System Verilog Lab6

After completing this lab, you should be able to:

Implement interface block.

Understand how to define I/O signals in Interface

Understand how to specify filelist while compilation.

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

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

LAB6:

Use Lab6 directory of this lab

Step 1: Implement a simple interface.

1. Open file interface.sv and define interface block with required ports.
   Ex:
   ```
   interface simple_bus(input clk);
   ```

2. Define signals list.
   Ex: Add this code in Section 1 in testbench.sv
   ```
   logic          rst;
   logic          wr;
   logic   [2:0]  addr;
   ……….
   // define all the necessary ports
   ```

Step 2:Declare interface instance.

1. Open file top.sv and add the following code in section 1
   Ex:
   ```
   simple_bus intf_inst(clk);
   ```

Step 3: DUT and program block instance connection

1. Instantiate the DUT using port instantiation by name in section 2 of top.sv.
   Ex:
   ```
   ram_dut dut_inst(.clk(clk),.rst(intf_inst.rst),.wr(intf_inst.wr)……………………);
   ```

2. Instantiate the program block by passing the instance on the interface. Add the following code in section 3 of top.sv
   Ex:
   ```
   testbench  tb_inst(intf_inst);
   ```

Step 4: Port list of program block.

1. Add the below code in section 1 of testbench.sv
   Ex:
   ```
     program testbench(simple_bus vif);
   ```

Step 5: Accessing the interface signals

1. Add vif.<signal_name> to all the necessary signals in Section 2.
   Ex:
```
     task reset_stimulus();
              $display("[Testbench] Reset applied to DUT");
              vif.rst <= 1;
              @(posedge vif.clk);
              vif.rst <= 0;
              $display("[Testbench] Reset completed");
    endtask

task write_stimulus(input reg[2:0] addr_t,reg [7:0] wdata_t);
    vif.wr<=0;
    vif.wdata<=wdata_t; exp_q.push_back(wdata_t);
    vif.addr<=addr_t;
    @(posedge vif.clk);
endtask

task read_stimulus(input reg[2:0] addr_t);
    vif.wr<=1;
    vif.addr<=addr_t;
    @(posedge vif.clk);
endtask
```

Step 6: Add self-checking mechanism.

1. Add the below code in section 3 of testbench.sv.

```
initial begin
forever begin
```

```
@(vif.rdata);
dut_out =exp_q.pop_front();
if(dut_out == vif.rdata) begin
            matched++;
            $display("[Info] Rdata = %0d at mem address =
%0d",vif.rdata,vif.addr);
    end
else begin
    mis_matched++;
    $display("[Error] Exp Rdata = %0d Actual Rdata = %0d",dut_out,vif.rdata);
    end
end
end
```

Step 7: Add Test Pass and Fail criteria.

1. Add a small function in section 4 of testbench.sv to check for mis_matched == 0.

```
function void result();
        $display("\n***************************");
        if(mis_matched == 0)
                $display("*******Test Passed********");
        else
                $display("*********Test Failed********");
endfunction
```

2. Call the result function at section 5.