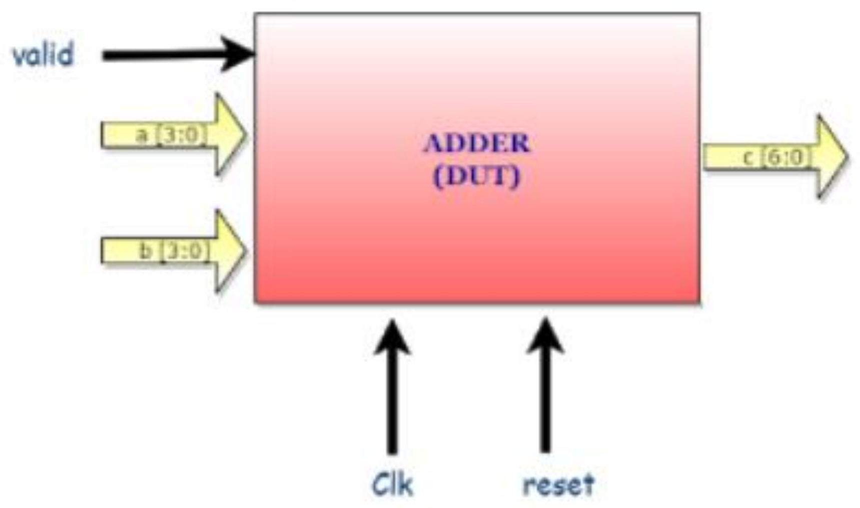
Exp-8: Write the System Verilog testbench for the synchronous 4-bit adder with monitor and scoreboard.

