CME 341: Introduction to Modelsim and Testbenches

Brian Berscheid

Department of Electrical and Computer Engineering
University of Saskatchewan

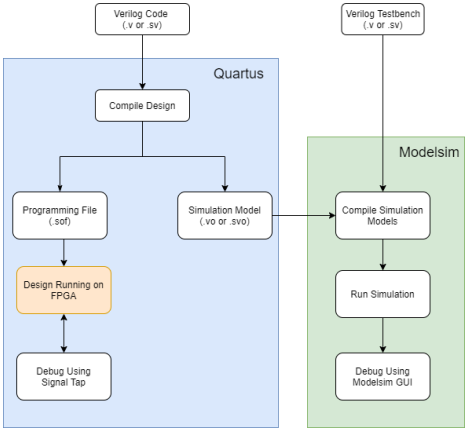
Today's agenda

1. Introduction to Simulation
2. Introduction to Testbenches
3. Modelsim Demonstrations

Introduction to Simulation

Methods of Debugging FPGA Designs

- Two main methods of debugging FPGA designs:
  - ▷ Run on FPGA and observe behavior. Probe internal signals with virtual logic analyzer (Signal Tap).
  - ▷ Use an HDL simulator to predict the behavior of an FPGA with the HDL design.
- Both are valuable (and will be used in CME 341), but simulation is generally preferred:
  - ▷ Often faster and more flexible
  - ▷ Easier to automate
  - ▷ Doesn't require hardware



Types of Simulation

- Two types of simulations are possible:
  - ▷ Functional simulation: Idealized simulation using original verilog code. Simulator has no knowledge of place and route results → unable to model propagation delays inside FPGA
  - ▷ Timing simulation: Accurate model of FPGA; uses timing model from compilation process

- Functional simulations are much faster to run, but can give incorrect results if tests are not designed carefully.

- CME 341 will focus on **timing simulations**; functional simulation will be studied in future classes

Requirements for a Timing Simulation

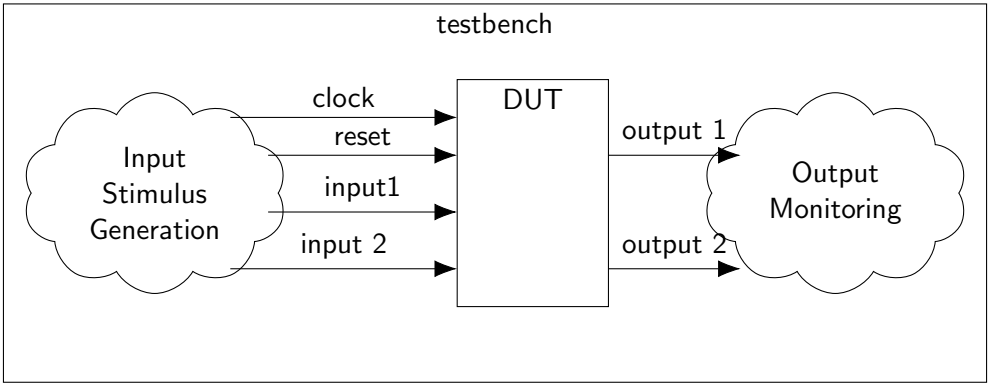
- To run a timing simulation, the following items are needed:
  - ▷ A simulator (software package)
    - We will use Modelsim in CME 341, but many others exist
    - Specifically, we will use Modelsim-Altera (provided with Quartus)
  - ▷ FPGA simulation libraries - model the basic building blocks of the FPGA being used
    - The required libraries are built into Modelsim-Altera
  - ▷ The timing simulation model (.vo file) for the design to be tested
    - Output by Quartus after a successful compile
  - ▷ A Verilog testbench (.sv file)
    - Discussed in detail in the next section

Introduction to Testbenches

What is a testbench?

