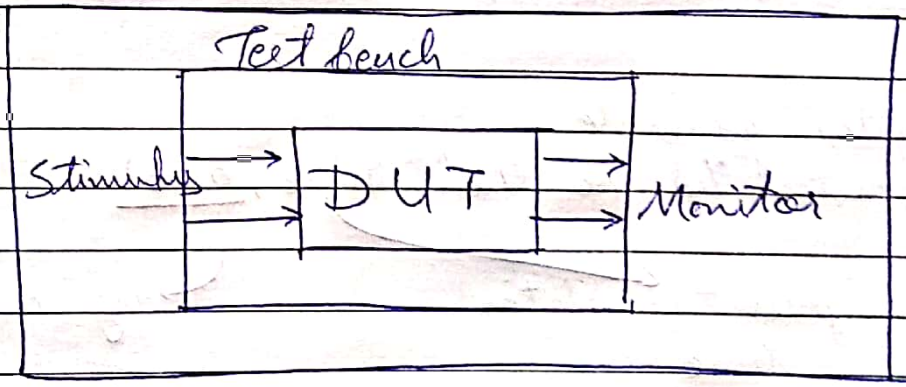
Lecture - 15

✱ Recap: Blocking & Non-Blocking Assignments

✱ Test Bench

- It is a procedural block that executes only once
- Used for simulation.
- Test bench generates clock, reset & required test vectors for a given DESIGN UNDER TEST (DUT)
- Inputs & outputs of the DUT should be connected to the testbench.
- Test bench uses "initial" block that executes only once.
- Test bench can also use "always" block to generate some test signal like a clock signal



✱ How to write a test bench?

* create a dummy template
  - declare inputs to the DUT as "reg" & outputs as wire.
  - instantiate the DUT
* initialize & monitor
  - assign some known values to the DUT inputs
  - monitor the DUT outputs for functional verification
* test bench can include various SIMULATOR DIRECTIVES like $display, $monitor, $dumpfile, $finish, etc.

✱ Simulator Directives

* $display : used to print immediate values of text or variable to stdout
  - syntax is very similar to "printf" of 'C'.

* $monitor : - similar to $display but does not print immediately
  - prints the value whenever the value changes i.e. 'event-driven'

* $display example

```
$display ("%b, %d %h", a, b, c);
```

reg a; integer b; reg [7:0] c;

* $monitor example

```
$monitor ("%d %b %b", $time, a, b)
```

$time → present simulation time

a, b → reg a, b;

* $finish : terminates the simulation process
  - Example: #100 $finish; //whenever $time ≥ 100, stop the simulation

* $dumpfile (filename);
  - specifies the file that will be used to store variables
  - file extension is .vcd (Value Change Dump)

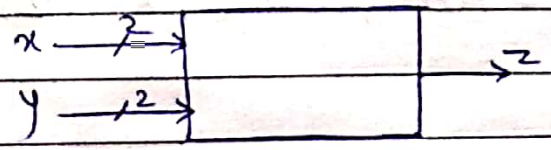
* $dumpoff;
  • stops the dumping of variables

* $dumpon;
  • resumes the previously stopped dumping of variables

* $dumpvars(level, list of variables); (or list of modules)
  • specifies which variables should be dumped
  • if level = 0, then all the variables are dumped.
  • if level = 1, only listed variables are dumped.

* $dumpall;
  • current values of all variables will be dumped

* Example: 2 bit equality checker




```
'timescale 1ns/100ps
module comparator(x, y, z);
  input [1:0] x, y;
  output z;
  assign z = (x[0] & y[0] & x[1] & y[1]) |
             (~x[0] & ~y[0] & x[1] & y[1]) |
             (~x[0] & ~y[0] & ~x[1] & ~y[1]) |
             (x[0] & y[0] & ~x[1] & ~y[1]);
endmodule
```

* Corresponding Test Bench

```
'timescale 1ns/100ps
module testbench;
    reg [1:0] x,y;
    wire z;
    comparator c2 (.x(x), .y(y), .z(z));
    initial
    begin
        $dumpfile ("comp.vcd");
        $dumpvars (0, testbench);
        x = 2'b01;
        y = 2'b00;
        #10 x = 2'b10; y = 2'b10;
        #10 x = 2'b01; y = 2'b11;
    end
    initial
        begin
            $monitor ("t = %d x = %2b y = %2b
                z = %d", $time, x, y, z);
        end
endmodule
```

DUT variable names

standard practise

stimulus → equal values