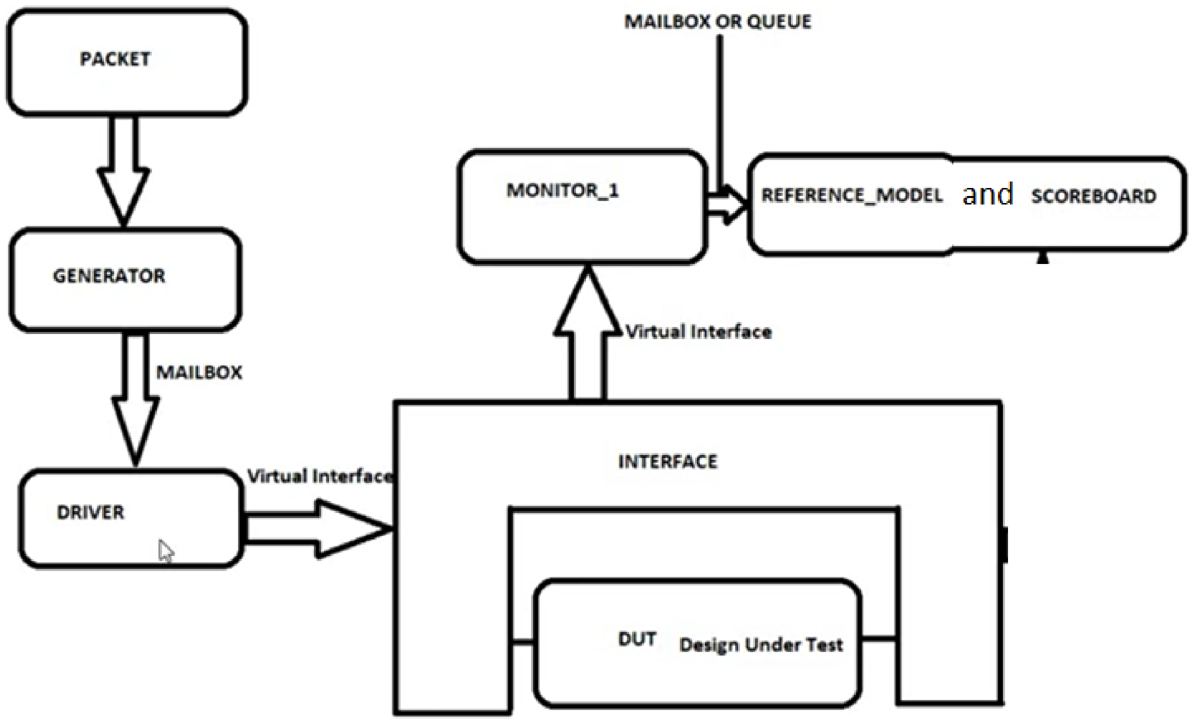
Experiment #6

Aim: Write the System Verilog testbench for a full adder with monitor and scoreboard.

Block level tesbench architecture is given below for your reference.



Example: Testbench development flow for half-adder given below can be referred.

1. **Write SV code for half adder**

```
module half_adder(s,c,a,b);
  input a,b;
  output s,c;
  xor x1( s,a,b);
  and a1(c,a,b);
endmodule
```

2. **Write SV interface for DUT.**

```
interface intf();

  logic a;
  logic b;
  logic sum;
  logic carry;

endinterface
```

