VERIFICATION OF DIGITAL SYSTEMS

UE21EC343AB5

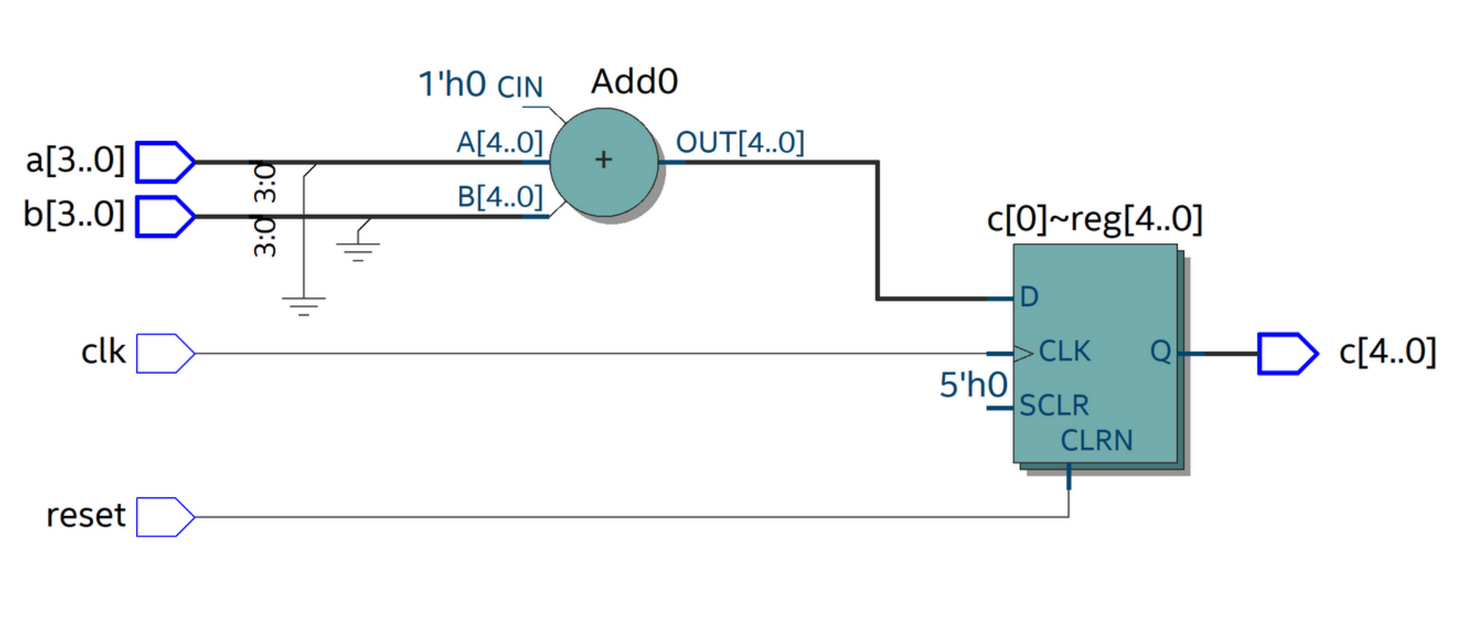
SYNCHRONOUS 4-BIT ADDER

ANIRUDH N S
PES2UG21EC015

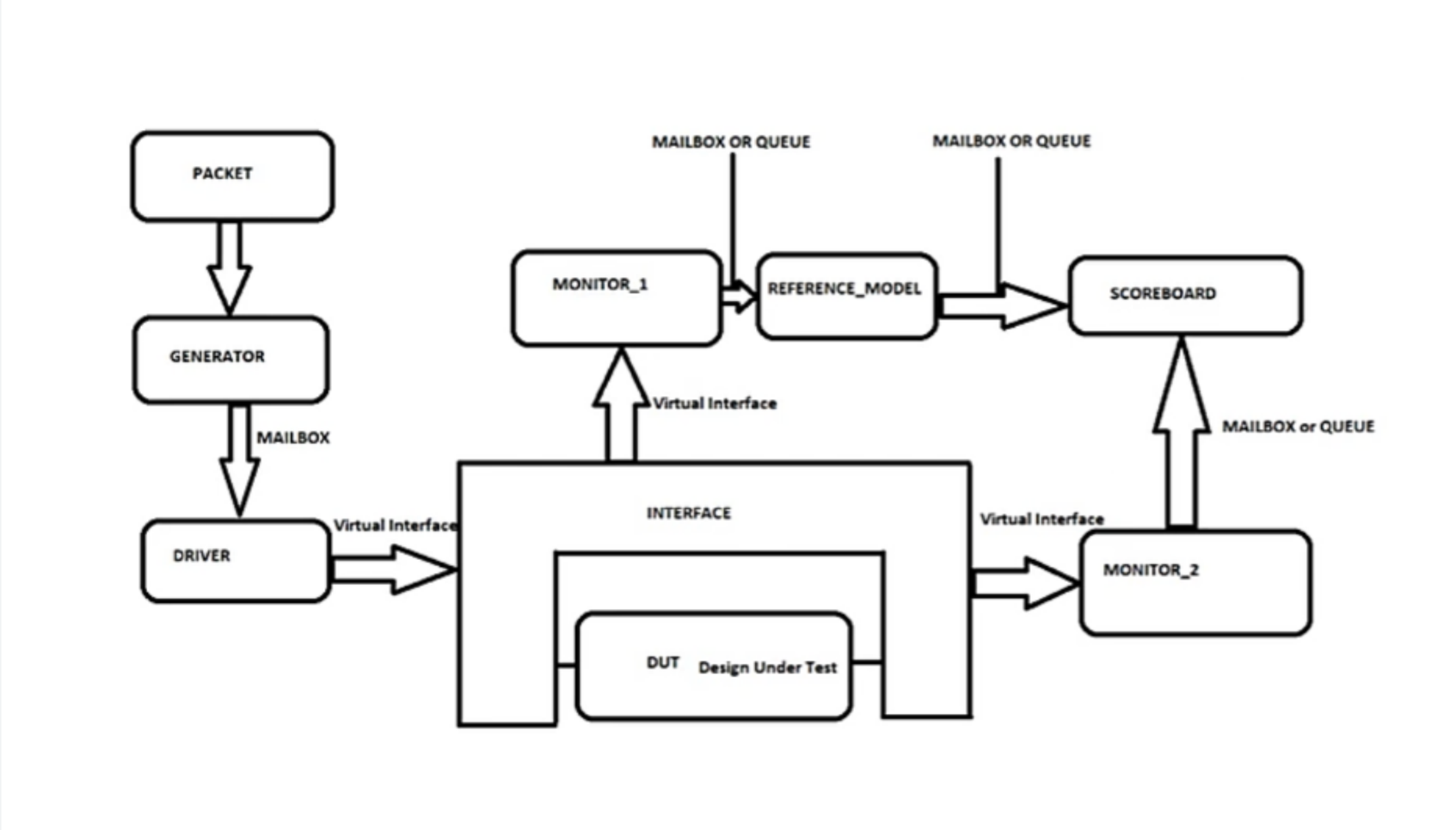
DESIGN FILE

design.sv

```
module adder(
  input clk,
  input reset,
  input [3:0] a,
  input [3:0] b,
  output reg [4:0] c
  );

  always @ (posedge clk, posedge reset)
  if ( reset )
    c <= 0;
  else
    c <= a+b;

endmodule
```



LAYERED TESTBENCH



PACKET/TRANSACTION

- We declare all the input and output signals available in the design of the packet class

testbench.sv | transaction.sv | generatc
scoreboard.sv | environment.sv | prog

```
class transaction;
  randc bit [3:0] a;
  rand bit [3:0] b;
           bit [4:0] c;
endclass
```

GENERATOR

- The generator will randomize the input signals. The randomized inputs are put into the mailbox using the put method.

testbench.sv | transaction.sv | generator.sv | driver.sv | interface.sv | monitor.sv | scoreboard.sv | environment.sv | program.sv | top.sv

```
`include "transaction.sv"
class generator;
  mailbox gen2driv;
  event ended;
  int repeat_count;
  transaction trans;

  function new ( mailbox gen2driv);
    this.gen2driv = gen2driv;
  endfunction

  task main;
    repeat(repeat_count)

    begin
      trans = new();
      if ( !trans.randomize()) $fatal ( "Gen randomization failed" ) ;
      gen2driv.put(trans);
    end
    -> ended;
  endtask

endclass
```

DRIVER

- The Driver drives the stimulus into the DUT through the interface.

