UMBC

AN HONORS UNIVERSITY IN MARYLAND

Department of Computer Science and Electrical Engineering

CMPE 415

Debugging and Monitoring Statements

Prof. Ryan Robucci

Simulation Commands

These commands are provided for use in procedural code.
All are stripped from code before synthesis

$display is evaluated and displayed immediately
$write is same as **$display**, but doesn't add newline
use \n as desired instead

$strobe is scheduled to be evaluated and displayed
at the end of the time step.
- Use it to report at the end of the timestep what the variables values are at is at the end of the timestep

$monitor is scheduled immediately and at the end of future time steps when the inputs change
- Only runs once per time step
- Use it to set up automatic repeated reporting of value changes

DUT module with some inbuilt debugging statements: $display and $strobe

```
module and4( y_out, x_in);
 input [3:0] x_in;    output y_out;
 reg y_out; integer k;
 always @ (x_in) begin:and_loop
  y_out = 1;
  for(k=0; k<=3; k=k+1) begin
   if (x_in[k] == 0) begin
    y_out = 0;
    // disable and_loop;  // faster sim
   end
   $display("$display at time=%0d: x_in='b%b y_out='b%b",
             $time, x_in,y_out);
   $strobe ("$strobe  at time=%0d: x_in='b%b y_out='b%b",
             $time, x_in,y_out);
  end //end for
 end //end always
endmodule
```

Testbench with $monitor

```
`timescale 1ns / 1ps
module and4_tb; // Design Unit Testbench
 parameter STOP_TIME = 10000;
 reg       [3:0] x_in;
 wire   y_out;
 and4 M1 (y_out,x_in);// Instantiate DUT

 // Create DUT response monitor
 initial $monitor ($time, " $monitor x_in = %b y_out = %b",
                   x_in, y_out);

 initial begin
  // Create DUT stimulus generator
  #10     x_in = 4'b0000;
  #10     x_in = 4'b0011;
 end

 initial #STOP_TIME $finish;
endmodule
```

Result

```
                    0 $monitor x_in = xxxx y_out = x
$display at time=10: x_in='b0000 y_out='b0
$display at time=10: x_in='b0000 y_out='b0
$display at time=10: x_in='b0000 y_out='b0
$display at time=10: x_in='b0000 y_out='b0
                   10 $monitor x_in = 0000 y_out = 0
$strobe at  time=10: x_in='b0000 y_out='b0
$strobe at  time=10: x_in='b0000 y_out='b0
$strobe at  time=10: x_in='b0000 y_out='b0
$strobe at  time=10: x_in='b0000 y_out='b0
$display at time=20: x_in='b0011 y_out='b1
$display at time=20: x_in='b0011 y_out='b1
$display at time=20: x_in='b0011 y_out='b0
$display at time=20: x_in='b0011 y_out='b0
                   20 $monitor x_in = 0011 y_out = 0
$strobe at time=20: x_in='b0011 y_out='b0
$strobe at time=20: x_in='b0011 y_out='b0
$strobe at time=20: x_in='b0011 y_out='b0
$strobe at time=20: x_in='b0011 y_out='b0
```

Testbench with $strobe and $display

```
`timescale 1ns / 1ps
module and_tb;   // Design Unit Testbench
  parameter STOP_TIME = 10000;
  reg [3:0] a;
  wire      y;

  // Instantiate DUT
  and I1 (y,a[3],a[2],a[1],a[0]);
  initial begin // Create DUT stimulus generator with print
      #10;
      a = 4'b1100;
      $strobe ($time,"%0t:$strobe1 a = %b y = %b", $time, a, y);
      $display ($time,"%0t:$display1 a = %b y = %b", $time, a, y);

      #10;
      a[0] = 1'b1;
      $strobe ($time,"%0t:$strobe2 a = %b y = %b", $time, a, y);
      $display ($time,"%0t:$display2 a = %b y = %b", $time, a, y);
      a[1] = 1'b1;
      $strobe ($time,"%0t:$strobe3 a = %b y = %b", $time, a, y);
      $display ($time,"%0t:$display3 a = %b y = %b", $time, a, y);
   end // initial begin
endmodule // and_tb
```

Result

1010000:$display1 a = 1100 y = x
1010000:$strobe1 a = 1100 y = 0
2020000:$display2 a = 1101 y = 0
2020000:$display3 a = 1111 y = 0
2020000:$strobe2 a = 1111 y = 0
2020000:$strobe3 a = 1111 y = 0

Testbench with $strobe and $display

```
`timescale 1ns / 1ps
module and_tb;   // Design Unit Testbench
  parameter STOP_TIME = 10000;
  reg [3:0] a;
  wire      y;

  // Instantiate DUT
  and I1 (y,a[3],a[2],a[1],a[0]);
  initial begin // Create DUT stimulus generator with print
      #10;
      a = 4'b1100;

      #10;
      $strobe ($time,"%0t:$strobe1 a = %b y = %b", $time, a, y);
      $display ($time,"%0t:$display1 a = %b y = %b", $time, a, y);
      a[0] = 1'b1;
      $strobe ($time,"%0t:$strobe2 a = %b y = %b", $time, a, y);
      $display ($time,"%0t:$display2 a = %b y = %b", $time, a, y);
      a[1] = 1'b1;
      $strobe ($time,"%0t:$strobe3 a = %b y = %b", $time, a, y);
      $display ($time,"%0t:$display3 a = %b y = %b", $time, a, y);
   end // initial begin
endmodule // and_tb
```

Result

2020000:$display1 a = 1100 y = 0
2020000:$display2 a = 1101 y = 0
2020000:$display3 a = 1111 y = 0
2020000:$strobe1 a = 1111 y = 1
2020000:$strobe2 a = 1111 y = 1
2020000:$strobe3 a = 1111 y = 1

More on Monitor

Only one $monitor may be active at a time
  subsequent $monitor calls while a monitor is active do not have an effect

$monitoron, $monitoroff can be used to disable (deactivate) and enable (activate) the monitor task

$monitor is typically only called once during a simulation unlike $display and $strobe

Printing Current Context in Hierarchy:

When printing, it is often useful to know which instance made the call:

```
module (...)
  inv1 M1(a,b);
  inv1 M1(c,d);
endmodule
module inv1(y,x);
output y
input x;
 always @ (x) begin
   $display("Instance %m is doing something");
 end
endmodule
```

See: "ESCAPE SEQUENCES IN FORMAT STRINGS" http://www.asic-world.com/verilog/vqref1.html

Variable Scope

Variable scope is the module,task,function, or named procedural block (begin..end) in which they are defined

For simulation, you will often want to peek downward into the hierarchy. Ex:

```
my_block M1 (a,b,c);
$monitor(M1.I1.my_procedure.count)
```

Upward searching for a variable not locally defined is automatic, but adhere to sensible coding practices.

File IO

Some file IO shown in testbench example on course website
$fopen, $fclose
File tasks $fdisplay, $fstrobe $fmonitor and $fwrite are like their non-file-IO counterparts
Syntax:

```
    handle1=$fopen("filenam1.suffix")
    handle2=$fopen("filenam2.suffix")

    //strobe data into filenam1.suffix
    $fstrobe(handle1, format, variable list)

    //write data into filenam2.suffix
    $fdisplay(handle2, format, variable list)

    //write data into filenam2.suffix all on one line. Put
    // in the format string where a new line is desired.
    $fwrite(handle2, format, variable list)
```

http://www.asic-world.com/verilog/sys_task_func1.html