DEPARTMENT OF ELECTRONICS AND COMMUNICATION

B M S COLLEGE OF ENGINEERING

(AUTONOMOUS COLLEGE UNDER VTU, BELGAUM)

BANGALORE – 560019

2020-21

AAT Report

on

SYSTEM VERILOG and VERFICATION

(19EC6PE3SV)

BY

Bhargav S

1BM18EC025

Course Instructor

Dr. Kiran Bailey

Assistant Professor, Dept of ECE

Table of Contents

1. About the Tools Used

- **Xilinx Vivado 2020.2 Webpack**

  Vivado Design Suite is a software suite produced by Xilinx for synthesis and analysis of HDL designs, superseding Xilinx ISE with additional features for system on a chip development and high-level synthesis. Vivado represents a ground-up rewrite and re-thinking of the entire design flow (compared to ISE).

  Vivado was introduced in April 2012 and is an integrated design environment (IDE) with system-to-IC level tools built on a shared scalable data model and a common debug environment. Vivado includes electronic system level (ESL) design tools for synthesizing and verifying C-based algorithmic IP; standards-based packaging of both algorithmic and RTL IP for reuse; standards-based IP stitching and systems integration of all types of system building blocks; and the verification of blocks and systems.

  The Vivado High-Level Synthesis compiler enables C, C++ and SystemC programs to be directly targeted into Xilinx devices without the need to manually create RTL. Vivado HLS is widely reviewed to increase developer productivity, and is confirmed to support C++ classes, templates, functions and operator overloading.[18][16] Vivado 2014.1 introduced support for automatically converting OpenCL kernels to IP for Xilinx devices. OpenCL kernels are programs that execute across various CPU, GPU and FPGA platforms.

  The Vivado Simulator is a component of the Vivado Design Suite. It is a compiled-language simulator that supports mixed-language, Tcl scripts, encrypted IP and enhanced verification.

  The Vivado IP Integrator allows engineers to quickly integrate and configure IP from the large Xilinx IP library. The Integrator is also tuned for MathWorks Simulink designs built with Xilinx's System Generator and Vivado High-Level Synthesis.

  The Vivado Tcl Store is a scripting system for developing add-ons to Vivado, and can be used to add and modify Vivado’ s capabilities. Tcl is the scripting language on which Vivado itself is based.[19] All of Vivado’ s underlying functions can be invoked and controlled via Tcl scripts.

Vivado Webpack is a unlicensed version of the complete design suite and allows users to run behavioral simulation and synthesis.

• EDA Playground

EDA Playground gives immediate hands-on exposure to simulating and synthesizing System Verilog, Verilog, VHDL, C++/SystemC, and other HDLs. All you need is a web browser.

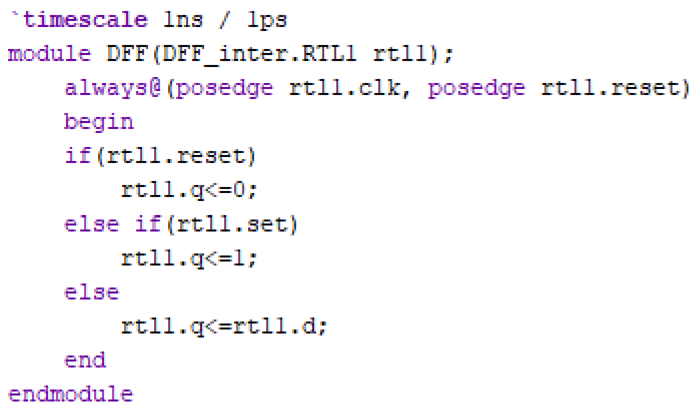
- With a simple click, run your code and see console output in real time.
- View waves for your simulation using EPWave browser-based wave viewer.
- Save your code snippets (“Playgrounds”).
- Share your code and simulation results with a web link. Perfect for web forum discussions or emails. Great for asking questions or sharing your knowledge.
- Quickly try something out
    - Try out a language feature with a small example.
    - Try out a library that you’re thinking of using.

2. AAT 1

2.1. LAB 1

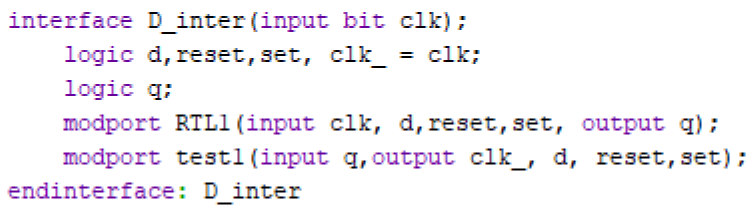
Design a D flip flop (.sv file) with asynchronous reset and synchronous set and write a test bench for it using interface block and mod ports.

- **DFF.sv**



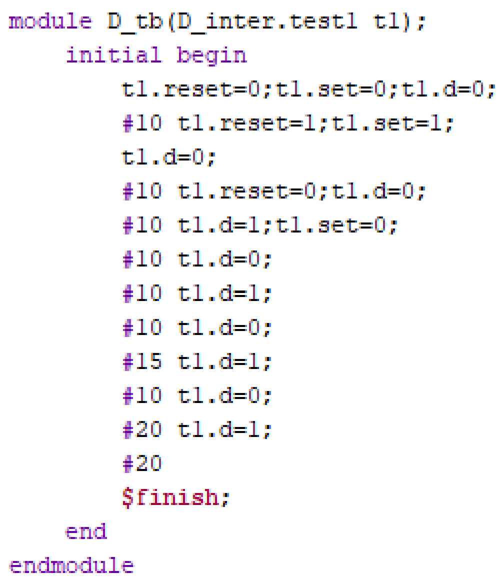
```
`timescale 1ns / 1ps
module DFF(DFF_inter.RTL1 rtl1);
    always@(posedge rtl1.clk, posedge rtl1.reset)
    begin
    if(rtl1.reset)
        rtl1.q<=0;
    else if(rtl1.set)
        rtl1.q<=1;
    else
        rtl1.q<=rtl1.d;
    end
endmodule
```

- **D_inter.sv**



```
interface D_inter(input bit clk);
    logic d,reset,set, clk_ = clk;
    logic q;
    modport RTL1(input clk, d,reset,set, output q);
    modport test1(input q,output clk_, d, reset,set);
endinterface: D_inter
```

- **D_tb.sv**

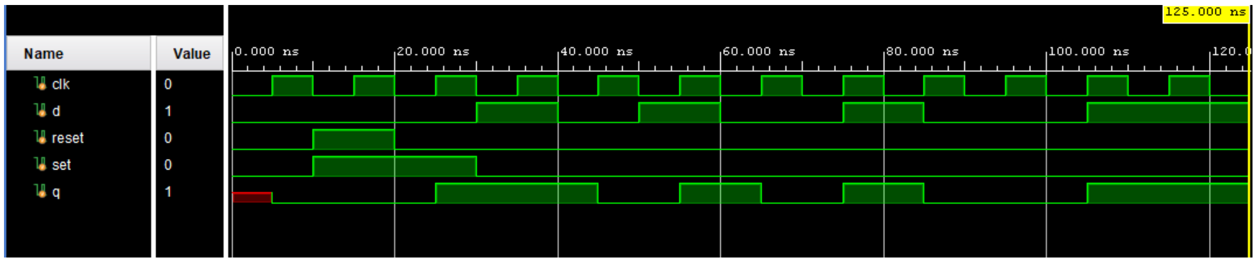


```
module D_tb(D_inter.test1 t1);
    initial begin
        t1.reset=0;t1.set=0;t1.d=0;
        #10 t1.reset=1;t1.set=1;
        t1.d=0;
        #10 t1.reset=0;t1.d=0;
        #10 t1.d=1;t1.set=0;
        #10 t1.d=0;
        #10 t1.d=1;
        #10 t1.d=0;
        #15 t1.d=1;
        #10 t1.d=0;
        #20 t1.d=1;
        #20
        $finish;
    end
endmodule
```

- **D_top.sv**

```
`timescale 1ns / 1ps
module D_top;
    bit clk;
    D_inter i1(clk);
    D_FF Dut(i1);
    D_tb T1(i1);
    always #5 clk=~clk;
    initial
    begin
        $dumpfile("top.vcd");
        $dumpvars;
        $monitor("time=%d, clk=%b, reset=%b,set=%b, d=%b, q=%b",
        $time, clk, i1.reset, i1.set,i1.d, i1.q);
    end
endmodule
```

- **Simulation Waveform**



2.2. <u>LAB 2</u>

Design a D flip flop (.v file) with asynchronous reset and synchronous set and write a test bench for it using interface block and mod ports.

- **DFF.v**

```
`timescale 1ns / 1ps
module DFF(input clk, d, reset,set, output reg q);
always@(posedge clk, posedge reset)
    begin
    if(reset)
        q<=0;
    else if (set)
        q<=1;
    else
        q<=d;
    end
endmodule
```

- **DFF_inter.sv**

```
interface DFF_inter(input bit clk);
    logic d,reset,set, clk_=clk;
    logic q;
    modport test1(input q,output clk_, d, reset,set);
    endinterface: DFF_inter
```

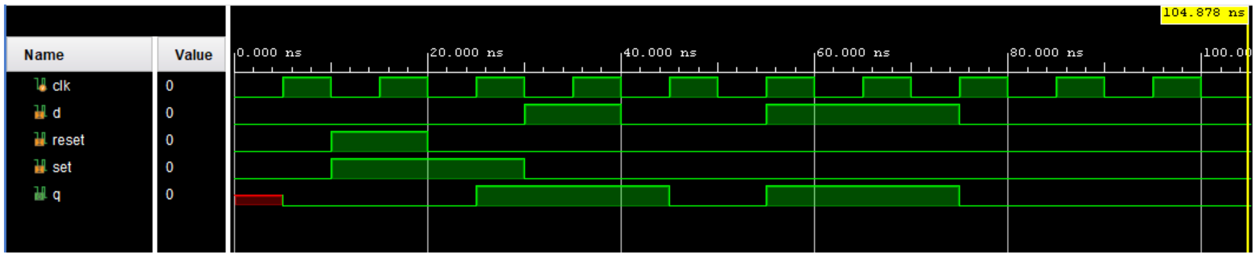
- **DFF_tb.sv**

```
module DFF_tb(DFF_inter.test1 t1);
    initial begin
        t1.reset=0;t1.set=0;
        t1.d=0;
        #10 t1.reset=1;t1.set=1;
        t1.d=0;
        #10 t1.reset=0; t1.d=0;
        #10 t1.d=1;t1.set=0;
        #10 t1.d=0;
        #10 t1.d=1;
        #10 t1.d=0;
        #20
    $finish;
    end
endmodule
```

- **top.sv**

```
`timescale 1ns / 1ps
module top;
    bit clk;
    DFF_inter i1(clk);
    DFF Dut(.clk(clk),.d(i1.d),.reset(i1.reset),.set(i1.set),.q(i1.q));
    DFF_tb T1(i1);
    always #5 clk=~clk;
    initial
    begin
        $dumpfile("top.vcd");
        $dumpvars;
        $monitor("time=%d, clk=%b, reset=%b,set=%b, d=%b, q=%b", $time, clk, i1.reset, i1.set,
        i1.d, i1.q);
    end
endmodule
```

- **Simulation Waveform**



2.3. LAB 4

Program 4.1

Program to demonstrate class instantiation.