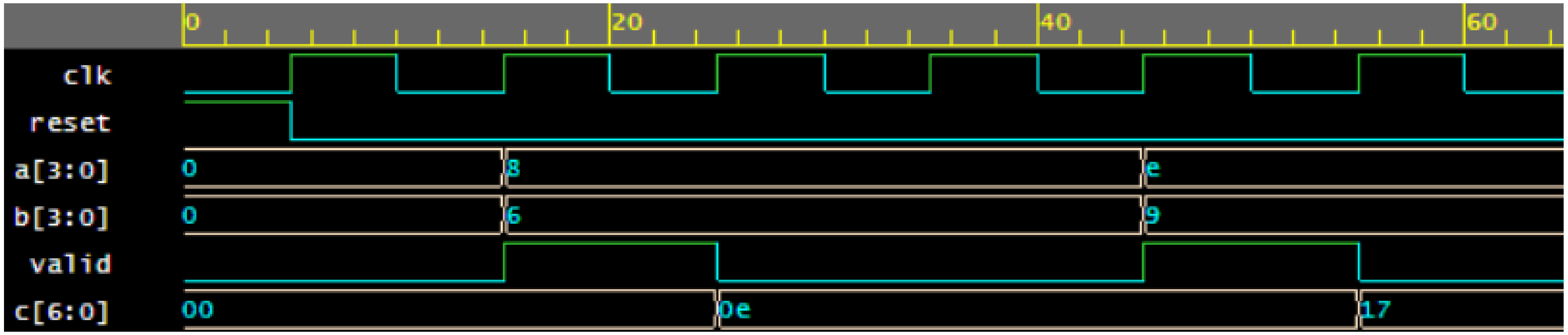
SV Testbench for Adder

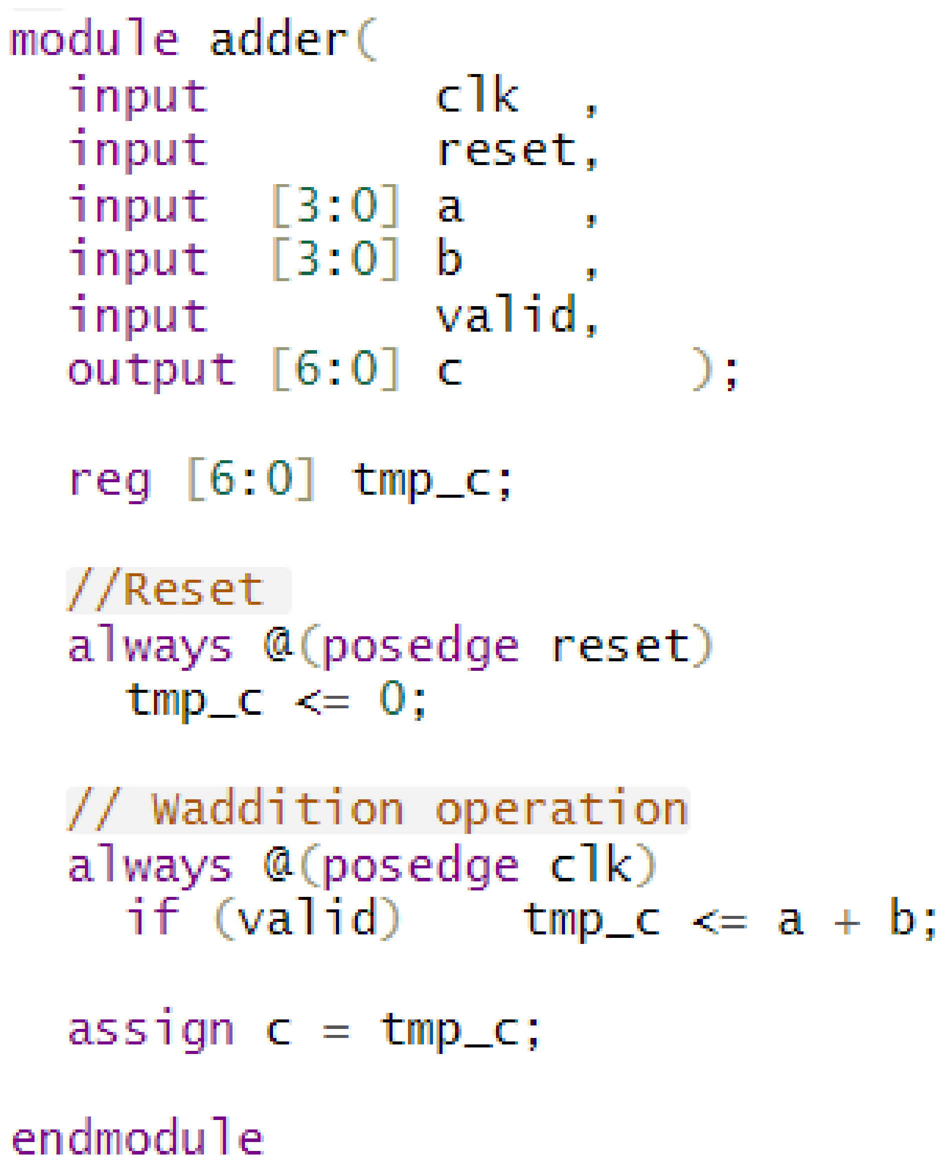


- Adder is,
  - fed with the inputs clock, reset, a, b and valic
  - has output is c.

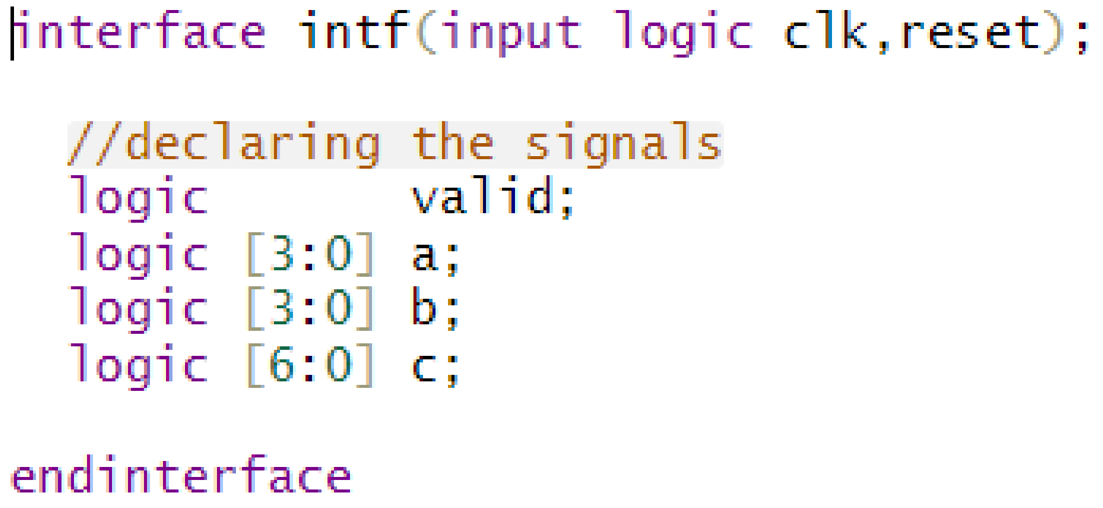
- The valid signal indicates the valid value on the a and b
- On valid signal, adder will add the a and b, drives the result in the next clock on c.

