An ECE241 ModelSim Primer

Stuart Byma

1 Introduction

ModelSim is an industry standard HDL simulation tool. It is powerful, and can simulate large systems quickly and easily, but has a much higher learning curve than QSim. This tutorial will give you a brief introduction to ModelSim and how to use it.

2 Getting Started

You should have in the same folder as this file several Verilog and other files:

- accserial.v
- accserial_tb.v
- wave.do

The design we wish to simulate is in accserial.v. We will use the Verilog testbench accserial_tb.v to instantiate the design, provide input, and drive the simulation.

2.1 The Testbench File

The testbench file will contain some Verilog that might look strange to you – this is because it is meant to drive a simulation, and not to be synthesized by a tool like Quartus II. As you may know, creating waveform files for simulation input can be arduous and time consuming. A testbench file allows us to write Verilog to drive the simulation, and also enables us to automate data collection and verification of the system we are testing.

The new thing you may see in the testbench is the Verilog `initial` statement. An initial block is executed only once at time zero (i.e. the very beginning of the simulation). If there are multiple `initial` statements, they are executed in order. Delay statements, like `# 10`, are used to delay the simulation – this allows us to set different input values at different times during the simulation.

REMEMBER: The testbench file is NOT synthesized to any real hardware, it is strictly for simulation. `initial` and time delay `#` are generally not used in synthesized hardware description. Occasionally we do need to initialize things in our design under test – in accserial.v we need to initialize a memory – and here we may use the `initial` statement. But we explicitly tell the Quartus synthesizer to ignore it, using a directive like this:

```
/* synthesis translate_off */
initial begin ... end
/* synthesis translate_on */
```

The Quartus II synthesis engine can synthesize some `initial` blocks, but I recommend you only use them for simulation at this point.

Have a close look at accserial_tb.v – it has many comments to help you understand how it works, and what everything means.

2.2 Simulating with ModelSim

Let's get started. First open ModelSim. Then, in the ModelSim command line, navigate to the directory containing the Verilog files.

```
> cd path/to/this/folder
```

We will mostly use the command line in this tutorial. It may be a little less intuitive, but it's great to learn because it can be much more efficient, and also allows us to easily make scripts to automate our workflow later.

First we need to create a working library for our simulation:

```
> vlib work
```

Next, we compile our Verilog files so that ModelSim can simulate them.

```
> vlog -novopt *.v
```

Note the flag `-novopt` – this disables optimization, so we can see all of the internal signals. `*.v` indicates all Verilog files in the present directory. Now the fun begins. We simulate the testbench file from the work library – remember, the testbench file instantiates the design we want to test.

```
> vsim work.accserial_tb
```

If the waveform viewer is not open, in the GUI click View → Wave. Run the command

```
> do wave.do
```

This is a script that will add signals to the waveform viewer. Alternatively, you can select instances from the Instance panel, and drag signals from the Object window into the waveform viewer. You will notice that the simulation has not actually been run yet. To run the simulation, type:

```
> run -all
```

This will run the simulation, until the simulation stops at the `$stop` command in the testbench file. ModelSim may open the testbench file to show you where the break occurred – you can ignore this and return to the waveform viewer panel. Now you should see the simulation results in the waveform viewer. You may need to zoom out a bit to get a better view (there are GUI buttons for this).

Now it's up to you to see if the circuit is working correctly!

Note that at this point the simulation is paused. To run the simulation `value` time units longer, type:

```
> run -step <value>
```

If you made changes to any of the Verilog files, restart the simulation, recompile the Verilog files, and then run the simulation again:

```
> restart -f
> vlog -novopt *.v
> run -all
```

3 Conclusion

Hopefully that will get you started – happy simulating!