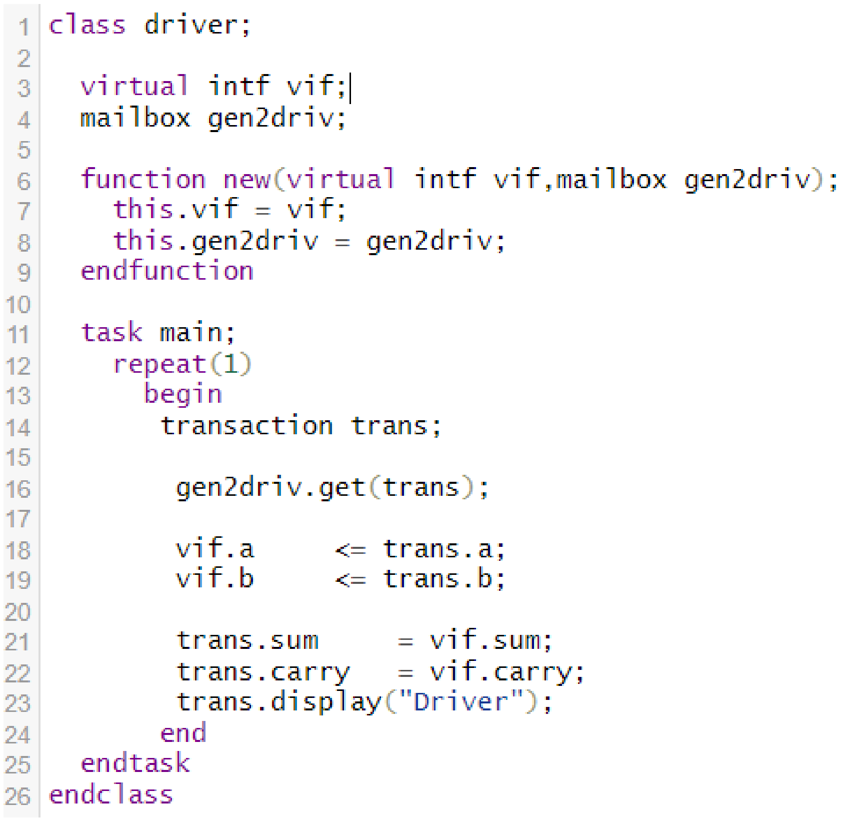
3. Write the SV class for packet-class (also called transaction class)

```
class transaction;

// Stimulus are declared with rand keyword

  rand bit a;
  rand bit b;

  bit sum;
  bit carry;

//Function for Displaying values of a , b and sum , carry
  function void display(string name);
    $display("-------------------------");
    $display(" %s ",name);
    $display("-------------------------");
    $display("a = %0d,    b = %0d",a,b);
    $display("sum = %0d, carry = %0d",sum,carry);
    $display("-------------------------");
  endfunction
```

4. Write the SV class for generator block.

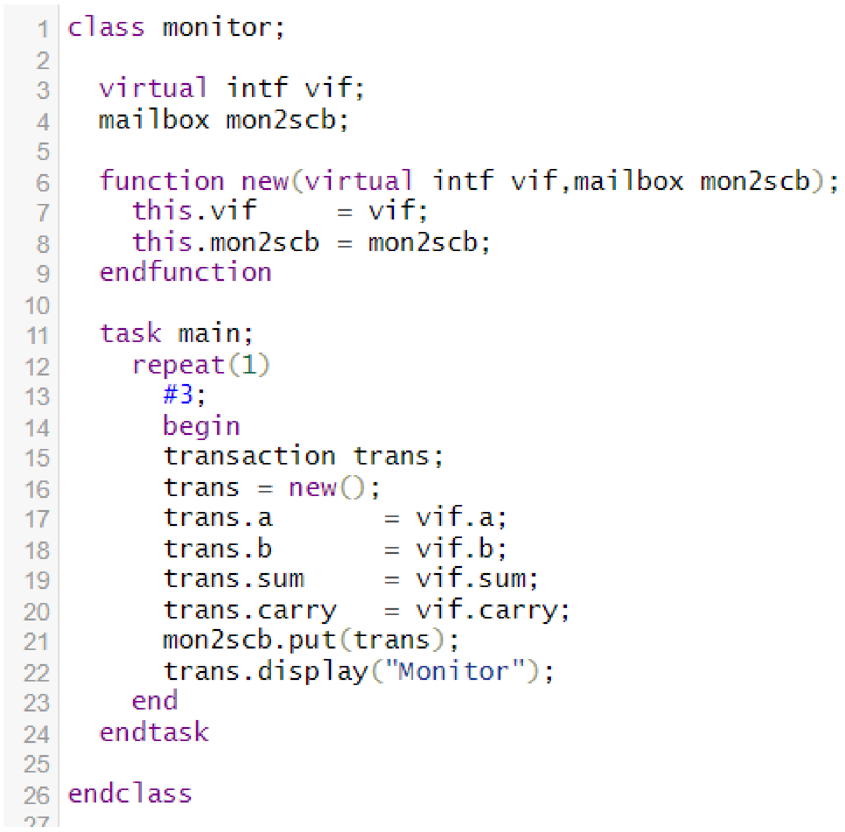
```
class generator;

  transaction trans; //Handle of Transaction class


  mailbox gen2driv;  //Mailbox declaration


  function new(mailbox gen2driv);  //creation of mailbox and constructor
    this.gen2driv = gen2driv;
  endfunction


  task main();

    repeat(1)
      begin
        trans = new();
        trans.randomize();
        trans.display("Generator");
        gen2driv.put(trans);
      end

  endtask

endclass
```