- Adder wavefrom



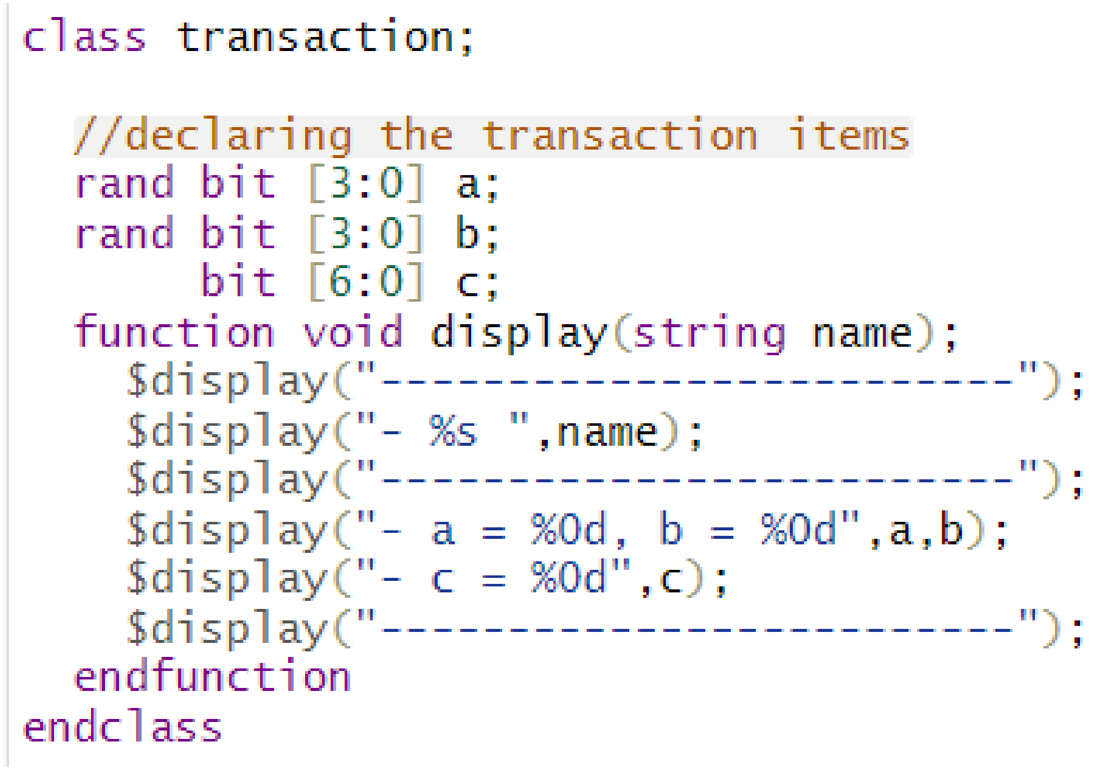


```
module adder(
  input         clk   ,
  input         reset,
  input   [3:0] a     ,
  input   [3:0] b     ,
  input         valid,
  output  [6:0] c         );

  reg [6:0] tmp_c;

  //Reset
  always @(posedge reset)
    tmp_c <= 0;

  // Waddition operation
  always @(posedge clk)
    if (valid)    tmp_c <= a + b;

  assign c = tmp_c;

endmodule
```



```
interface intf(input logic clk,reset);

  //declaring the signals
  logic       valid;
  logic [3:0] a;
  logic [3:0] b;
  logic [6:0] c;

endinterface
```

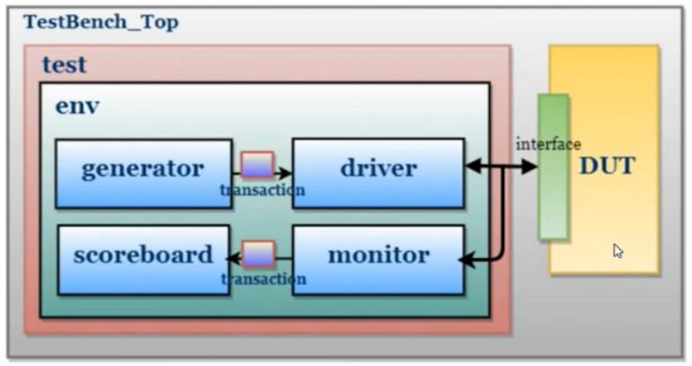
Packet class

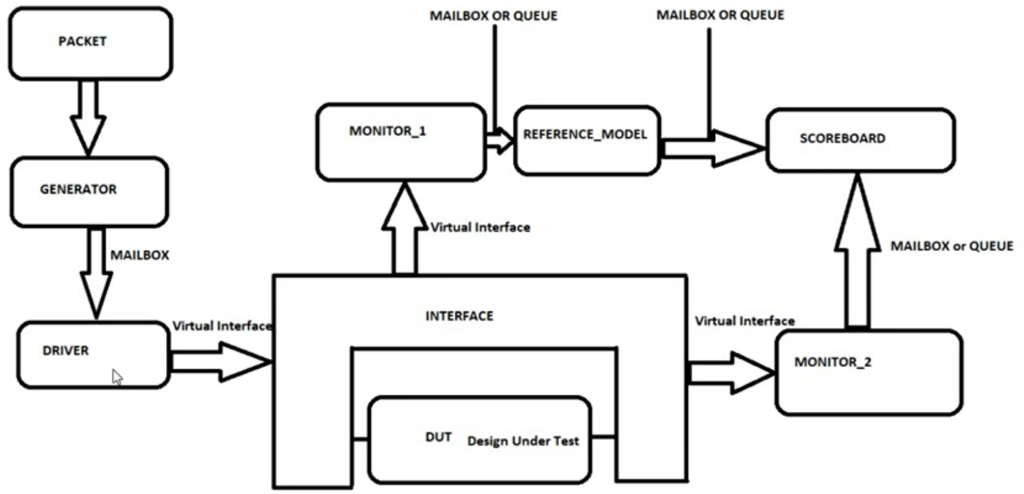


```
class transaction;

  //declaring the transaction items
  rand bit [3:0] a;
  rand bit [3:0] b;
       bit [6:0] c;
  function void display(string name);
    $display("-------------------------");
    $display("- %s ",name);
    $display("-------------------------");
    $display("- a = %0d, b = %0d",a,b);
    $display("- c = %0d",c);
    $display("-------------------------");
  endfunction
endclass
```

