Modelsim Without the Gui

I have found the Modelsim gui to be very slow and frustrating. So, instead of using it, I prefer to use the command-line version of Modelsim, and "printf-style" debugging. Here's how to do this.

Preparation

First, put all of your `.v` files in a single directory. Go to that directory and type:

```
vlib work
```

Running

Now, type this magic commands:

```
vlog *.v && vsim -c -do 'run 1000ns;quit' TopLevelModule
```

Where `TopLevelModule` is the name of the top level module (ie your testbench). Remember, if your top level module is called `Foo`, it should be in a file called `Foo.v`.

If, for example, you used this as your top-level module,

```
module TestVerilog;
  initial begin
     $display("hello, world!");
  end
endmodule
```

You should see something roughly like this:

```
Model Technology ModelSim SE vlog 6.3a Compiler 2007.06 Jun 25 2007
-- Compiling module TestVerilog

Top level modules:
        TestVerilog
Reading /usr/local/ModelSim/6.3a/modeltech/tcl/vsim/pref.tcl

# 6.3a

# vsim -c -quiet TestVerilog
# //  ModelSim SE 6.3a Jun 25 2007 Linux 2.6.22.7-57.fc6
# //
# //  Copyright 1991-2007 Mentor Graphics Corporation
# //              All Rights Reserved.
# //
# //  THIS WORK CONTAINS TRADE SECRET AND
# //  PROPRIETARY INFORMATION WHICH IS THE PROPERTY
# //  OF MENTOR GRAPHICS CORPORATION OR ITS LICENSORS
# //  AND IS SUBJECT TO LICENSE TERMS.
# //
VSIM 1> run 100ns;quit
```

```
# hello, world!
```

Debugging

In order to debug our code, we're going to break the rules just a tiny bit.

 Be sure you undo these changes before submitting any work, or you will lose lots of points!

We're going to insert `$display` statements in our hardware code (ie outside the testbench), where it is normally prohibited. However, in order to ensure that our `$display` statements show us what "real hardware" would do, we must stick to a very important rule:

You may add a `$display` statement only if it is inside an `always @(posedge clk)` block, and only if that block appears **before any other block or instantiation in the module**.

Here is an example:

```
module TestVerilog;
   reg clk;
   initial clk = 0;
   always #(5) clk = ~clk;
   SubModule submod(clk);
endmodule

module SubModule(clk);
   input clk;
   wire  [3:0] acc_in;
   wire  [3:0] acc_out;

   always @(posedge clk) begin
      $display(acc_in, acc_out);
   end

   assign acc_in = acc_out + 1;
   Reg    myreg(.clk(clk), .out(acc_out), .in(acc_in));

endmodule
```

When you run this, you should see:

```
VSIM 1> run 100ns;quit

#  1 0
#  2 1
#  3 2
#  4 3
#  5 4
#  6 5
#  7 6
#  8 7
#  9 8
# 10 9
```

Note how the input to the accumulator register and the output differ by one at every instant in time.

Watching for Unusual Conditions

You can also include conditional `$display` statements to alert you if some situation comes up that you weren't expecting. For example, we could change the body of the `always` block in the previous example to:

```
always @(posedge clk) begin
   if (acc_in==3) begin
     $display("crisis! ", acc_out);
   end
end
```

And we would get this:

```
VSIM 1> run 100ns;quit
# crisis!  2
```

Often this sort of debugging can be far more useful than trying to dig through several hundred nanoseconds' worth of waveforms in the gui.

Remember, the code you submit must not contain `$display` statements anywhere outside the testbench!

Viewing Waveforms

You can view the waveform output from your program using the program gtkwave. First, however, you must make sure that your testbench is dumping the appropriate information.

Add the following line to the top of your testbench module's body:

```
initial $dumpfile("waveform.vcd");
```

Then, locate each signal that you would like to observe (don't forget the clock!), go into the verilog file in which it appears, and add the following line to the top of the module's body:

```
initial $dumpvars(variablename);
```

... where `variablename` is replaced with the name of the variable you would like to observe.

Then, run your program in modelsim as shown above

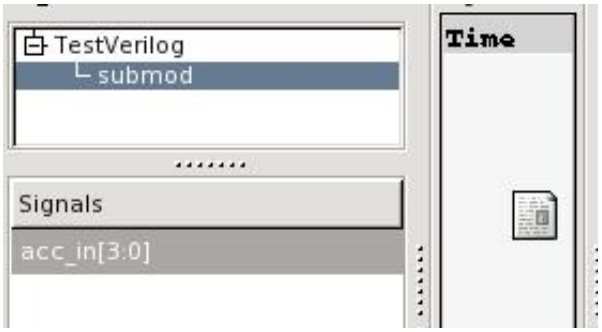
```
vlog *.v && vsim -c -do 'run -all;quit' TopLevelModule
```

This will generate a file called `waveform.vcd`. Now, open `gtkwave` to view this file (you must use one of the `ilinux` machines for this; the solaris machines will not work):

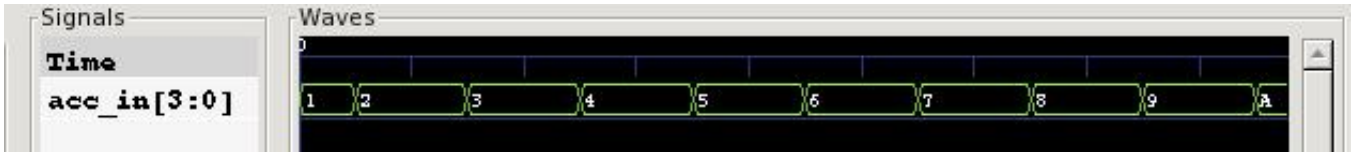
```
/home/tmp/cs61c-tb/prefix/bin/gtkwave waveform.vcd
```

When gtkwave opens, you will need to do a few things. First click on the tree to expand the list of signals:

Then, drag the signals you want to view into the middle pane:



Finally, click on the signal you want to view, and it should appear in the waveform area.



rendered from WIX