- **4.1.sv**

```
class sv_class;

    //class properties
    int x;

    //method-1
    task set(int i);
        x = i;
    endtask

    //method-2
    function int get();
        return x;
    endfunction

endclass


program sv_class_ex;

    sv_class class_1; // Create Handle

    initial begin
        sv_class class_2 = new(); // Creating handle and Object
        class_1 = new(); // Creating object for the Handle

        class_1.set(10);
        class_2.set(20);

        $display("\tclass_1 :: Value of x = %0d", class_1.get());
        $display("\tclass_2 :: Value of x = %0d", class_2.get());
    end

endprogram
```

- **Simulation Output**

```
# KERNEL:       class_1 :: Value of x = 10
# KERNEL:       class_2 :: Value of x = 20
```

Program 4.2

Program to demonstrate constructor for a class.

- **4.2.sv**

```
class packet;

    //class properties
    bit [31:0] addr;
    bit [31:0] data;
    bit write;
    string pkt_type;

    //constructor
    function new(bit [31:0] addr, data, bit write, string pkt_type);
        this.addr = addr;
        this.data = data;
        this.write = write;
        this.pkt_type = pkt_type;
    endfunction

    /*alternate method
        this.addr = addr;
        this.data = data;
        this.write = write;
        this.pkt_type = pkt_type;
    */

    //method to display class prperties

    function void display();
        $display("---------------------------------------------------------");
        $display("\t addr = %0h",addr);
        $display("\t data = %0h",data);
        $display("\t write = %0h",write);
        $display("\t pkt_type = %0s",pkt_type);
        $display("---------------------------------------------------------");
    endfunction

endclass

module sv_constructor;

    packet pkt;

    initial begin
        pkt = new(32'h10,32'hFF,1,"GOOD_PKT");
        pkt.display();
    end

endmodule
```

- **Simulation Output**

```
# KERNEL: ---------------------------------------------------------
# KERNEL:        addr = 10
# KERNEL:        data = ff
# KERNEL:        write = 1
# KERNEL:        pkt_type = GOOD_PKT
# KERNEL: ---------------------------------------------------------
```

Program 4.3

Program to demonstrate the use of $random and $urandom.

- **4.3.sv**

```
program main1;

    initial begin
        int a;

      repeat(10) begin
            #10 a=$random; //or #10 a=$urandom_range(0,70);
            $display("a=%d\n", a);
        end
    end

endprogram
```

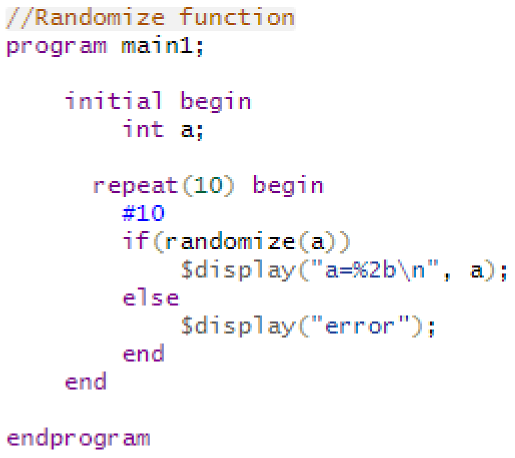
- **Simulation Output**

```
# KERNEL: a=  303379748
# KERNEL:
# KERNEL: a=-1064739199
# KERNEL:
# KERNEL: a=-2071669239
# KERNEL:
# KERNEL: a=-1309649309
# KERNEL:
# KERNEL: a=  112818957
# KERNEL:
# KERNEL: a= 1189058957
# KERNEL:
# KERNEL: a=-1295874971
# KERNEL:
# KERNEL: a=-1992863214
# KERNEL:
# KERNEL: a=   15983361
# KERNEL:
# KERNEL: a=  114806029
```

Program 4.4

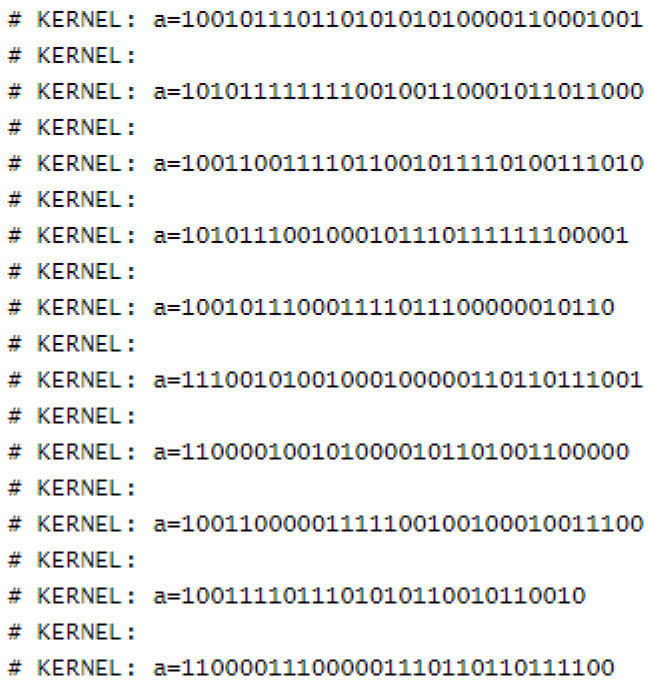
Program to demonstrate Randomization function or method.

- **4.4.sv**



```
//Randomize function
program main1;

    initial begin
        int a;

      repeat(10) begin
        #10
        if(randomize(a))
            $display("a=%2b\n", a);
        else
            $display("error");
        end
    end

endprogram
```

- **Simulation Output**



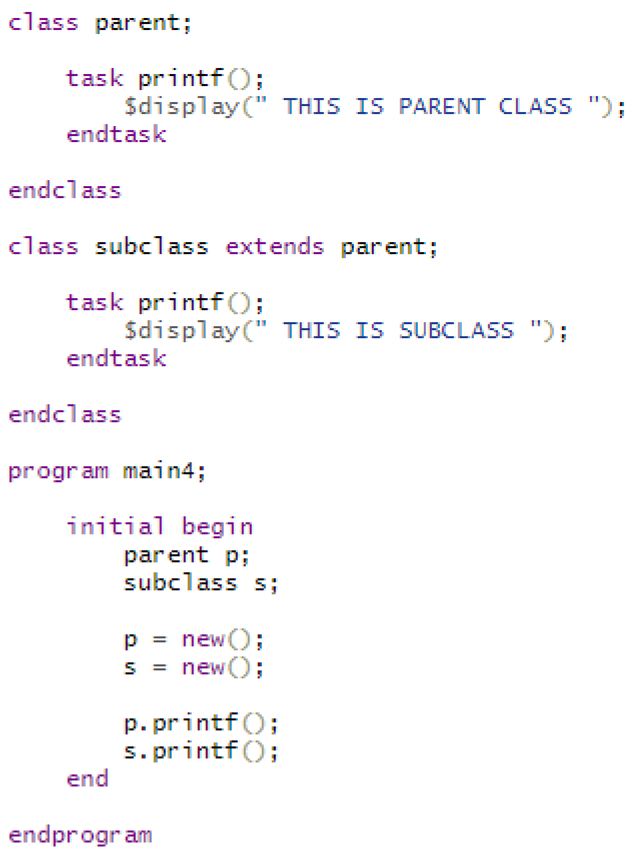
```
# KERNEL: a=10010111011010101010000110001001
# KERNEL:
# KERNEL: a=10101111111100100110001011011000
# KERNEL:
# KERNEL: a=10011001111011001011110100111010
# KERNEL:
# KERNEL: a=1010111001000101110111111100001
# KERNEL:
# KERNEL: a=1001011100011110111000000010110
# KERNEL:
# KERNEL: a=11100101001000100000110110111001
# KERNEL:
# KERNEL: a=1100001001010000101101001100000
# KERNEL:
# KERNEL: a=10011000001111100100100010011100
# KERNEL:
# KERNEL: a=100111101110101011001011001 0
# KERNEL:
# KERNEL: a=1100001110000011101101101111 00
```

2.4. <u>Lab 5</u>

<u>Program 5.1</u>

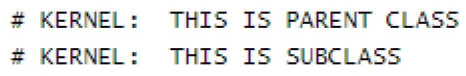
Program to demonstrate Inheritance and Overriding.

- **5.1.sv**



```
class parent;

    task printf();
        $display(" THIS IS PARENT CLASS ");
    endtask

endclass

class subclass extends parent;

    task printf();
        $display(" THIS IS SUBCLASS ");
    endtask

endclass

program main4;

    initial begin
        parent p;
        subclass s;

        p = new();
        s = new();

        p.printf();
        s.printf();
    end

endprogram
```

- **Simulation Output**



```
# KERNEL:  THIS IS PARENT CLASS
# KERNEL:  THIS IS SUBCLASS
```

<u>Program 5.2</u>

Program to demonstrate Super class.