Generator class

```
class generator;

  //declaring transaction class
  rand transaction trans;

  //repeat count, to specify number of items to generate
  int  repeat_count;

  //mailbox, to generate and send the packet to driver
  mailbox gen2driv;

  //event, to indicate the end of transaction generation
  event ended;

  //constructor
  function new(mailbox gen2driv);
    //getting the mailbox handle from env, in order to share the transaction packet between
the generator and driver, the same mailbox is shared between both.
    this.gen2driv = gen2driv;
  endfunction

  //main task, generates(create and randomizes) the repeat_count number of transaction
packets and puts into mailbox
  task main();
    repeat(repeat_count) begin
    trans = new();
    if( !trans.randomize() ) $fatal("Gen:: trans randomization failed");
      trans.display("[ Generator ]");
      gen2driv.put(trans);
    end
    -> ended; //triggering indicatesthe end of generation
  endtask

endclass
```

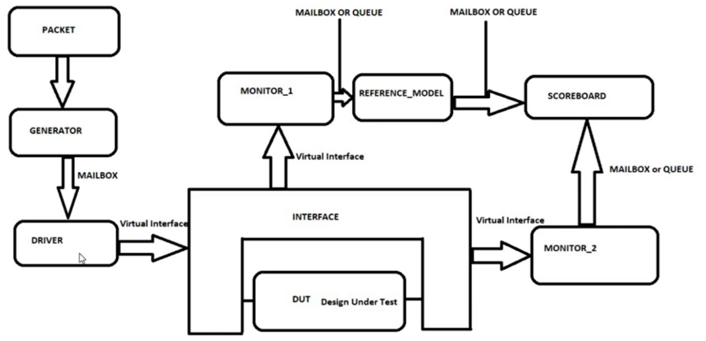




Driver: gets the packet from generator and drive the transaction paket items into interface (interface is connected to DUT, so the items driven into interface signal will get driven in to DUT)

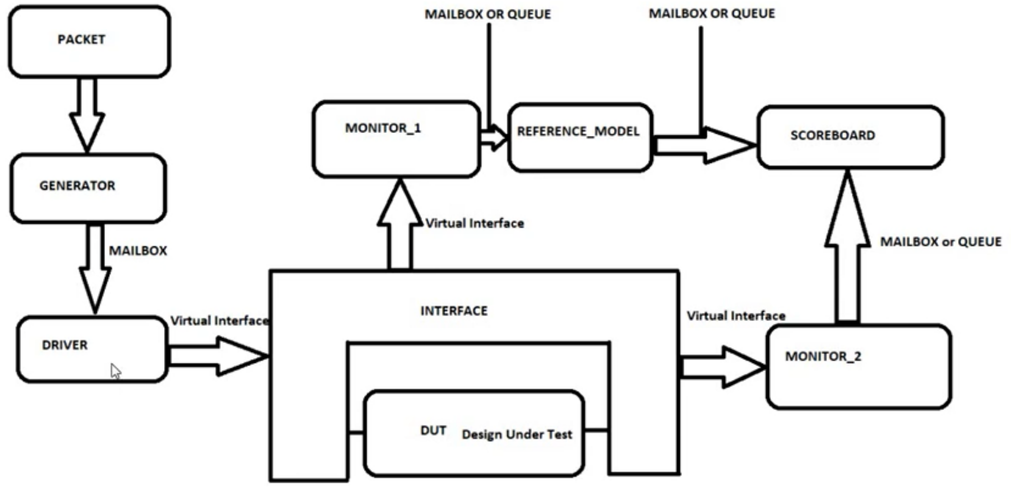
```
class driver;

  //used to count the number of transactions
  int no_transactions;

  //creating virtual interface handle
  virtual intf vif;

  //creating mailbox handle
  mailbox gen2driv;

  //constructor
  function new(virtual intf vif,mailbox gen2driv);
    //getting the interface
    this.vif = vif;
    //getting the mailbox handles from  environment
    this.gen2driv = gen2driv;
  endfunction
```

```
//Reset task, Reset the Interface signals to default/initial values
task reset;
  wait(vif.reset);
  $display("[ DRIVER ] ----- Reset Started -----");
  vif.a <= 0;
  vif.b <= 0;
  vif.valid <= 0;
  wait(!vif.reset);
  $display("[ DRIVER ] ----- Reset Ended   -----");
endtask

//drivers the transaction items to interface signals
task main;
  forever begin
    transaction trans;
    gen2driv.get(trans);
    @(posedge vif.clk);
    vif.valid <= 1;
    vif.a     <= trans.a;
    vif.b     <= trans.b;
    @(posedge vif.clk);
    vif.valid <= 0;
    trans.c   = vif.c;
    @(posedge vif.clk);
    trans.display("[ Driver ]");
    no_transactions++;
  end
endtask
```