- A testbench is a special Verilog module that is used to test an HDL design.
- The testbench is not passed through Quartus and does not run inside the FPGA. It is solely used by the simulator.
- The key responsibilities of a testbench are:
  - ▷ Instantiates the design to be tested (DUT - design under test)
  - ▷ Generates the necessary stimulus signals to test the design
  - ▷ May automatically monitor the design's performance against expectations
- Testbench design is a "sub-specialty" within the field of FPGA / ASIC design. It is a very complex skill in high demand in industry.
- We will only cover the very basics in CME 341. CME students will study testbench design in detail in CME 435.

Testbench block diagram



Testbenches: Basics to remember

- Module is generally named based on the design that is being tested
  - ▷ Ex. A testbench to test a module called "aaa" would typically be named "aaa_testbench" or "aaa_tb"
- Testbenches normally have NO inputs and NO outputs.
- Testbenches have internal signals (wires or registers) corresponding to each input and output port of the DUT.
- Since testbenches never run inside the FPGA, they can use certain features of Verilog that don't map to hardware.
  - ▷ The distinction is often referred to as "simulation code" vs "synthesizable code".
  - ▷ The DUT must use synthesizable code which maps to hardware and can be compiled by Quartus.
  - ▷ Testbenches are not bound by this restriction.

Testbenches: Advice for beginners

- The most difficult part of writing a testbench is figuring out what are the critical test scenarios that must be exercised in order to verify correct operation of the DUT.

- For all but the smallest designs, it is not possible to exhaustively test all possible combinations of inputs / signal timings.
  - ▷ Think carefully about the worst-case / most stressful combinations of input signals for the DUT. Try to make sure your testbench covers all of these.
  - ▷ Proper testbench design requires a solid understanding of the design requirements.

- Try to set your DUT code aside and not think about it when writing your test scenarios.

Example Testbench

```
‘timescale 1 ns / 1 ps // compiler directive that must be included
                       // the first time listed is used for the unit
                       // of time on waveform plots
                       // the second time listed indicates precision.
                       // all delays are rounded to the ‘‘precision time’’.
                       // allowable units of time are ms, us, ns, and ps
                       // allowable numbers are 1, 10, 100
module set_clear_latch_testbench(); // no inputs or outputs

reg n_set, n_clear;  // registers for DUT inputs
wire Q, n_Q;         // wires for DUT outputs

initial
#150 $stop;  // stop the simulation at 150 ns
```

Example Testbench (continued... generating n_set)

```
initial
begin
#10 n_set = 1'b0;   // schedule n_set to be 0 at t=10 ns
#10 n_set = 1'b1;   // schedule n_set to be 1 at t=10+10=20 ns
#55 n_set = 1'b1;   // schedule n_set to be 1 at t=20+55=75 ns
end

initial
#65 n_set = 1'b0;   // schedule n_set to be 0 at t=65 ns
```

Example Testbench (continued... generating n_clear)

```
initial
begin
n_clear = 1'b1;        // schedule n_clear to be 1 at t=0
#50 n_clear = 1'b0;    // schedule n_clear to be 0 at t=50 ns
#5 n_clear = 1'b1;    // schedule n_clear to be 1 at t=50+5=55 ns
#10 n_clear = 1'b0;    // schedule n_clear to be 0 at t=55+10=65 ns
#15 n_clear = 1'b1;    // schedule n_clear to be 0 at t=65+15=80 ns
end
```

Example Testbench (continued... instantiating DUT)

```
// instantiate the prototype for the set clear latch
set_clear_latch latch_1(.n_set(n_set),
.n_clear(n_clear),
.Q(Q)
);

endmodule
```

Comments on example testbench

- Verilog is a parallel language; all of the procedures starting with `initial` run in parallel.
- The #XX notation tells the simulator to wait for XX time units before proceeding.
  - ▷ Note that this IS NOT a synthesizable construct. It can be used in testbenches only.
- Exercise: Try to sketch out the waveforms for the n_set and n_clear. Then sketch the output waveforms you would expect the set-clear latch to produce. Does this testbench do a good job of testing the latch?

Modelsim Demonstrations

Thank you!
Have a great day!