse button. It has the advantage of highlighting VHDL keywords. You can select a different font type or size with:

Tools → Edit Preferences → Source Windows

The public domain editor `vim` and its GUI variant `gvim` with syntax highlighting are suitable alternatives if you are familiar with the editors. Yet other alternatives are: `xed` and `gedit`.

11 Troubleshooting

11.1 Corrupt Local Library

If you do not manage to compile or simulate your design while you can't find a direct explanation, it may help to remove the `work` directory with the Linux command:

```
rm -rf work
```

followed by recompiling all files in your project.

11.2 Small Fonts in Built-in Editor

It may happen that the source-file viewer/editor shows extremely small fonts. This is likely to be fixed by:

Tools → Edit Preferences

followed by a click on Restore default fonts.