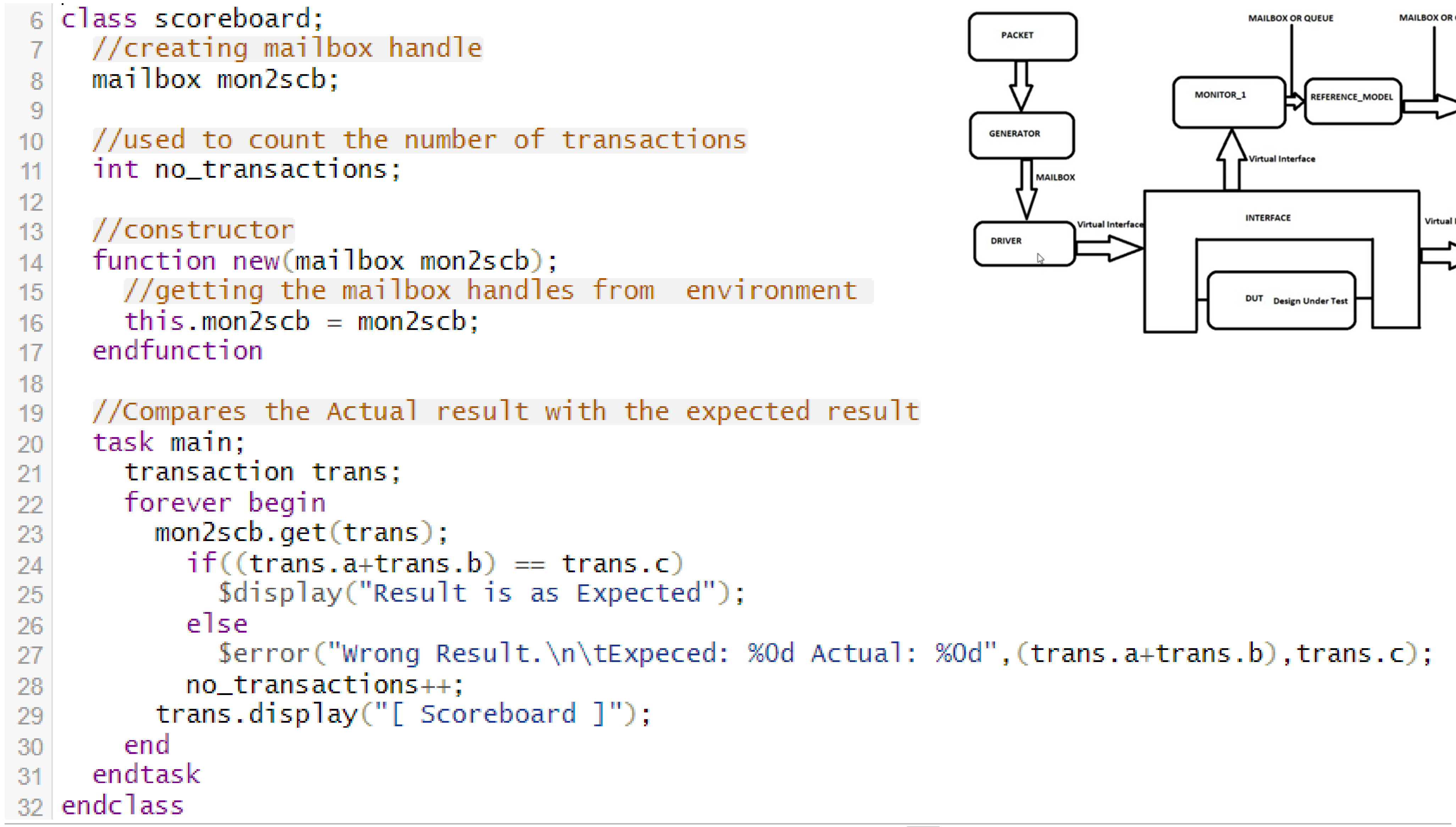
class monitor: Samples the interface signals, captures into transaction packet and send the packet to scoreboard.

```
class monitor;

  //creating virtual interface handle
  virtual intf vif;

  //creating mailbox handle
  mailbox mon2scb;

  //constructor
  function new(virtual intf vif,mailbox mon2scb);
    //getting the interface
    this.vif = vif;
    //getting the mailbox handles from  environment
    this.mon2scb = mon2scb;
  endfunction
```

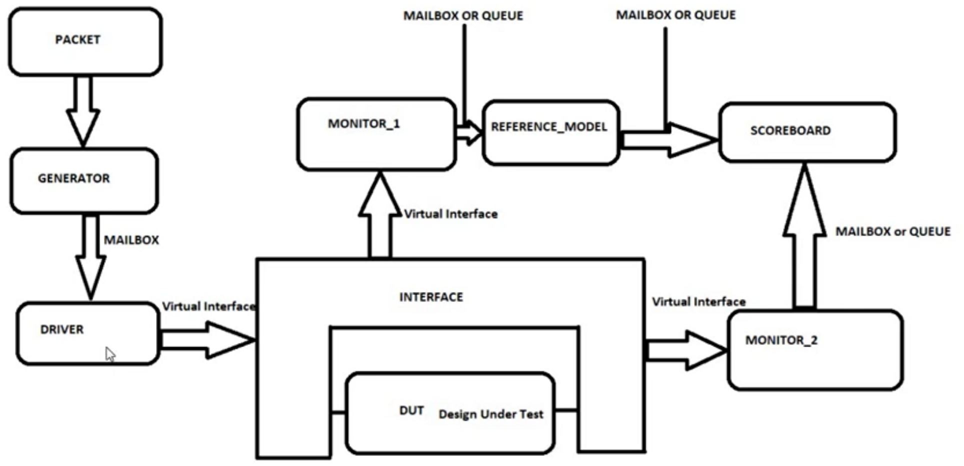


```
  //Samples the interface signal and send the
sample packet to scoreboard
  task main;
    forever begin
      transaction trans;
      trans = new();
      @(posedge vif.clk);
      wait(vif.valid);
      trans.a   = vif.a;
      trans.b   = vif.b;
      @(posedge vif.clk);
      trans.c   = vif.c;
      @(posedge vif.clk);
      mon2scb.put(trans);
      trans.display("[ Monitor ]");
    end
  endtask

endclass
```

