# EXPERIMENT 6 CODE

```
//DESIGN
module full_adder(input a,b,cin,
output s,cy);
  assign {cy,s} = a+b+cin;
endmodule

//TRANSACTION
class transaction;
  rand bit a;
  rand bit b;
  rand bit cin;
  bit s;
  bit cy;
  function void display(string name);
      $display("@ %s component:
     inputs are %0d,%0d,%0d, and
     outputs: sum=%0d,
     cy=%0d",name,a,b,cin,s,cy);
  endfunction
endclass

//GENERATOR
class generator;
  transaction trans;
  mailbox gen2driv;
  function new(mailbox gen2driv);
    this.gen2driv = gen2driv;
  endfunction
  task main();
    trans = new();
    trans.randomize();
    trans.display("Generator");
    gen2driv.put(trans);
  endtask
endclass

//DRIVER
class driver;
  virtual intf vif;
  mailbox gen2driv;
  transaction trans;

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

  task main();
    gen2driv.get(trans);

    vif.a <= trans.a;
    vif.b <= trans.b;
    vif.cin <= trans.cin;
    trans.s = vif.s;
    trans.cy = vif.cy;
```

```
trans.display("Driver");
  endtask

endclass


//MONITOR
class monitor;
  virtual intf vif;
  mailbox mon2scb;
  transaction trans;

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

  task main();
    #3;
    trans = new();
    trans.a = vif.a;
    trans.b = vif.b;
    trans.cin = vif.cin;
    trans.s = vif.s;
    trans.cy = vif.cy;
    mon2scb.put(trans);
    trans.display("Monitor");
  endtask
endclass

//SCOREBOARD
class scoreboard;
  mailbox mon2scb;
  transaction trans;

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

  task main();
    mon2scb.get(trans);

    if({trans.cy,trans.s} ==
trans.a+trans.b+trans.cin)
      $display("Success");
    else
      $error("Wrong Result");
    trans.display("Scoreboard");
  endtask
endclass


//INTERFACE
interface intf();
  logic a,b,cin,s,cy;
endinterface


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

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

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

```
  endfunction

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

  task run;
    test();
    $finish;
  endtask
endclass

//TEST
`include "environment.sv"

program test(intf i_intf);
  environment env;

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

```
//TB_TOP
`include "interface.sv"
`include "test"
module tbench_top;
  intf i_intf();
  test t1(i_intf);
  full_adder f1(i_intf.a,i_intf.b,i_intf.cin,i_intf.s,i_intf.cy);
endmodule
```

## EXPERIMENT 7 CODE

```
//DESIGN
module full_adder(input [3:0] a,b,
output [3:0] sum, output cy);
  assign {cy,sum} = a+b;
endmodule

//TRANSACTION
class transaction;
  rand bit [3:0] a;
  rand bit [3:0] b;
  bit [3:0] sum;
  bit cy;

  function void display(string name);
    $display("@ %s component: inputs
are %0d,%0d, and outputs:
{cy,sum}=%0d",name,a,b,{cy,sum});
  endfunction
endclass
```

```
//GENERATOR
class generator;
  transaction trans;
  mailbox gen2driv;

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

  task main();
    trans = new();
    trans.randomize();
    trans.display("Generator");
    gen2driv.put(trans);
  endtask

endclass
```

```
//DRIVER
class driver;
  virtual intf vif;
  mailbox gen2driv;
  transaction trans;

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

  task main();
    gen2driv.get(trans);

    vif.a <= trans.a;
    vif.b <= trans.b;
    trans.sum = vif.sum;
    trans.cy = vif.cy;
    trans.display("Driver");
  endtask

endclass


//MONITOR
class monitor;
  virtual intf vif;
  mailbox mon2scb;
  transaction trans;

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

task main();
    #3;
    trans = new();
    trans.a = vif.a;
    trans.b = vif.b;
    trans.sum = vif.sum;
    trans.cy = vif.cy;
    mon2scb.put(trans);
    trans.display("Monitor");
  endtask
endclass

//SCOREBOARD
class scoreboard;
  mailbox mon2scb;
  transaction trans;
  function new(mailbox mon2scb);
    this.mon2scb = mon2scb;
  endfunction

  task main();
    mon2scb.get(trans);

    if({trans.cy,trans.sum} ==
trans.a+trans.b)
      $display("Success");
    else
      $error("Wrong Result");
    trans.display("Scoreboard");
  endtask
endclass

//INTERFACE
interface intf();
  logic [3:0] a,b,sum;
  logic cy;
endinterface
```

```
//ENVIRONMENT

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

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

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

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

  task run;
    test();
    #5;
    test();
    $finish;
  endtask
endclass

//TEST
`include "environment.sv"

program test(intf i_intf);
  environment env;

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


//TB_TOP
`include "interface.sv"     `include "test"
module tbench_top;
  intf i_intf();      test t1(i_intf);
  full_adder f1(i_intf.a,i_intf.b,i_intf.sum,i_intf.cy);
endmodule
```