- **5.2.sv**

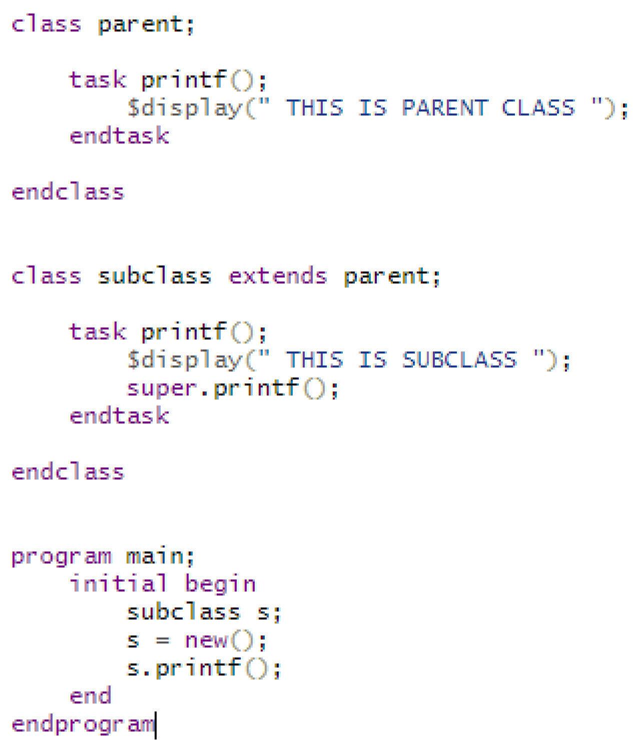
```
class parent;

    task printf();
        $display(" THIS IS PARENT CLASS ");
    endtask

endclass


class subclass extends parent;

    task printf();
        super.printf();
    endtask

endclass


program main;
    initial begin
        subclass s;
        s = new();
        s.printf();
    end
endprogram
```

- **Simulation Output**

```
# KERNEL:  THIS IS PARENT CLASS
```

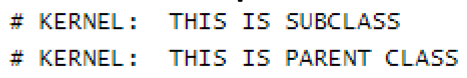
<u>Program 5.3</u>
Program to demonstrate addition of new functionality to a sub class.

- **5.3.sv**



```
class parent;

    task printf();
        $display(" THIS IS PARENT CLASS ");
    endtask

endclass


class subclass extends parent;

    task printf();
        $display(" THIS IS SUBCLASS ");
        super.printf();
    endtask

endclass


program main;
    initial begin
        subclass s;
        s = new();
        s.printf();
    end
endprogram
```

- **Simulation Output**



```
# KERNEL:  THIS IS SUBCLASS
# KERNEL:  THIS IS PARENT CLASS
```

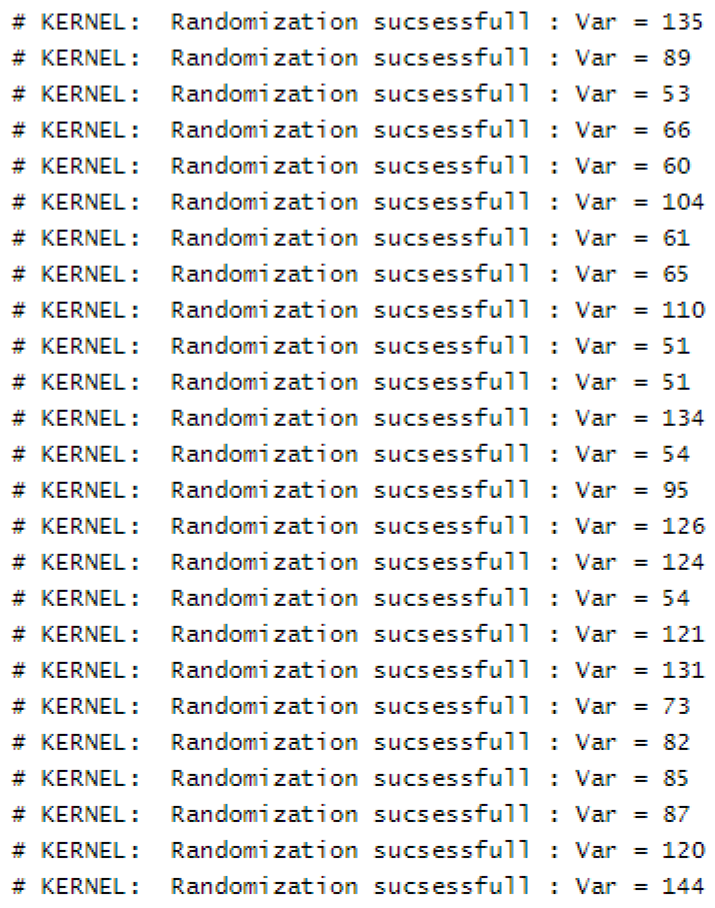
<u>Program 5.4</u>

Program to demonstrate_override of the existing constraint and replace with new constraint

- **5.4.sv**

```
class Base;

    rand integer Var;
    constraint range { Var < 100 ; Var > 0 ;}

endclass


class baseA extends Base;
    // Overrighting the Base class constraints.
    constraint range { Var < 150 ; Var > 50 ;}
endclass


program main;
    baseA obj1;

    initial begin
        obj1= new();
      for(int i=0 ; i < 25 ; i++)
            if(obj1.randomize())
                $display(" Randomization sucsessfull : Var = %0d ",obj1.Var);
            else
                $display("Randomization failed");
    end

endprogram
```

- **Simulation Output**



```
# KERNEL:  Randomization sucsessfull : Var = 135
# KERNEL:  Randomization sucsessfull : Var = 89
# KERNEL:  Randomization sucsessfull : Var = 53
# KERNEL:  Randomization sucsessfull : Var = 66
# KERNEL:  Randomization sucsessfull : Var = 60
# KERNEL:  Randomization sucsessfull : Var = 104
# KERNEL:  Randomization sucsessfull : Var = 61
# KERNEL:  Randomization sucsessfull : Var = 65
# KERNEL:  Randomization sucsessfull : Var = 110
# KERNEL:  Randomization sucsessfull : Var = 51
# KERNEL:  Randomization sucsessfull : Var = 51
# KERNEL:  Randomization sucsessfull : Var = 134
# KERNEL:  Randomization sucsessfull : Var = 54
# KERNEL:  Randomization sucsessfull : Var = 95
# KERNEL:  Randomization sucsessfull : Var = 126
# KERNEL:  Randomization sucsessfull : Var = 124
# KERNEL:  Randomization sucsessfull : Var = 54
# KERNEL:  Randomization sucsessfull : Var = 121
# KERNEL:  Randomization sucsessfull : Var = 131
# KERNEL:  Randomization sucsessfull : Var = 73
# KERNEL:  Randomization sucsessfull : Var = 82
# KERNEL:  Randomization sucsessfull : Var = 85
# KERNEL:  Randomization sucsessfull : Var = 87
# KERNEL:  Randomization sucsessfull : Var = 120
# KERNEL:  Randomization sucsessfull : Var = 144
```

<u>Program 5.5</u>
Program to demonstrate over riding of data members.

- **5.5.sv**

```
class base;

    int N = 3;

    function int get_N();
        return N;
    endfunction

endclass


class ext extends base;

    int N = 4;

    function int get_N();
        return N;
    endfunction

    function int get_N1();
        return super.N;
    endfunction

endclass

program main;
  initial
  begin
    ext e = new();
    base b = e; // Note same object!

    $display(b.get_N()); // "3"
    $display(e.get_N()); // "4"
    $display(e.get_N1()); // "3" - super.N
  end
endprogram
```

- **Simulation Output**

```
# KERNEL:          3
# KERNEL:          4
# KERNEL:          3
```

<u>Program 5.6</u>
Program to demonstrate Polymorphism.

- **5.6.1.sv**

```
class A;

    task disp();
        $display(" This is class A ");
    endtask

endclass


class EA extends A;

    task disp ();
        $display(" This is Extended class A ");
    endtask

endclass


program main;

    EA my_ea;
    A my_a;

    initial begin
        my_a = new();
        my_a.disp();
        my_ea = new();
        my_a = my_ea;
        my_a.disp();
    end

endprogram
```

- **Simulation Output**

```
# KERNEL:  This is class A
# KERNEL:  This is class A
```

- **5.6.2.sv**

```
class A;

    virtual task disp();
        $display(" This is class A ");
    endtask

endclass


class EA extends A;

    task disp();
        $display(" This is Extended class A ");
    endtask

endclass


program main;

    EA my_ea;
    A my_a;

    initial begin
        my_a = new();
        my_a.disp();
        my_ea = new();
        my_a = my_ea;
        my_a.disp();
    end

endprogram
```

- **Simulation Output**

```
# KERNEL:  This is class A
# KERNEL:  This is Extended class A
```

<u>Program 5.7</u>

Program to demonstrate static class properties.

- **5.7.1.sv**

```
class A;

    static int i;
  task set(int x);
      i = x;
    endtask

endclass


program main ;

    A obj_1;
    A obj_2;

    initial begin
        obj_1 = new();
        obj_2 = new();
        obj_1.set(123);
      $display("In obj_1, i = %d",obj_1.i);
      $display("In obj_2, i = %d",obj_2.i);

    end

endprogram
```

- **Simulation Output**

```
# KERNEL: In obj_1, i =          123
# KERNEL: In obj_2, i =          123
```

- **5.7.2.sv**

```
class A;

    static int i;
  task set(int x);
      i = x;
    endtask

endclass


program main;

    A obj_1;

    initial begin
        obj_1 = new();
        obj_1.set(123);
        $display(A::i);
    end

endprogram
```

- **Simulation Output**

```
# KERNEL:         123
```

Program 5.8
Program to demonstrate static methods.

- **5.8.sv**

```
class A;

    static task incr();
        int j; //automatic variable
        j++;
        $display("J is %d",j);
    endtask

endclass


program main;

    A obj_1;
    A obj_2;

    initial begin
        $display("Static task - static task with automatic variables");
        obj_1 = new();
        obj_2 = new();
        obj_1.incr();
        obj_2.incr();
        obj_1.incr();
        obj_2.incr();
        obj_1.incr();
        $display("Static task - Each call to task will create a separate copy of
'j' and increment it");
    end

endprogram
```

- **Simulation Output**

```
# KERNEL: Static task - static task with automatic variables
# KERNEL: J is           1
# KERNEL: J is           1
# KERNEL: J is           1
# KERNEL: J is           1
# KERNEL: J is           1
# KERNEL: Static task - Each call to task will create a separate copy of 'j' and increment it
```

Program 5.9
Program to demonstrate static lifetime method.

