Debugging and Monitoring Statements

CMPE 415  Programmable Logic Devices

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
        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
0 $monitor x_in = x0000 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

$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: $\text{display1} \quad a = 1100 \quad y = x$
1010000: $\text{strobe1} \quad a = 1100 \quad y = 0$
2020000: $\text{display2} \quad a = 1101 \quad y = 0$
2020000: $\text{display3} \quad a = 1111 \quad y = 0$
2020000: $\text{strobe2} \quad a = 1111 \quad y = 0$
2020000: $\text{strobe3} \quad a = 1111 \quad y = 0$
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", $time, a, y);
end // initial begin
endmodule // and_tb
2020000:$\text{display1 } a = 1100 \ y = 0$
2020000:$\text{display2 } a = 1101 \ y = 0$
2020000:$\text{display3 } a = 1111 \ y = 0$
2020000:$\text{strobe1 } a = 1111 \ y = 1$
2020000:$\text{strobe2 } a = 1111 \ y = 1$
2020000:$\text{strobe3 } a = 1111 \ y = 1$
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.
When printing, it is often useful to know which instance made the call:

```verilog
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 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:

```pascal
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

- File access is performed using system tasks and functions
- Files must be opened before used and should be closed
- Open files are managed/tracked using a 32-bit reg that must be declared before use
  - STDIN is pre-opened for reading, and STDOUT and STDERR are pre-opened for append and correspond to 0,1,2

File Output Example

```verilog
module fw_test;
    initial begin:block1 //naming a block allows
        //creating local variables
        reg [31:0] fd1;
        reg [7:0] x;
        x = 8'b1;
        fd1 = $fopen("output.txt"); //default mode
            // is overwrite
        if (fd1 == 0) begin
            $display("file open error");
            $finish();
        end
        $fdisplay(fd1,"x:"); //without a formating string
        $fdisplay(fd1,x); //without a variable
        $fdisplay(fd1,"display:%08b",x);
        $fwrite(fd1,"write:%08b",x);
        $fstrobe(fd1,"strobe:%08b",x);
        #1 //required last time advance to make strobe
            // output before file close
        $fclose(fd1);
    end
endmodule // file
```

Result in output.txt:

```
x:
  1
display:00000001
write:00000010strobe:00000001
```