# EXPERIMENT 8 CODE

```
//DESIGN
module full_adder(input [3:0]
a,b,input clk,reset, output reg
[6:0]sum);
  always@(posedge clk)begin
    if(reset) sum = 0;
    else sum = a+b;
  end
endmodule

//TRANSACTION
class transaction;
  rand bit [3:0] a;
  rand bit [3:0] b;
  bit [6:0] sum;

  function void display(string name);
      $display("@ %s component: inputs
      are %0d,%0d, and outputs:
      sum=%0d",name,a,b,sum);
  endfunction
endclass

//GENERATOR
class generator;
  transaction trans;
  mailbox gen2driv;
  int count;

event drv_done;

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

 task main();
    trans = new();
    repeat(count)begin
      trans.randomize();
      trans.display("Generator");
      gen2driv.put(trans);
      @(drv_done);
    end
  endtask
endclass

//DRIVER
class driver;
  virtual intf vif;
  mailbox gen2driv;
  transaction trans;

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

  task main();
    forever begin
    gen2driv.get(trans);
    vif.a <= trans.a;
    vif.b <= trans.b;
    trans.sum = vif.sum;
    trans.display("Driver");
    end
  endtask
endclass
```

```
//MONITOR
class monitor;
  virtual intf vif;
  mailbox mon2scb;
  transaction trans;
 event drv_done;

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

  task main();
    forever begin
    trans = new();
    @(posedge vif.clk);
    #1;
    trans.a = vif.a;
    trans.b = vif.b;

    trans.sum = vif.sum;
    mon2scb.put(trans);
    trans.display("Monitor");
    ->drv_done;
    end
  endtask
endclass
//SCOREBOARD
class scoreboard;
  mailbox mon2scb;
  transaction trans;
  int no_trans,x;
  function new(mailbox mon2scb);
    this.mon2scb = mon2scb;
  endfunction

task main();
    forever begin
    mon2scb.get(trans);
    no_trans++;
    trans.display("Scoreboard");
    if(trans.sum ==
trans.a+trans.b)begin
      x++;
      $display("Success \n\n");
    end
    else
      $error("Wrong Result");
    end endtask
endclass

//INTERFACE
interface intf(input logic clk,reset);
  logic [3:0] a,b; logic [6:0] sum;
endinterface

//ENVIRONMENT
`include "transaction.sv"
`include "generator.sv"
`include "driver.sv"
`include "monitor.sv"
`include "scoreboard.sv"
class environment;
  generator gen;
  driver drv;
  monitor mon;
  scoreboard scb;
  mailbox m1,m2;
  event dr;
```

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

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

  task run;
    gen.count = 6;
```

```
    gen.drv_done = dr;
    mon.drv_done = dr;
    test();
    wait(gen.count == scb.no_trans);
    $display("%0d ON %0d TRANSACTIONS
SUCCESSFULL",scb.x,scb.no_trans);
     $finish;
  endtask
endclass

//TEST
`include "environment.sv"
program test(intf i_intf);
  environment env;
  initial begin
    env = new(i_intf);
    env.run();
  end
endprogram
```

```
//TB_TOP
`include "interface.sv"
`include "test"
module tbench_top;
  bit clk,reset;
  always #5 clk = !clk;
  intf i_intf(clk,reset);
  test t1(i_intf);
```

```
full_adderf1(i_intf.a,i_intf.b,i_intf.
clk,i_intf.reset,i_intf.sum);
  initial begin
    $dumpfile("dump.vcd"); $dumpvars;
    reset = 1; #5; reset = 0;
  end
endmodule
```

# EXPERIMENT 9 CODE

```
//constraint_1
class packet;
  rand bit [3:0] addr;
  constraint addr_range { addr > 5; }
endclass

module const_blocks;
  initial begin
    packet pkt;
    pkt = new();
    repeat(10) begin
      pkt.randomize();
      $display("\taddr = %0d",pkt.addr);
    end
  end
endmodule

//constraint_2
class packet;
  rand bit [3:0] addr;
  constraint addr_range;
endclass

constraint packet :: addr_range { addr > 5; }

module const_blocks;
  initial begin
    packet pkt;
    pkt = new();
    repeat(10) begin
      pkt.randomize();
      $display("\taddr = %0d",pkt.addr);
    end
  end
endmodule
```

```
//constraint_3
class packet;
  rand bit [3:0] addr;
  rand bit [3:0] start;
  rand bit [3:0] stop;
  constraint addr_range;
endclass

constraint packet :: addr_range { addr inside {[start:stop]}; }

module const_blocks;
  initial begin
    packet pkt;
    pkt = new();
    repeat(10) begin
      pkt.randomize();
      $display("start is %0d, stop is %0d",pkt.start, pkt.stop);
      $display("\taddr = %0d",pkt.addr);
    end
  end
endmodule

//constraint_4
class packet;
  rand bit [3:0] addr;
  rand bit [3:0] start;
  rand bit [3:0] stop;
  constraint addr_range;
endclass

constraint packet :: addr_range {! (addr inside {[start:stop]});
                                  start < stop;}

module const_blocks;
  initial begin
    packet pkt;
    pkt = new();
    repeat(10) begin
      pkt.randomize();
      $display("start is %0d, stop is %0d",pkt.start, pkt.stop);
      $display("\taddr = %0d",pkt.addr);
    end
  end
endmodule
```