- **5.9.sv**

```
class A;

    task static incr();
        int i; //static variable
        $display(" i is %d ",i);
        i++;
    endtask

endclass


program main;

    A a;
    A b;

    initial begin
        $display("Static lifetime - non static task with static variables");
        a = new();
        b = new();
        a.incr();
        b.incr();
        a.incr();
        b.incr();
        a.incr();
      $display("Static lifetime - Each call to task will use a single value of 'i'
and increment it");
    end

endprogram
```

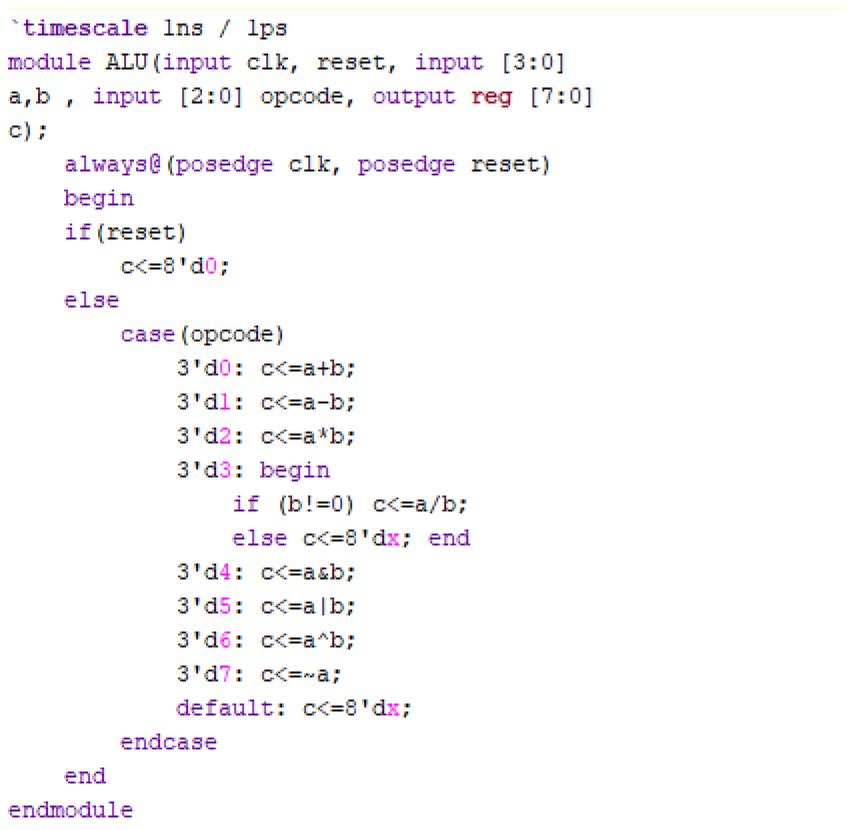
- **Simulation Output**

```
# KERNEL: Static lifetime - non static task with static variables
# KERNEL:  i is          0
# KERNEL:  i is          1
# KERNEL:  i is          2
# KERNEL:  i is          3
# KERNEL:  i is          4
# KERNEL: Static lifetime - Each call to task will use a single value of 'i' and increment it
```

2.5. <u>LAB 6</u>

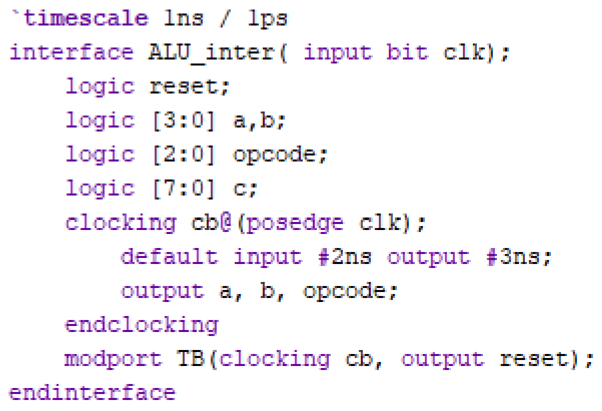
Design a 4-bit ALU that has 3-bit opcode to decide the operation. Test the module using interface and clocking blocks, mod ports and randomize the test stimulus along with constraints.

- **ALU.v**



```
`timescale 1ns / 1ps
module ALU(input clk, reset, input [3:0]
a,b , input [2:0] opcode, output reg [7:0]
c);
    always@(posedge clk, posedge reset)
    begin
    if(reset)
        c<=8'd0;
    else
        case(opcode)
            3'd0: c<=a+b;
            3'd1: c<=a-b;
            3'd2: c<=a*b;
            3'd3: begin
                if (b!=0) c<=a/b;
                else c<=8'dx; end
            3'd4: c<=a&b;
            3'd5: c<=a|b;
            3'd6: c<=a^b;
            3'd7: c<=~a;
            default: c<=8'dx;
        endcase
    end
endmodule
```

- **ALU_inter.sv**



```
`timescale 1ns / 1ps
interface ALU_inter( input bit clk);
    logic reset;
    logic [3:0] a,b;
    logic [2:0] opcode;
    logic [7:0] c;
    clocking cb@(posedge clk);
        default input #2ns output #3ns;
        output a, b, opcode;
    endclocking
    modport TB(clocking cb, output reset);
endinterface
```

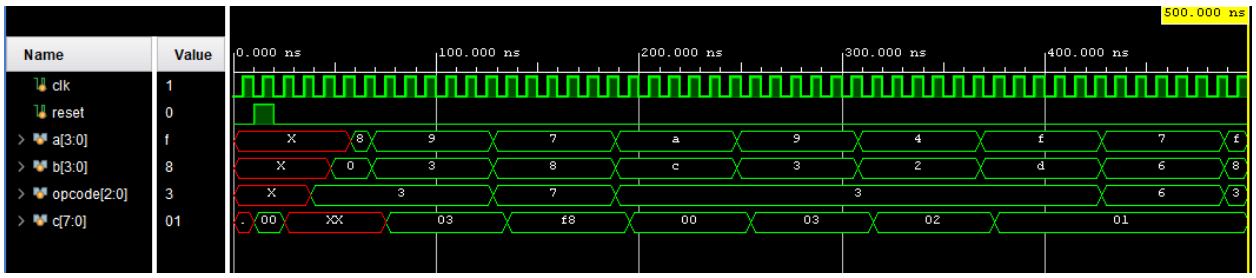
- **ALU_tb.sv**

```
`timescale 1ns / 1ps
program ALU_tb(ALU_inter.TB T1);
    class random;
    rand bit [3:0] a, b;
    rand bit [2:0] opcode;
    constraint C1 {opcode inside {[5:7],3};}
    constraint C2 {a inside {[1:10],15};}
    constraint C3 {b>0;b<14;}
endclass
random input_data=new();
    initial
    begin
        T1.reset<=0;
        #10 T1.reset<=1;
        #10 T1.reset<=0;
        #10 T1.cb.opcode<=3'd3;
        #10 T1.cb.b<=4'd0;
        #10 T1.cb.a<=4'd8;end
        initial begin
            repeat(100)
            begin
            if(input_data.randomize())
            begin
           #60 T1.cb.a<= input_data.a;
           T1.cb.b<= input_data.b;
           T1.cb.opcode<= input_data.opcode;
       end
   else
       $display("Data not randomized");
   end
   end
endprogram
```

- **ALU_top.sv**

```
module top;
    bit clk;
    ALU_inter i1(clk);
    ALU Dut( .clk(clk), .reset(i1.reset), .a(i1.a), .b(i1.b), .c(i1.c), .opcode(i1.opcode));
    ALU_tb T1(i1);
    always #5 clk=~clk;
    initial
    begin
        $dumpfile("top.vcd");
        $dumpvars;
        $monitor("time=%d, clk=%b, reset=%b, a=%b,b=%b,opcode=%b,c=%b",
            $time, clk, i1.reset,i1.a,i1.b,i1.opcode, i1.c);
        #500
        $finish;
    end
endmodule
```

- **Simulation Waveforms**



2.6. LAB 7

Design a 4-bit ALU that has 3-bit opcode to decide the operation. Test the module using interface and clocking blocks, mod ports, randomize the test stimulus along with constraints and drive the stimulus using driver and transactor class.

- **ALU.v**

```
`timescale 1ns / 1ps
module ALU(input clk, reset, input [3:0]
a,b , input [2:0] opcode, output reg [7:0]
c);
    always@(posedge clk, posedge reset)
    begin
    if(reset)
        c<=8'd0;
    else
        case(opcode)
            3'd0: c<=a+b;
            3'd1: c<=a-b;
            3'd2: c<=a*b;
            3'd3: begin
                if (b!=0) c<=a/b;
                else c<=8'dx; end
            3'd4: c<=a&b;
            3'd5: c<=a|b;
            3'd6: c<=a^b;
            3'd7: c<=~a;
            default: c<=8'dx;
        endcase
    end
endmodule
```

- **ALU_inter.sv**

```
interface ALU_inter( input bit clk);
    logic reset;
    logic [3:0] a,b;
    logic [2:0] opcode;
    logic [7:0] c;
    clocking cb@(posedge clk);
        default input #2ns output #3ns;
        output a, b, opcode;
    endclocking
    modport TB(clocking cb, output reset);
endinterface
```