Scoreboard: gets the packet from monitor, Generated the expected result and compares with the actual result received from Monitor.



```
class scoreboard;
  //creating mailbox handle
  mailbox mon2scb;

  //used to count the number of transactions
  int no_transactions;

  //constructor
  function new(mailbox mon2scb);
    //getting the mailbox handles from  environment
    this.mon2scb = mon2scb;
  endfunction

  //Compares the Actual result with the expected result
  task main;
    transaction trans;
    forever begin
      mon2scb.get(trans);
        if((trans.a+trans.b) == trans.c)
          $display("Result is as Expected");
        else
          $error("Wrong Result.\n\tExpeced: %0d Actual: %0d",(trans.a+trans.b),trans.c);
        no_transactions++;
      trans.display("[ Scoreboard ]");
    end
  endtask
endclass
```

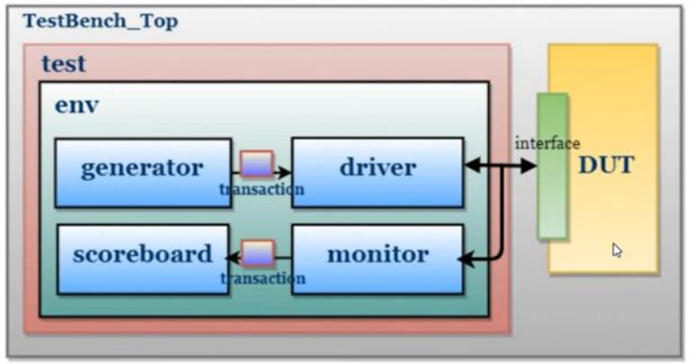


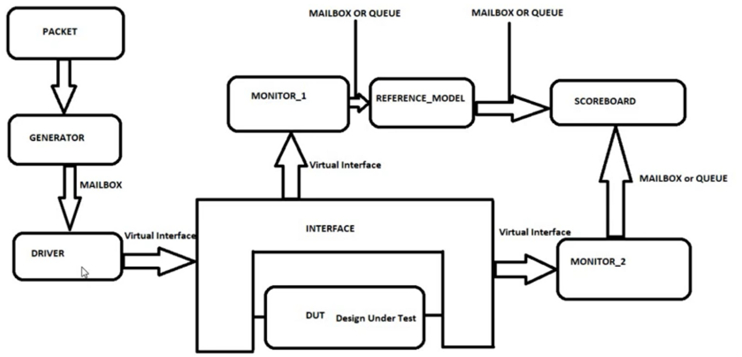
class environment

```
`include "transaction.sv"
`include "generator.sv"
`include "driver.sv"
`include "monitor.sv"
`include "scoreboard.sv"
class environment;

  //generator and driver instance
  generator      gen;
  driver         driv;
  monitor        mon;
  scoreboard     scb;

  //mailbox handle's
  mailbox gen2driv;
  mailbox mon2scb;

  //virtual interface
  virtual intf vif;
```

