System Verilog Lab5

After completing this lab, you should be able to:

Implement testbench using program block.

Understand how to define I/O signals in program block

Understand how to specify the direction of the signals in program block.

*Implement **top** module which connects dut and TB.*

Verify DUT behaviour with the help of self-checking mechanism in program block(testbench).

LAB5:

Use Lab5 directory of this lab

Step 1: Convert module testbench to program block based testbench.

1. Open file testbench.sv and define program block with required ports.
   Ex:
   ```
   program testbench(clk,rst,addr,wr,wdata,rdata);
   ```

2. Define signal direction for the above added port list.
   Note:
   1. Define signal as output if your requirement is to drive signal in TB.
   2. Define signal as input if your requirement is to sample/read signal in TB.

   Ex: Add this code in Section 1 in testbench.sv
   ```
   input clk;
   output rst;
   output [7:0] wdata;
   ……….
   // define direction for all the ports
   ```
   // **Signal directions in TB are opposite to DUT port directions.**

Step 2:Define required TB methods.

1. Copy entire Section 4 from Lab4/testbench.sv file to Lab5/testbench.sv

Step 3: Collect DUT output in TB (program block).

1. Copy entire section 7 from Lab4/testbench.sv file to Lab5/testbench.sv

Step 4: Start the Verification flow.

1. Copy the entire section 5 and 6 from Lab4/testbench.sv file to Lab5/testbench.sv

   Ex:

   ```
     initial begin
     reset_stimulus();
     ……….
     end
   ```

Step 5: Define module top and instantiate DUT and TB.

1. Open file top.sv and define module top with required bunch of signals to connect DUT (ram_dut) and TB(program block).

   Ex:

   ```
    module top();
    logic clk;
    logic rst;
    logic [7:0] wdata;
    ……….
    ……….
    endmodule
   ```

2. Initialize clk to 0 in initial block and generate clock.

   Add this code in Section 2 in top.sv

   Ex:

   ```
   initial clk=0;
   always #5 clk = ~clk;
   ```

3. Instantiate ram_dut with instance name dut_inst along with port connections. use name based port mapping.

   Add this code in Section 3 in top.sv

   Ex:

   ```
   ram_dut dut_inst(.clk(clk),.rst(rst)……………….);
   ```

4. Instantiate program block with instance name tb_inst along with port connections. Use name based port mapping.

   Add this code in Section 4 in top.sv

   Ex:

   ```
    testbench tb_inst(.clk(clk),.rst(rst)………………..);
   ```

Step 6: Run the simulation and validate the output of DUT with the results printed by self-checking mechanism.

1. Check the test Passed or Failed and debug the if there are any failuers.