- **ALU_tb.sv**

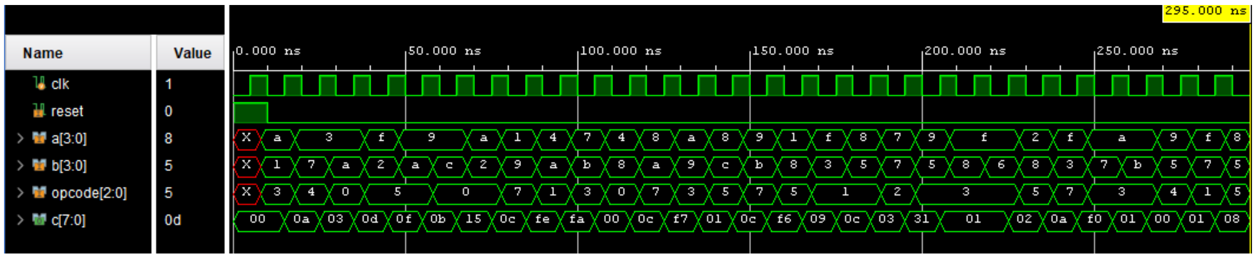
```
class transactor;
    rand bit [3:0] a,b;
    rand bit [2:0] opcode;
    constraint C1 {opcode inside {[0:5],7};}
    constraint C2 {a inside {[1:10],15};}
    constraint C3 {b>0;b<14;}
endclass

class driver;
    transactor tx;
    virtual ALU_inter.TB il;
    function new(virtual ALU_inter.TB il_new);
        this.il=il_new;
        this.tx=new();
        $display ("%d : Driver: new method created", $time);
    endfunction
    task drive_data();
        begin
        repeat(30) @( il.cb)
        begin
        if((tx.randomize()))
        begin
            il.cb.a<=tx.a;
            il.cb.b<=tx.b;
```

```
                il.cb.opcode<=tx.opcode;
            end
            else
                $display("%d : Randomization error", $time);
            end
            end
        endtask
    endclass

    program ALU_tb(ALU_inter.TB itb);
        driver drv=new(itb);
        initial
        begin
        itb.reset<=1;
        #10 itb.reset<=0;
        end
        initial
        begin
        drv.drive_data();
        end
    endprogram
```

- **ALU_top.sv**

```
module top;
    bit clk;
    ALU_inter Ia (clk);
    ALU DUT(clk, Ia.reset, Ia.a, Ia.b , Ia.opcode, Ia.c);
    ALU_tb TB1 (Ia.TB);
    always #5 clk=~clk;
    initial
    begin
        $dumpfile("top.vcd");
        $dumpvars;
        $monitor("time=%d, clk=%b, reset=%b, a=%b,b=%b,opcode=%b, c=%b",
                        $time, clk, Ia.reset,Ia.a,Ia.b,Ia.opcode, Ia.c);
    #500 $finish;
    end
endmodule
```

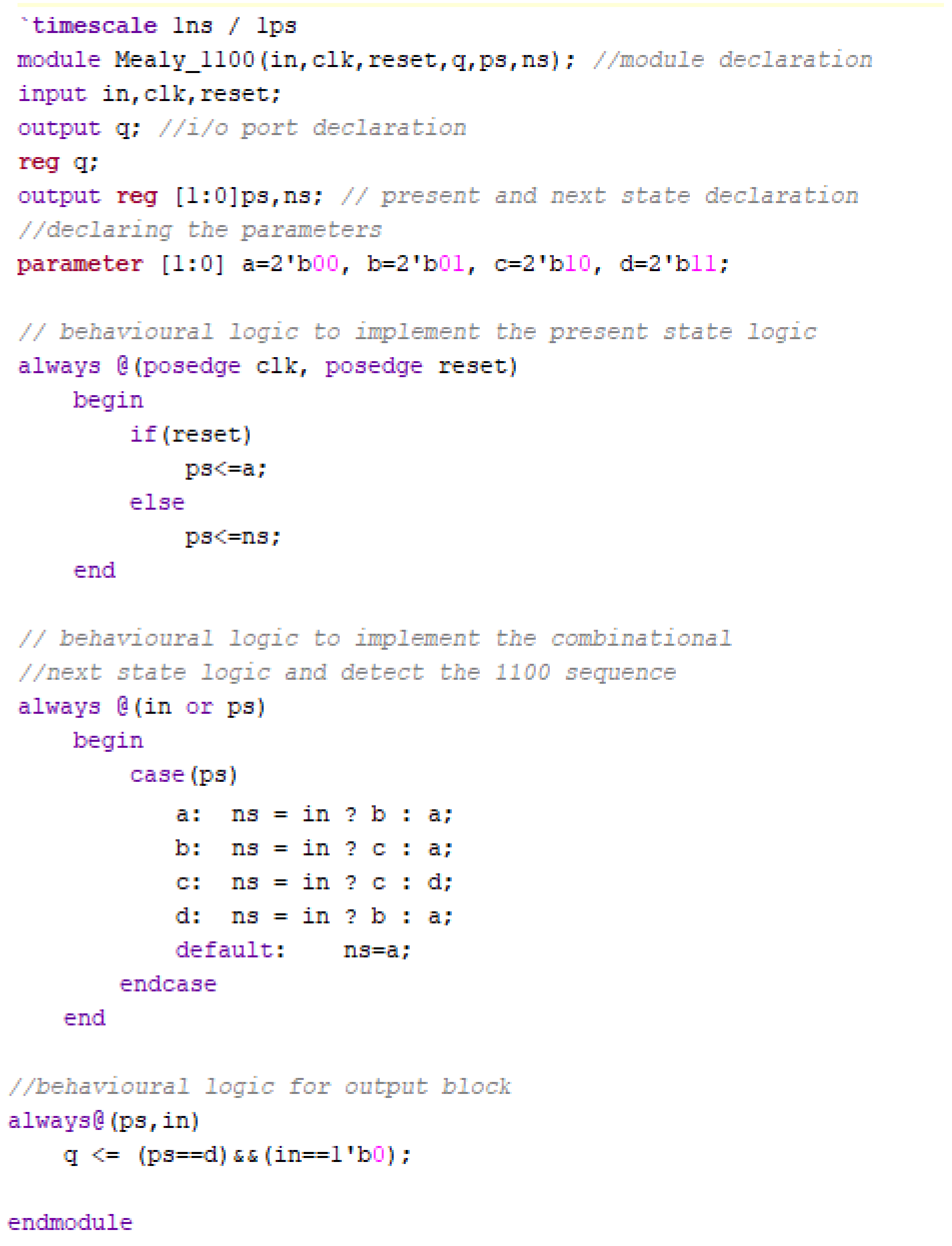
- **Simulation Waveforms**


295.000 ns
Name
Value
clk 1
reset 0
a[3:0] 8
b[3:0] 5
opcode[2:0] 5
c[7:0] 0d
0.000 ns
50.000 ns
100.000 ns
150.000 ns
200.000 ns
250.000 ns
X a 3 f 9 a 1 4 7 4 8 a 8 9 1 f 8 7 9 f 2 f a 9 f 8
X 1 7 a 2 a c 2 9 a b 8 a 9 c b 8 3 5 7 5 8 6 8 3 7 b 5 7 5
X 3 4 0 5 0 7 1 3 0 7 3 5 7 5 1 2 3 5 7 3 4 1 5
00 0a 03 0d 0f 0b 15 0c fe fa 00 0c f7 01 0c f6 09 0c 03 31 01 02 0a f0 01 00 01 08

2.7. LAB 8

Design a Mealy FSM to detect the sequence 1100 and write the test bench with transactor to randomize the appropriate inputs, Driver class with cover group for functional coverage.

- **Mealy_1100.v**



```
`timescale 1ns / 1ps
module Mealy_1100(in,clk,reset,q,ps,ns); //module declaration
input in,clk,reset;
output q; //i/o port declaration
reg q;
output reg [1:0]ps,ns; // present and next state declaration
//declaring the parameters
parameter [1:0] a=2'b00, b=2'b01, c=2'b10, d=2'b11;

// behavioural logic to implement the present state logic
always @(posedge clk, posedge reset)
    begin
        if(reset)
            ps<=a;
        else
            ps<=ns;
    end

// behavioural logic to implement the combinational
//next state logic and detect the 1100 sequence
always @(in or ps)
    begin
        case(ps)
            a:  ns = in ? b : a;
            b:  ns = in ? c : a;
            c:  ns = in ? c : d;
            d:  ns = in ? b : a;
            default:    ns=a;
        endcase
    end

//behavioural logic for output block
always@(ps,in)
    q <= (ps==d)&&(in==1'b0);

endmodule
```

- **inter.sv**

```
interface inter (input bit clk);
    bit in, reset;
    bit [1:0] ns,ps;
    bit q;
    clocking cbi @(posedge clk);
        default input #2ns output #3ns;
        input q;
        input ns,ps;
    endclocking
    modport mod_test(clocking cbi, input clk, output reset, in);
endinterface
```

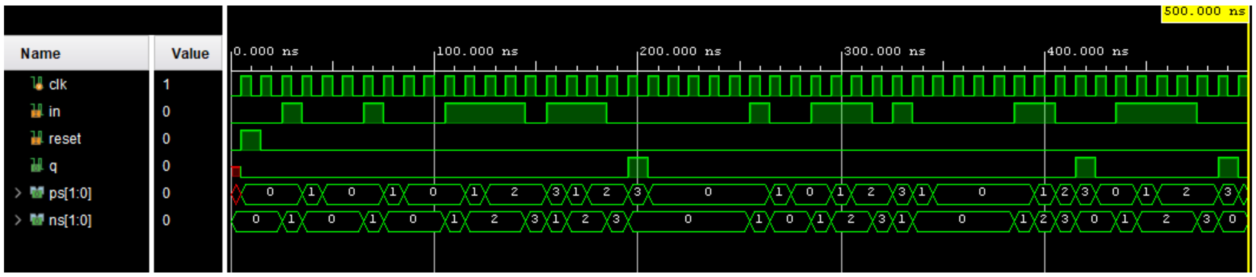
- **test.sv**

```
class transactor;
    rand bit in;
    constraint c1 {in inside {0,1};}
endclass: transactor

class driver;
    transactor tx;
    virtual inter.mod_test intf_o;
    covergroup cg;
        in: coverpoint intf_o.in{
        bins t0={0};
        bins t1={1};}
        reset: coverpoint intf_o.reset{
        bins r1={0};}
        ns: coverpoint intf_o.cbi.ns{
        bins t2[]={0,1};
        bins t3[]={2,3};}
        ps: coverpoint intf_o.cbi.ps{
        bins t4[]={0,1};
        bins t5[]={2,3};}
        q: coverpoint intf_o.cbi.q;
    endgroup
    function new(virtual inter.mod_test intf_o_new);
        this.intf_o=intf_o_new;
        this.cg=new();
    endfunction: new
    task drive_data();
        tx=new();
        if(!(tx.randomize()))
            $display("%0d : Randomization Error", $time);
        else
        begin
            intf_o.in<=tx.in;
            cg.sample();
            end
        endtask: drive_data
    endclass: driver

    program test(inter.mod_test intf_o);
        driver drv;
        initial begin
            #5 intf_o.reset<=1;
            #10 intf_o.reset<=0;
        end
        initial begin
            drv=new(intf_o);
            repeat(100)@(intf_o.cbi)
                drv.drive_data();
        end
    endprogram: test
```