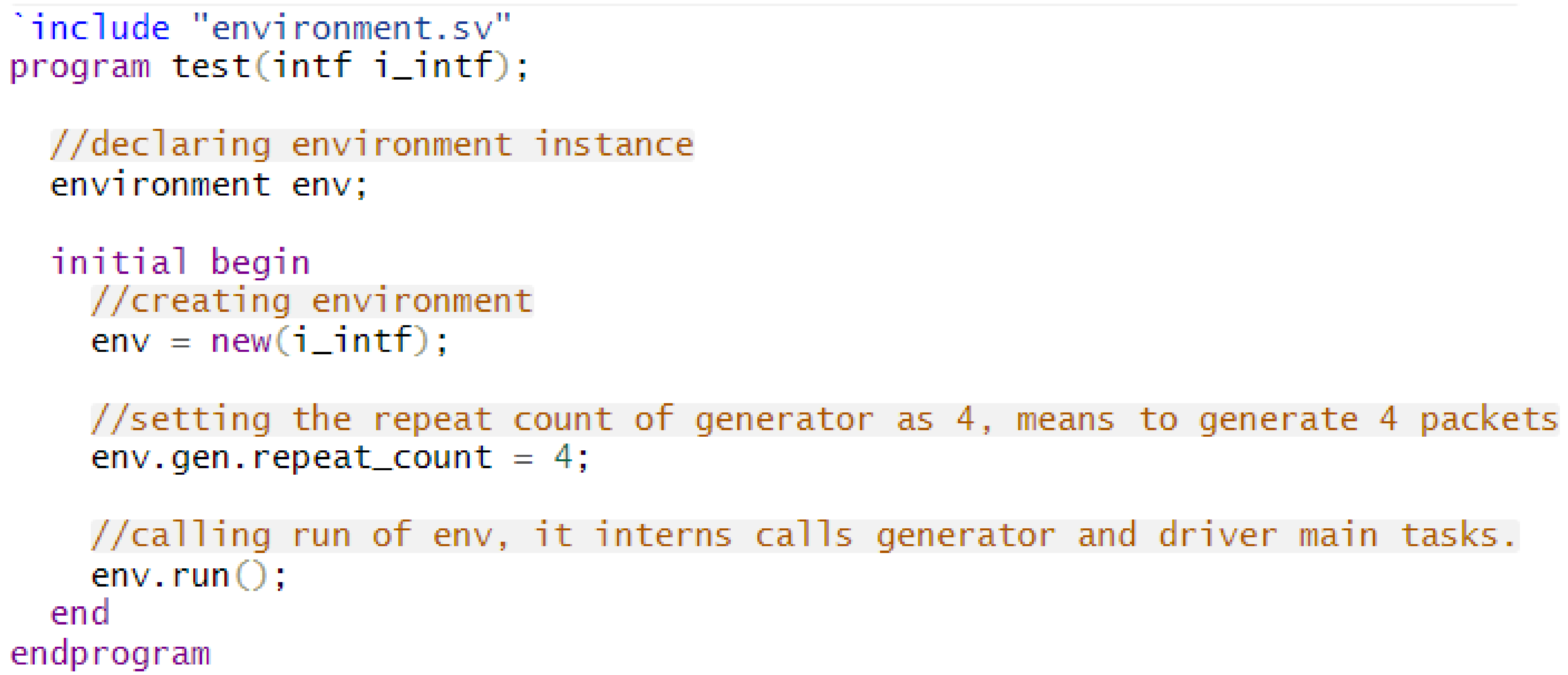
```
//creating the mailbox (Same handle will be shared across
generator and driver)
    gen2driv = new();
    mon2scb  = new();

    //creating generator and driver
    gen  = new(gen2driv);
    driv = new(vif,gen2driv);
    mon  = new(vif,mon2scb);
    scb  = new(mon2scb);
  endfunction

  //
  task pre_test();
    driv.reset();
  endtask

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

```
  task post_test();
    wait(gen.ended.triggered);
    wait(gen.repeat_count == driv.no_transactions); //Optional
    wait(gen.repeat_count == scb.no_transactions);
  endtask

  //run task
  task run;
    pre_test();
    test();
    post_test();
    $finish;
  endtask