5. Write the SV class for driver block.



```
class driver;

  virtual intf vif;
  mailbox gen2driv;

  function new(virtual intf vif,mailbox gen2driv);
    this.vif = vif;
    this.gen2driv = gen2driv;
  endfunction

  task main;
    repeat(1)
      begin
       transaction trans;

        gen2driv.get(trans);

        vif.a     <= trans.a;
        vif.b     <= trans.b;

        trans.sum      = vif.sum;
        trans.carry    = vif.carry;
        trans.display("Driver");
       end
  endtask
endclass
```

6. Write the SV class for monitor block.

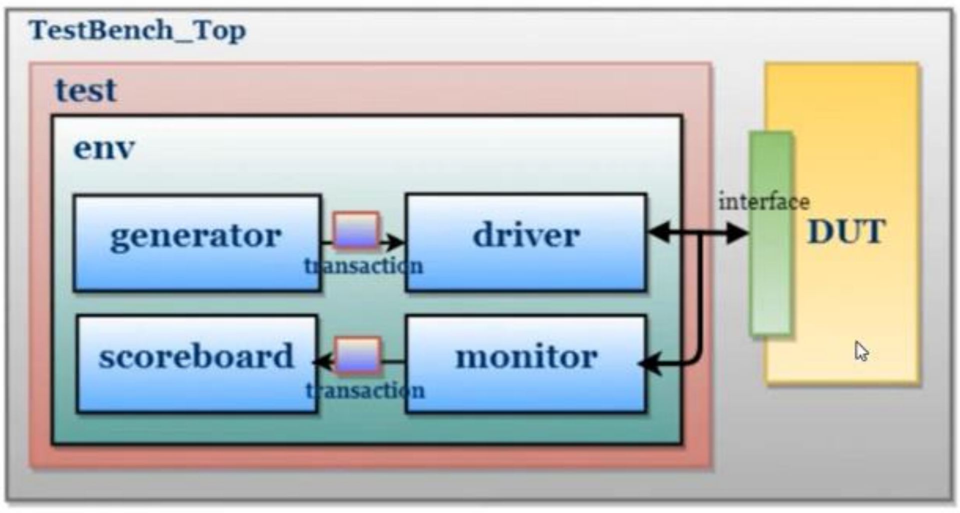


```
class monitor;

  virtual intf vif;
  mailbox mon2scb;

  function new(virtual intf vif,mailbox mon2scb);
    this.vif      = vif;
    this.mon2scb = mon2scb;
  endfunction

  task main;
    repeat(1)
      #3;
      begin
      transaction trans;
      trans = new();
      trans.a         = vif.a;
      trans.b         = vif.b;
      trans.sum       = vif.sum;
      trans.carry     = vif.carry;
      mon2scb.put(trans);
      trans.display("Monitor");
    end
  endtask

endclass
```

7. Write the SV class for scoreboard block.

```
class scoreboard;

  mailbox mon2scb;

  function new(mailbox mon2scb);
    this.mon2scb = mon2scb;
  endfunction

  task main;
    transaction trans;
    repeat(1)
      begin
      mon2scb.get(trans);

        if(((trans.a ^ trans.b) == trans.sum)  && ((trans.a & trans.b) == trans.carry))
          $display("Result is as Expected");
        else
          $error("Wrong Result");

         trans.display("Scoreboard");
    end
  endtask

endclass
```

8. Write the SV class for environment block.



```
`include "transaction.sv"
`include "generator.sv"
`include "driver.sv"
`include "monitor"
`include "scoreboard"

class environment;
  generator      gen;
  driver         driv;
  monitor        mon;
  scoreboard     scb;
  mailbox m1;
  mailbox m2;

  virtual intf vif;
  function new(virtual intf vif);
    this.vif = vif;
    m1   = new();
    m2   = new();
    gen  = new(m1);
    driv = new(vif,m1);
    mon  = new(vif,m2);
    scb  = new(m2);
  endfunction

  task test();
    fork
      gen.main();
      driv.main();
      mon.main();
      scb.main();
    join
  endtask


  task run;
    test();
    $finish;
  endtask

endclass
```

9. Write SV testbench with program block

```
`include "environment.sv"

program test(intf i_intf);
  environment env;

  initial
    begin
      env = new(i_intf);
      env.run();
    end

endprogram
```

10. Write SV code for testbench top to connect the interface, testbench and DUT. Run it and check the output.

```
`include "interface.sv"
`include "test"
module tbench_top;

  intf i_intf();

  test t1(i_intf);

  half_adder h1 (
    .a(i_intf.a),
    .b(i_intf.b),
    .s(i_intf.sum),
    .c(i_intf.carry)
  );

endmodule
```