- **top.sv**

```
module top();
    bit clk;
    always #5 clk=~clk;
    inter intf(clk);
    Mealy_1100 f1(.clk(clk), .reset(intf.reset), .in(intf.in),
                    .ns(intf.ns), .ps(intf.ps), .q(intf.q));
    test tb(intf);
    initial begin
        $dumpfile("top.vcd");
        $dumpvars;
        $monitor($time,"clk=%b, reset=%b, in=%b, ns=%b, ps=%b, q=%b",
                clk, intf.reset, intf.in, intf.ns, intf.ps, intf.q);
        #500 $finish;
    end
endmodule
```

- **Simulation Waveforms**



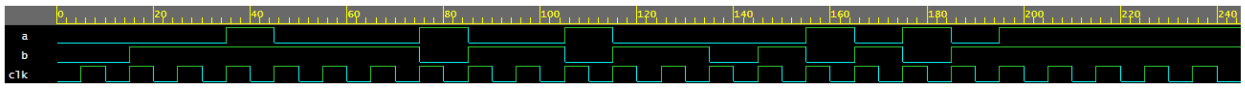
2.8. LAB 9

Program 9.1
Program to demonstrate simple assertions.

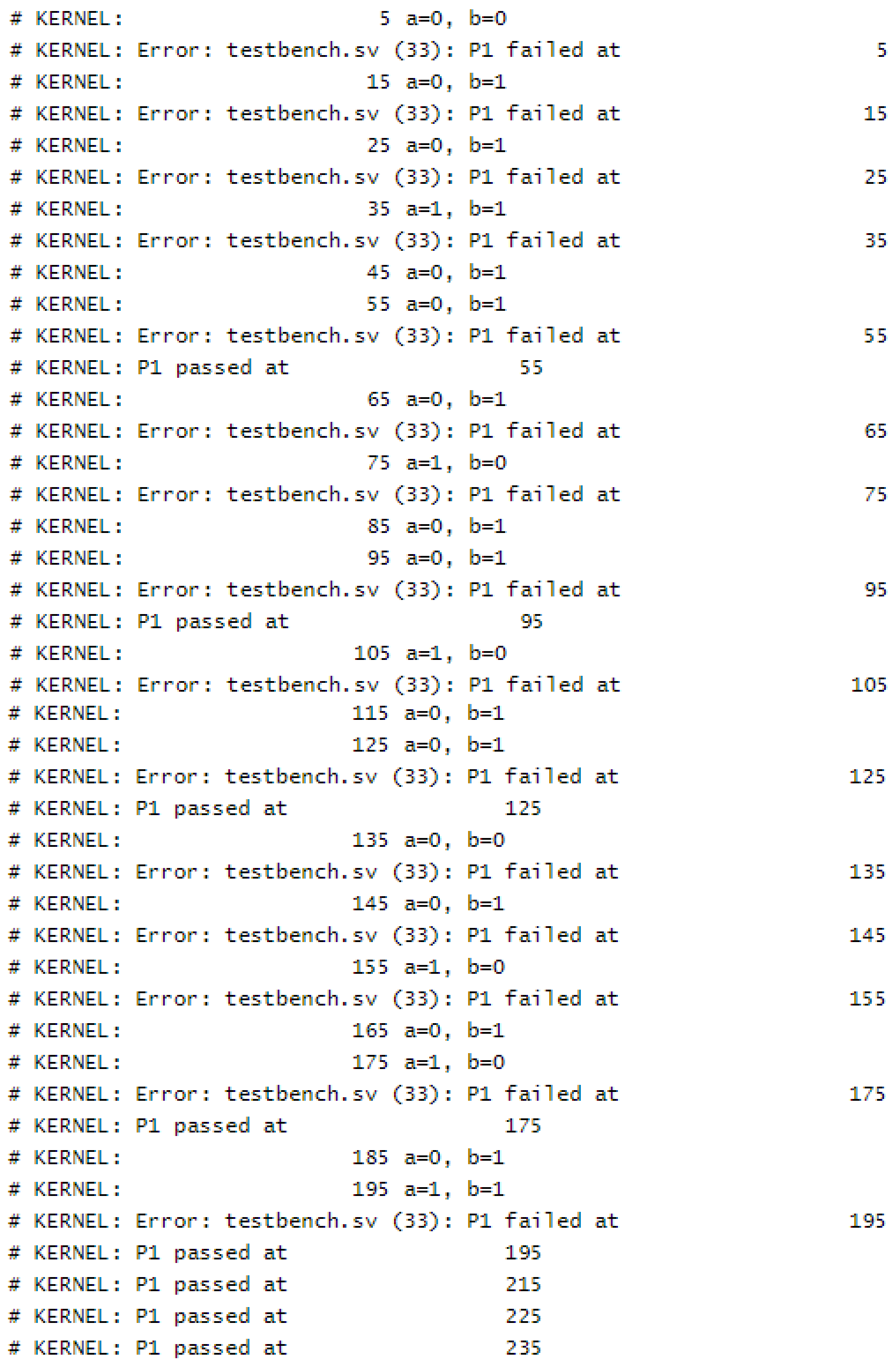
- **9.1.sv**

```
module test;

    bit a, b;
    bit clk;

    always #5 clk = ~clk;

    initial begin
      repeat(20) @(posedge clk) begin
            {a, b} = $random;
            $display($time, " a=%b, b=%b", a, b);
        end

        #50 $finish;
    end

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

    sequence s1;
        a ##1 b;
    endsequence

    property p1;
        @(posedge clk) s1;
    endproperty

    assert property (p1)
        $display("P1 passed at", $time);
    else
        $error("P1 failed at ", $time);

endmodule
```

- **Simulation Waveforms**



- **Simulation Output**



```
# KERNEL:                    5 a=0, b=0
# KERNEL: Error: testbench.sv (33): P1 failed at                     5
# KERNEL:                   15 a=0, b=1
# KERNEL: Error: testbench.sv (33): P1 failed at                    15
# KERNEL:                   25 a=0, b=1
# KERNEL: Error: testbench.sv (33): P1 failed at                    25
# KERNEL:                   35 a=1, b=1
# KERNEL: Error: testbench.sv (33): P1 failed at                    35
# KERNEL:                   45 a=0, b=1
# KERNEL:                   55 a=0, b=1
# KERNEL: Error: testbench.sv (33): P1 failed at                    55
# KERNEL: P1 passed at                    55
# KERNEL:                   65 a=0, b=1
# KERNEL: Error: testbench.sv (33): P1 failed at                    65
# KERNEL:                   75 a=1, b=0
# KERNEL: Error: testbench.sv (33): P1 failed at                    75
# KERNEL:                   85 a=0, b=1
# KERNEL:                   95 a=0, b=1
# KERNEL: Error: testbench.sv (33): P1 failed at                    95
# KERNEL: P1 passed at                    95
# KERNEL:                  105 a=1, b=0
# KERNEL: Error: testbench.sv (33): P1 failed at                   105
# KERNEL:                  115 a=0, b=1
# KERNEL:                  125 a=0, b=1
# KERNEL: Error: testbench.sv (33): P1 failed at                   125
# KERNEL: P1 passed at                   125
# KERNEL:                  135 a=0, b=0
# KERNEL: Error: testbench.sv (33): P1 failed at                   135
# KERNEL:                  145 a=0, b=1
# KERNEL: Error: testbench.sv (33): P1 failed at                   145
# KERNEL:                  155 a=1, b=0
# KERNEL: Error: testbench.sv (33): P1 failed at                   155
# KERNEL:                  165 a=0, b=1
# KERNEL:                  175 a=1, b=0
# KERNEL: Error: testbench.sv (33): P1 failed at                   175
# KERNEL: P1 passed at                   175
# KERNEL:                  185 a=0, b=1
# KERNEL:                  195 a=1, b=1
# KERNEL: Error: testbench.sv (33): P1 failed at                   195
# KERNEL: P1 passed at                   195
# KERNEL: P1 passed at                   215
# KERNEL: P1 passed at                   225
# KERNEL: P1 passed at                   235
```

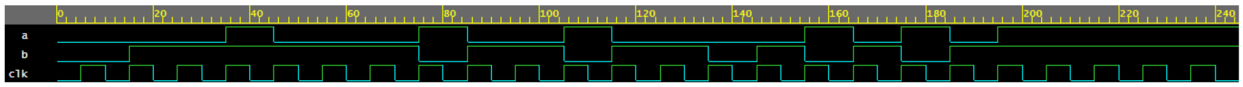
Program 9.2

Program to demonstrate simple assertions.

- **9.2.sv**

```
module test;

    bit a, b;
    bit clk;

    always #5 clk = ~clk;

    initial begin
        repeat(20) @(posedge clk) begin
            {a, b} = $random;
            $display($time, " a=%b, b=%b", a, b);
        end

        #50 $finish;
    end

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

    sequence s1;
        (a==0) && (b==1);
    endsequence

    sequence s2;
        (a==1);
    endsequence

    property p1;
        @(posedge clk) s1 or s2 ;
    endproperty

    assert property (p1)
        $display("P1 passed at", $time);
    else
        $error("P1 failed at ", $time);
endmodule
```

- **Simulation Waveforms**



- **Simulation Output**

```
# KERNEL:                     5 a=0, b=0
# KERNEL: Error: testbench.sv (37): P1 failed at                     5
# KERNEL:                    15 a=0, b=1
# KERNEL:                    25 a=0, b=1
# KERNEL: P1 passed at                    25
# KERNEL:                    35 a=1, b=1
# KERNEL:                    45 a=0, b=1
# KERNEL:                    55 a=0, b=1
# KERNEL:                    65 a=0, b=1
# KERNEL:                    75 a=1, b=0
# KERNEL:                    85 a=0, b=1
# KERNEL:                    95 a=0, b=1
# KERNEL:                   105 a=1, b=0
# KERNEL:                   115 a=0, b=1
# KERNEL:                   125 a=0, b=1
# KERNEL:                   135 a=0, b=0
# KERNEL:                   145 a=0, b=1
# KERNEL: Error: testbench.sv (37): P1 failed at                   145
# KERNEL:                   155 a=1, b=0
# KERNEL: P1 passed at                   155
# KERNEL:                   165 a=0, b=1
# KERNEL:                   175 a=1, b=0
# KERNEL:                   185 a=0, b=1
# KERNEL:                   195 a=1, b=1
```

Program 9.3

Program to implement a 3-bit counter. Test the same using interface block with assertions and mod ports in SV environment.