testbench.sv | transaction.sv | generator.sv | driver.sv | interface.sv | monit
scoreboard.sv | environment.sv | program.sv | top.sv

```
`include "generator.sv"
class driver;
  virtual intf_adr vif;
  mailbox gen2driv;
  int no_trans;


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

  task reset;
    wait ( vif.reset );
    vif.a <= 0;
    vif.b <= 0;
    wait ( !vif.reset );
  endtask

  task main;
    forever
    begin
      transaction trans;
      gen2driv.get(trans);

      @ (posedge vif.clk)
      vif.a <= trans.a;
      vif.b <= trans.b;
      trans.c = vif.c;
      no_trans++;
    end
  endtask

endclass
```

INTERFACE

testbench.sv | transaction.sv | generator.sv | driver.sv | interface.sv | monitor.sv | scoreboard.sv | environment.sv | program.sv | top.sv

```
interface intf_adr( input logic clk, reset);
  logic [3:0] a;
  logic [3:0] b;
  logic [4:0] c;

  covergroup cg @ (posedge clk);
    cov_a : coverpoint a;
    cov_b : coverpoint b;
    cov_c : coverpoint c;
  endgroup

  cg cg_1 = new();


endinterface
```

MONITOR

- The DUT receives the stimulus and generates the outputs. The outputs are sampled/collected in the monitor

testbench.sv | transaction.sv | generator.sv | driver.sv | interface.sv | monitor.sv
scoreboard.sv | environment.sv | program.sv | top.sv

SV/Verilog Te

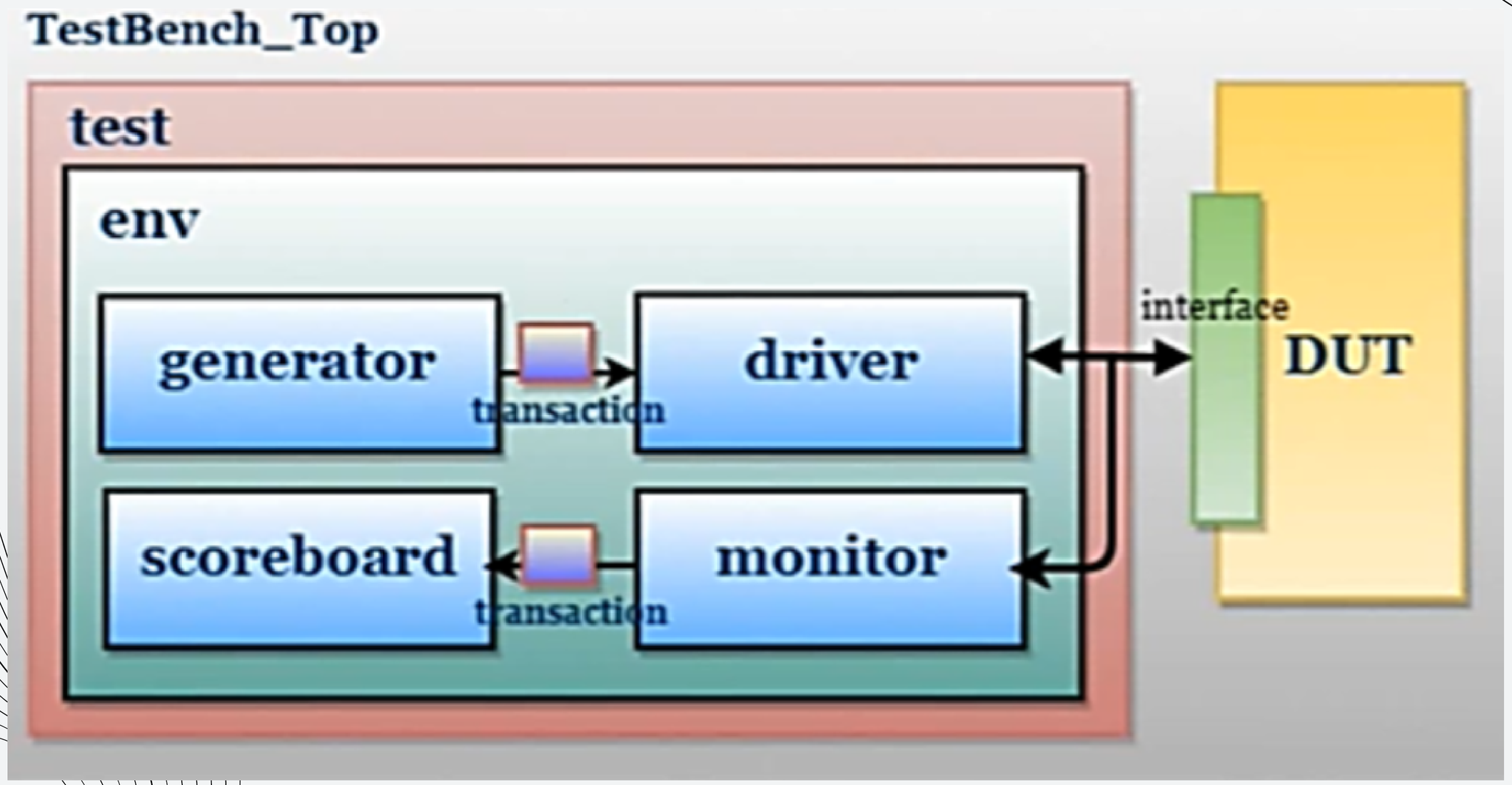
```
`include "driver.sv"
class monitor;
   virtual intf_adr vif;
   mailbox mon2scb;

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

   task main;
     forever
     begin
       transaction trans;
       trans = new();
       @ (posedge vif.clk)
       trans.a = vif.a;
       trans.b = vif.b;
       trans.c = vif.c;
       mon2scb.put(trans);
     end
   endtask

 endclass
```