endclass
```





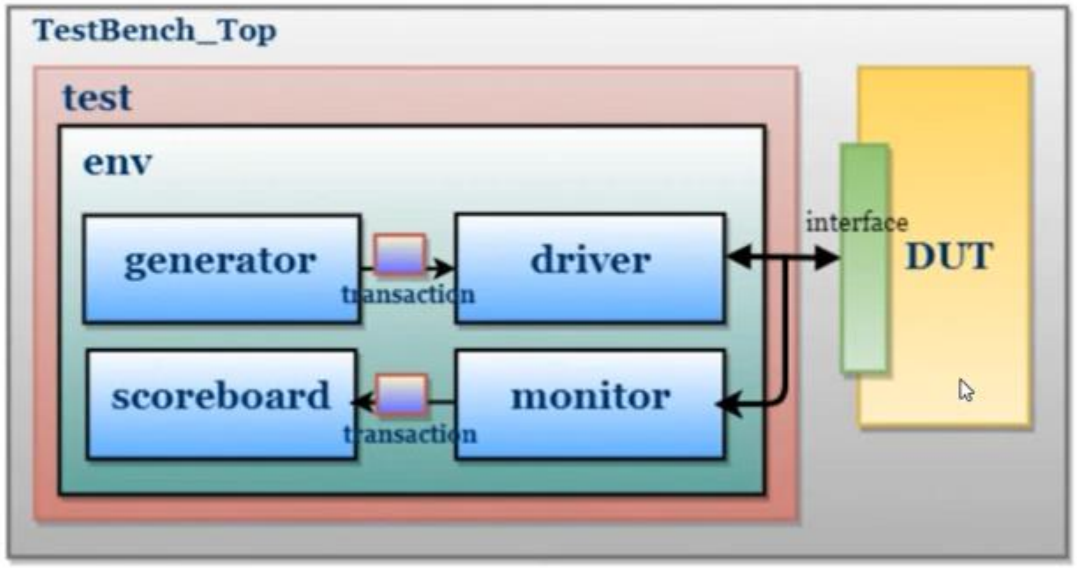
Random test



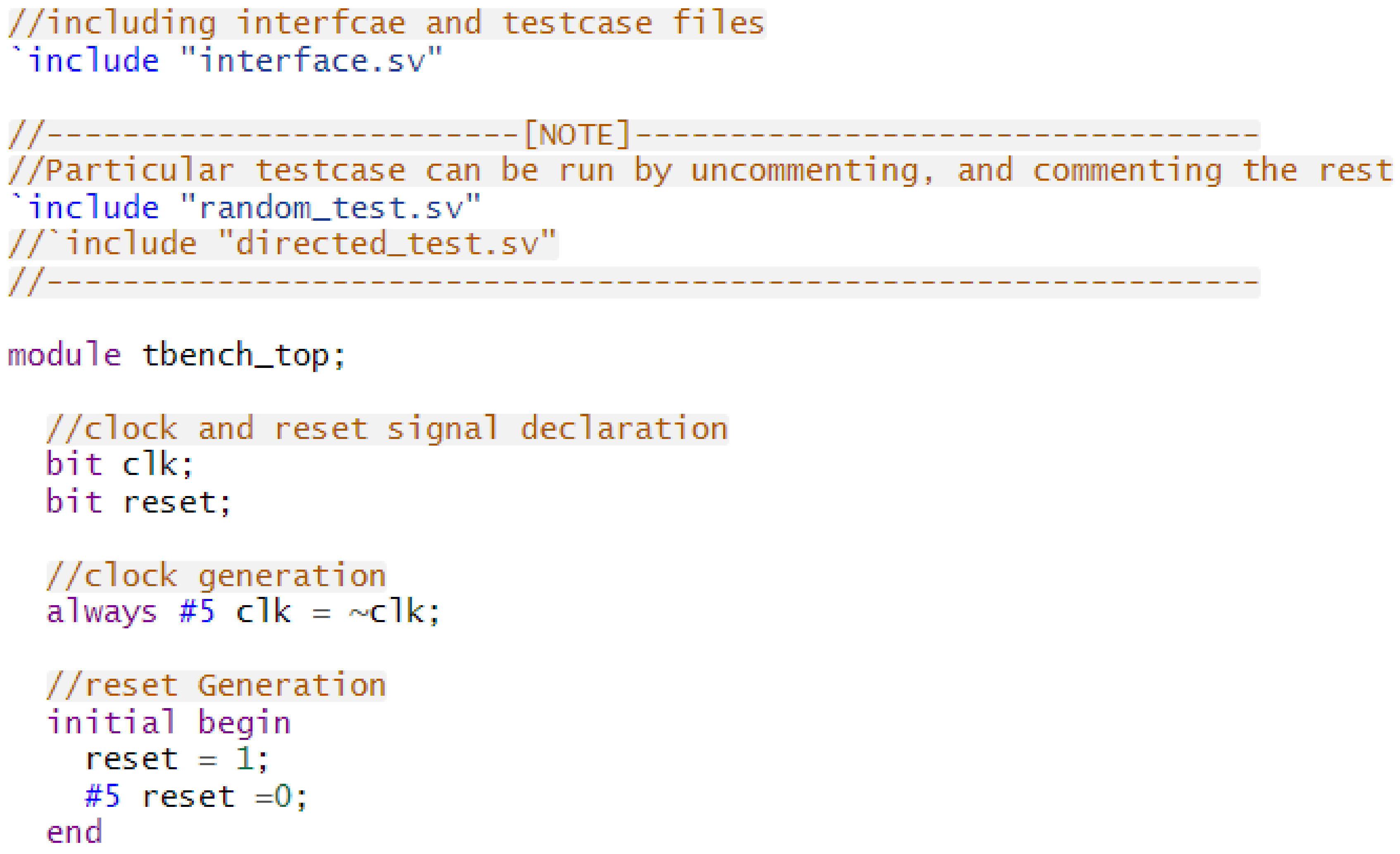
```
`include "environment.sv"
program test(intf i_intf);

  //declaring environment instance
  environment env;

  initial begin
    //creating environment
    env = new(i_intf);

    //setting the repeat count of generator as 4, means to generate 4 packets
    env.gen.repeat_count = 4;

    //calling run of env, it interns calls generator and driver main tasks.
    env.run();
  end
endprogram
```

Directed test

```
`include "environment.sv"
program test(intf i_intf);

  class my_trans extends transaction;

    bit [1:0] count;

    function void pre_randomize();
      a.rand_mode(0);
      b.rand_mode(0);

      a = 10;
      b = 12;
    endfunction

  endclass

  //declaring environment instance
  environment env;
  my_trans my_tr;

  initial begin
    //creating environment
    env = new(i_intf);

    my_tr = new();

    //setting the repeat count of generator as 4, means
to generate 4 packets
    env.gen.repeat_count = 10;

    env.gen.trans = my_tr;

    //calling run of env, it interns calls generator and
driver main tasks.
    env.run();
  end
endprogram
```



Testbench top: tbench_top or testbench top, this is the top most file, in which DUT(Design Under Test) and Verification environment are connected



```
//including interfcae and testcase files
`include "interface.sv"

//-------------------------[NOTE]---------------------------------
//Particular testcase can be run by uncommenting, and commenting the rest
`include "random_test.sv"
//`include "directed_test.sv"
//----------------------------------------------------------------

module tbench_top;

  //clock and reset signal declaration
  bit clk;
  bit reset;

  //clock generation
  always #5 clk = ~clk;

  //reset Generation
  initial begin
    reset = 1;
    #5 reset =0;
  end
  //creatinng instance of interface, inorder to connect DUT and testcase
  intf i_intf(clk,reset);

  //Testcase instance, interface handle is passed to test as an argument
  test t1(i_intf);

  //DUT instance, interface signals are connected to the DUT ports
  adder DUT (
    .clk(i_intf.clk),
    .reset(i_intf.reset),
    .a(i_intf.a),
    .b(i_intf.b),
    .valid(i_intf.valid),
    .c(i_intf.c)
   );

  //enabling the wave dump
  initial begin
    $dumpfile("dump.vcd"); $dumpvars;
  end
endmodule
```