- **counter.v**

```
`timescale 1ns / 1ps
module counter(input clk,reset, output reg [2:0] c);

    always@(posedge clk, posedge reset) begin
        if(reset)
            c<=3'b000;
        else
            c <= c+1;
    end

endmodule
```

- **inter.sv**

```
interface inter (input bit clk);

    logic reset;
    logic [2:0] c;
    modport tb(output reset);

    sequence s1;
        (reset==0) ##[0:$](c==3'd5);
    endsequence

    property p1;
        @(posedge clk) s1;
    endproperty:p1

    a1:assert property (p1)
        $display("count=5,P1 passed");
    else
        $display("P1 failed");

    property p2;
        @(posedge clk) (reset==1)|-> (c==3'b000);
    endproperty:p2

     a2: assert property(p2)
         $display("counter reset,P2 passed");
     else
         $display("P2 failed");

 endinterface
```

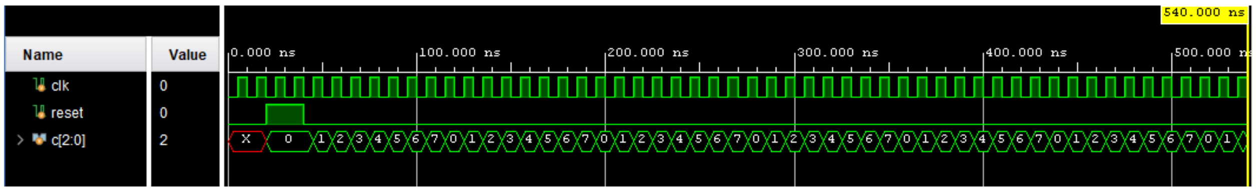
- **test.sv**

```
program test(inter.tb T1);

    initial begin
        T1.reset<=0;
        #20
        T1.reset<=1;
        #20
        T1.reset<=0;
        #500
        $finish;
    end

endprogram: test
```

- **top.sv**

```
module top();

    bit clk;

    always #5 clk=~clk;

    inter intf(clk);
    counter f1(.clk(clk),.reset(intf.reset),.c(intf.c));
    test tb(intf);

    initial begin
        // dump *everything*
        $dumpfile("top.vcd");
        $dumpvars;
        $shm_open( "top.shm",,,0);
        $shm_probe( "AC", top );
    end

endmodule
```

- **Simulation Waveforms**



- **Simulation Outputs**

```
Time: 25 ns Started: 25 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
P1 failed

Time: 25 ns Started: 25 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:25
counter reset,P2 passed

Time: 35 ns Started: 35 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
P1 failed

Time: 35 ns Started: 35 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:25
counter reset,P2 passed

Time: 95 ns Started: 5 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
```

```
count=5,P1 passed

Time: 95 ns Started: 15 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 95 ns Started: 45 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 95 ns Started: 55 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 95 ns Started: 65 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 95 ns Started: 75 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 95 ns Started: 85 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 95 ns Started: 95 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 175 ns Started: 105 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 175 ns Started: 115 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 175 ns Started: 125 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 175 ns Started: 135 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
```

```
count=5,P1 passed

Time: 175 ns Started: 145 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 175 ns Started: 155 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 175 ns Started: 165 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 175 ns Started: 175 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 255 ns Started: 185 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 255 ns Started: 195 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 255 ns Started: 205 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 255 ns Started: 215 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 255 ns Started: 225 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 255 ns Started: 235 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 255 ns Started: 245 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
```

```
count=5,P1 passed

Time: 255 ns Started: 255 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 335 ns Started: 265 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 335 ns Started: 275 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 335 ns Started: 285 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 335 ns Started: 295 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 335 ns Started: 305 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 335 ns Started: 315 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 335 ns Started: 325 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 335 ns Started: 335 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 415 ns Started: 345 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 415 ns Started: 355 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
```

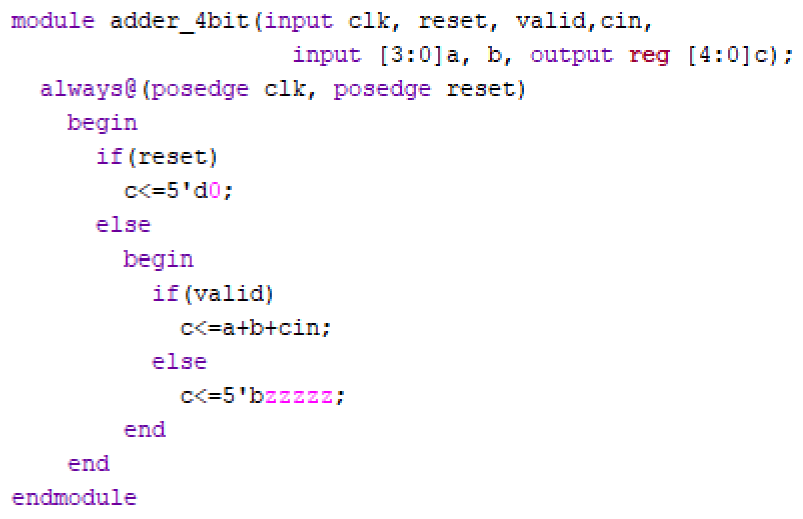
```
count=5,P1 passed

Time: 415 ns Started: 365 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 415 ns Started: 375 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 415 ns Started: 385 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 415 ns Started: 395 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 415 ns Started: 405 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 415 ns Started: 415 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 495 ns Started: 425 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 495 ns Started: 435 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 495 ns Started: 445 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 495 ns Started: 455 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 495 ns Started: 465 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
```

```
count=5,P1 passed

Time: 495 ns Started: 475 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 495 ns Started: 485 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed

Time: 495 ns Started: 495 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 1/Lab 9/Lab 9.srcs/sim_1/new/inter.sv Line:16
count=5,P1 passed
$finish called at time : 540 ns :
```

2.9. LAB 10

Program to demonstrate System Verilog Layered Test Bench Architecture with Environment class for a 4-bit adder.

- **adder_4bit.v**



```
module adder_4bit(input clk, reset, valid,cin,
                    input [3:0]a, b, output reg [4:0]c);
  always@(posedge clk, posedge reset)
    begin
      if(reset)
        c<=5'd0;
      else
        begin
          if(valid)
            c<=a+b+cin;
          else
            c<=5'bzzzzz;
        end
    end
endmodule
```

- **inter.sv**

```
interface inter (input bit clk);
  bit valid,reset,cin;
  bit [3:0]a,b;
  bit [4:0]c;

  clocking cb@(posedge clk);
    default input #2 output #2;
    output valid, a, b,cin;
    input c;
  endclocking

  modport tb(output reset, clocking cb);
endinterface
```

- **test.sv**

```
//transactor
class transactor;
  randc bit [3:0]a,b;
  randc bit cin;
  bit valid;
  bit [4:0] c;
endclass

//generator
class generator;
  transactor tx;
  mailbox gen2drv;
  virtual inter intf;

  function new(mailbox gen2drv, virtual inter intf);
    this.gen2drv=gen2drv;
    this.intf=intf;
    tx=new();
  endfunction

  task create();
    repeat(20)@(posedge intf.clk)
      begin
        if(tx.randomize())
          begin
            $display(tx.a, tx.b, tx.cin);
            gen2drv.put(tx);
          end
        else
          $display("Randomization error");
      end
  endtask
endclass

//driver
```

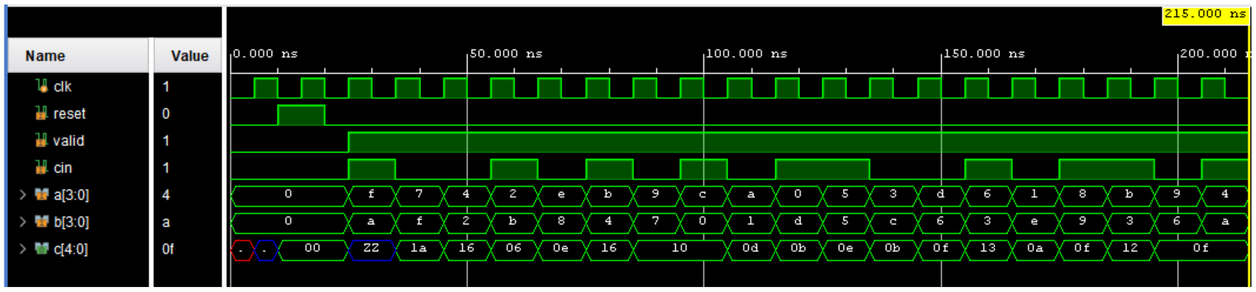
```
class driver;
  transactor tx;
  virtual inter intf;
  mailbox gen2drv;
  mailbox drv2scb;

  function new(virtual inter intf, mailbox gen2drv, mailbox drv2scb);
    this.intf=intf;
    this.gen2drv=gen2drv;
    this.drv2scb=drv2scb;
  endfunction

  task drive_data();
    repeat(20)@(posedge intf.clk)
      begin
        gen2drv.get(tx);
        $display(tx.a, tx.b, tx.cin);
        intf.valid<=1;
        intf.a<=tx.a;
        intf.b<=tx.b;
        intf.cin<=tx.cin;
        drv2scb.put(tx);
      end
  endtask
endclass

//monitor class
class monitor;
  transactor tx;
  virtual inter intf;
  mailbox mon2scb;

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

  task samp_values();
    repeat(20)@(posedge intf.clk)
      begin
        tx.a=intf.a;
        tx.b=intf.b;
        tx.cin=intf.cin;
        tx.c=intf.c;
        mon2scb.put(tx);
      end
  endtask
endclass
```

```
//scoreboard
class scoreboard;
  mailbox mon2scb;
  int trans;

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

  task comp();
    transactor tx;
    mon2scb.get(tx);
    if((tx.a+tx.b+tx.cin)==tx.c)
      $display("Result correct, a=%d, b=%d,cin=%d, c=%d",tx.a,tx.b,tx.cin,tx.c);
    else
      $display("BUG");
    trans++;
  endtask
endclass

//environment class
class environment;
  virtual inter intf;
  mailbox drv2scb;
  mailbox gen2drv;
  mailbox mon2scb;
  generator gen;
  driver drv;
  monitor mon;
  scoreboard scb;

  function new(virtual inter intf);
    this.intf=intf;
  endfunction

  function void build();
    drv2scb=new();
    gen2drv=new();
    mon2scb=new();
    gen=new(gen2drv,intf);
    drv=new(intf,gen2drv,drv2scb);
    mon=new(intf,mon2scb);
    scb=new(mon2scb);
  endfunction
  task rst();
    intf.reset<=0;
    #10 intf.reset<=1;
    #10 intf.reset<=0;
  endtask

  task start();
    rst();
    fork
      gen.create();
      drv.drive_data();
      mon.samp_values();
      scb.comp();
    join
  endtask
endclass


  //test class
  program test(inter.tb T1);
    environment env;
    initial
      begin
        env=new(T1);
        env.build();
        env.start();
      end
  endprogram
```