SCOREBOARD

- The DUT receives the stimulus and generates the outputs. The outputs are sampled/collected in the monitor

testbench.sv | transaction.sv | generator.sv | driver.sv | interface.sv | monitor.sv | scoreboard.sv | environment.sv | program.sv | top.sv

```
`include "monitor.sv"
class scoreboard;

    virtual intf_adr vif;
    mailbox mon2scb;
    int no_trans;

    bit [4:0] GoldenFile_Var;

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

    task main;
    forever
    begin
        transaction trans;
        mon2scb.get(trans);
        GoldenFile_Var = trans.a + trans.b;
        $display (  " \t a = %0d, \t b = %0d, \t c = %0d ",  trans.a, trans.b, trans.c );
      assert ( (GoldenFile_Var) == trans.c) $display( " Success " ) ;
        else  $error ( " Gone wrong " );
        no_trans++;
    end
    endtask

endclass
```


TestBench_Top
test
env
generator
transaction
driver
interface
DUT
scoreboard
transaction
monitor

ENVIRONMENT

testbench.sv | transaction.sv | generator.sv | driver.sv | interface.sv | monitor.sv | scoreboard.sv | environment.sv | program.sv | top.sv

```
`include "scoreboard.sv"
class environment;
    virtual intf_adr vif;
    mailbox gen2driv, mon2scb;
    generator gen;
    driver driv ;
    monitor mon;
    scoreboard scb;


    function new ( virtual intf_adr vif );
        this.vif  =  vif;
        gen2driv = new();
        mon2scb = new();
        gen = new ( gen2driv );
        driv = new ( vif, gen2driv );
        mon = new ( vif, mon2scb );
        scb = new (mon2scb );
    endfunction

    task pre_run;
        driv.reset();
    endtask

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

    task post_run;
        wait ( gen.ended.triggered );
        wait ( gen.repeat_count == driv.no_trans );
        wait ( gen.repeat_count == scb.no_trans );
    endtask

    task main;
        pre_run();
        test();
        post_run();
    endtask

endclass
```

TEST

testbench.sv | transaction.sv | generator.sv | driver.sv | interfa
scoreboard.sv | environment.sv | program.sv | top.sv

```
`include "environment.sv"
program test ( intf_adr intf ) ;
    environment env;
    initial begin
      env = new ( intf ) ;
      env.gen.repeat_count = 400;
      env.main();
    end

endprogram
```

TOP

testbench.sv | transaction.sv | generator.sv | driver.sv | i
scoreboard.sv | environment.sv | program.sv | top.sv

```
module top;
    bit clk, reset;

    always #5 clk = ~clk;

    intf_adr intf_1(clk, reset);
    test test_1(intf_1);
    adder DUT (
      .clk (intf_1.clk),
      .reset (intf_1.reset),
      .a (intf_1.a),
      .b (intf_1.b),
      .c (intf_1.c)
      );

    initial begin
    reset = 1 ; clk = 0;
    #30; reset = 0;
    end

    initial
    begin
    $dumpfile ("dump.vcd");
    $dumpvars;
    end

endmodule
```

THANK YOU