- **top.sv**

```
  //top module
  module top;
    bit clk;
    always #5 clk=~clk;
    inter i1(clk);
    test Test_b(i1);
    adder_4bit A1(clk, i1.reset, i1.valid,i1.cin, i1.a, i1.b, i1.c);
    initial
      begin
        $dumpfile("top.vcd");
        $dumpvars();
        $monitor($time, "clk=%d, reset=%d, valid=%d,a=%d,b=%d, cin=%d, c=%d",
                     clk,i1.reset,i1.valid, i1.a,i1.b,i1.cin,i1.c);
        #500 $finish;
      end
  endmodule
```

- **Simulation Waveforms**


215.000 ns
Name
Value
0.000 ns
50.000 ns
100.000 ns
150.000 ns
200.000 ns
clk 1
reset 0
valid 1
cin 1
a[3:0] 4
b[3:0] a
c[4:0] 0f
0 f 7 4 2 e b 9 c a 0 5 3 d 6 1 8 b 9 4
0 a f 2 b 8 4 7 0 1 d 5 c 6 3 e 9 3 6 a
00 ZZ 1a 16 06 0e 16 10 0d 0b 0e 0b 0f 13 0a 0f 12 0f

- **Simulation Outputs**

```
0clk=0, reset=0, valid=0,a= 0,b= 0, cin=0, c= 0
                 5clk=1, reset=0, valid=0,a= 0,b= 0, cin=0, c= 0
                10clk=0, reset=1, valid=0,a= 0,b= 0, cin=0, c= 0
                15clk=1, reset=1, valid=0,a= 0,b= 0, cin=0, c= 0
                20clk=0, reset=0, valid=0,a= 0,b= 0, cin=0, c= 0
15101
Result correct, a= 0, b= 0,cin=0, c= 0
15101
                25clk=1, reset=0, valid=1,a=15,b=10, cin=1, c= 0
                30clk=0, reset=0, valid=1,a=15,b=10, cin=1, c= 0
 7150
 7150
                35clk=1, reset=0, valid=1,a= 7,b=15, cin=0, c=26
                40clk=0, reset=0, valid=1,a= 7,b=15, cin=0, c=26
 4 20
 4 20
                45clk=1, reset=0, valid=1,a= 4,b= 2, cin=0, c=22
                50clk=0, reset=0, valid=1,a= 4,b= 2, cin=0, c=22
 2111
 2111
                55clk=1, reset=0, valid=1,a= 2,b=11, cin=1, c= 6
                60clk=0, reset=0, valid=1,a= 2,b=11, cin=1, c= 6
14 80
14 80
                65clk=1, reset=0, valid=1,a=14,b= 8, cin=0, c=14
                70clk=0, reset=0, valid=1,a=14,b= 8, cin=0, c=14
11 41
11 41
                75clk=1, reset=0, valid=1,a=11,b= 4, cin=1, c=22
                80clk=0, reset=0, valid=1,a=11,b= 4, cin=1, c=22
 9 70
 9 70
                85clk=1, reset=0, valid=1,a= 9,b= 7, cin=0, c=16
```

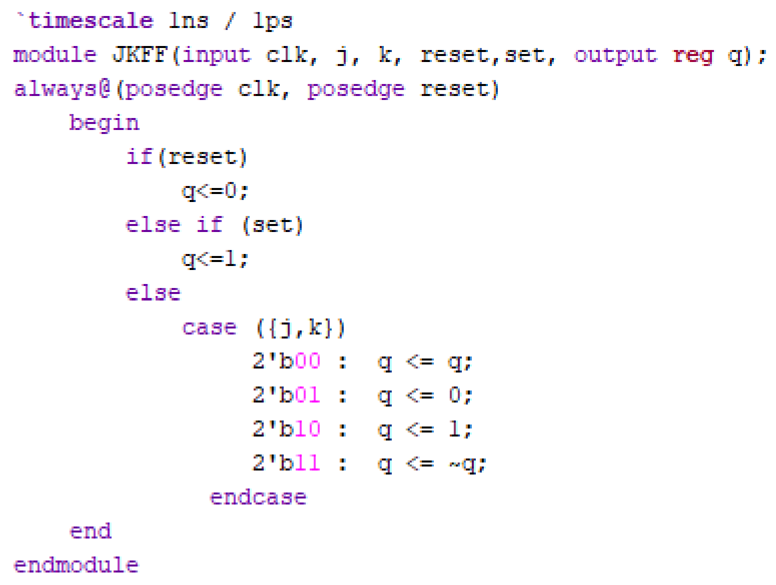
```
                  90clk=0, reset=0, valid=1,a= 9,b= 7, cin=0, c=16
12 01
12 01
                  95clk=1, reset=0, valid=1,a=12,b= 0, cin=1, c=16
                 100clk=0, reset=0, valid=1,a=12,b= 0, cin=1, c=16
10 10
10 10
                 105clk=1, reset=0, valid=1,a=10,b= 1, cin=0, c=13
                 110clk=0, reset=0, valid=1,a=10,b= 1, cin=0, c=13
 0131
 0131
                 115clk=1, reset=0, valid=1,a= 0,b=13, cin=1, c=11
                 120clk=0, reset=0, valid=1,a= 0,b=13, cin=1, c=11
 5 51
 5 51
                 125clk=1, reset=0, valid=1,a= 5,b= 5, cin=1, c=14
                 130clk=0, reset=0, valid=1,a= 5,b= 5, cin=1, c=14
 3120
 3120
                 135clk=1, reset=0, valid=1,a= 3,b=12, cin=0, c=11
                 140clk=0, reset=0, valid=1,a= 3,b=12, cin=0, c=11
13 60
13 60
                 145clk=1, reset=0, valid=1,a=13,b= 6, cin=0, c=15
                 150clk=0, reset=0, valid=1,a=13,b= 6, cin=0, c=15
 6 31
 6 31
                 155clk=1, reset=0, valid=1,a= 6,b= 3, cin=1, c=19
                 160clk=0, reset=0, valid=1,a= 6,b= 3, cin=1, c=19
 1140
 1140
                 165clk=1, reset=0, valid=1,a= 1,b=14, cin=0, c=10
                 170clk=0, reset=0, valid=1,a= 1,b=14, cin=0, c=10
 8 91
 8 91
                 175clk=1, reset=0, valid=1,a= 8,b= 9, cin=1, c=15
                 180clk=0, reset=0, valid=1,a= 8,b= 9, cin=1, c=15
11 31
11 31
                 185clk=1, reset=0, valid=1,a=11,b= 3, cin=1, c=18
                 190clk=0, reset=0, valid=1,a=11,b= 3, cin=1, c=18
 9 60
 9 60
                 195clk=1, reset=0, valid=1,a= 9,b= 6, cin=0, c=15
```

```
                200clk=0, reset=0, valid=1,a= 9,b= 6, cin=0, c=15
 4101
 4101
                205clk=1, reset=0, valid=1,a= 4,b=10, cin=1, c=15
                210clk=0, reset=0, valid=1,a= 4,b=10, cin=1, c=15
 7 10
 7 10
$finish called at time : 215 ns :
```

3. AAT 2

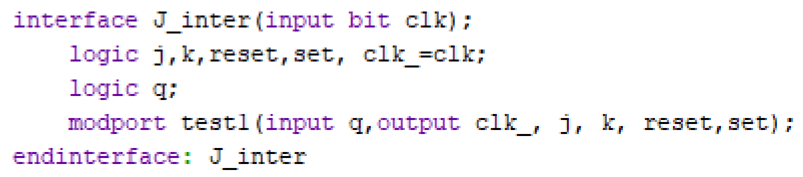
3.1. Design JKFF with asynchronous reset and synchronous set. Test the same using interface block and mod ports in SV environment for code coverage.

- **JKFF.v**



```
`timescale 1ns / 1ps
module JKFF(input clk, j, k, reset,set, output reg q);
always@(posedge clk, posedge reset)
    begin
        if(reset)
            q<=0;
        else if (set)
            q<=1;
        else
            case ({j,k})
                 2'b00 :  q <= q;
                 2'b01 :  q <= 0;
                 2'b10 :  q <= 1;
                 2'b11 :  q <= ~q;
              endcase
    end
endmodule
```

- **J_inter.sv**



```
interface J_inter(input bit clk);
    logic j,k,reset,set, clk_=clk;
    logic q;
    modport test1(input q,output clk_, j, k, reset,set);
endinterface: J_inter
```

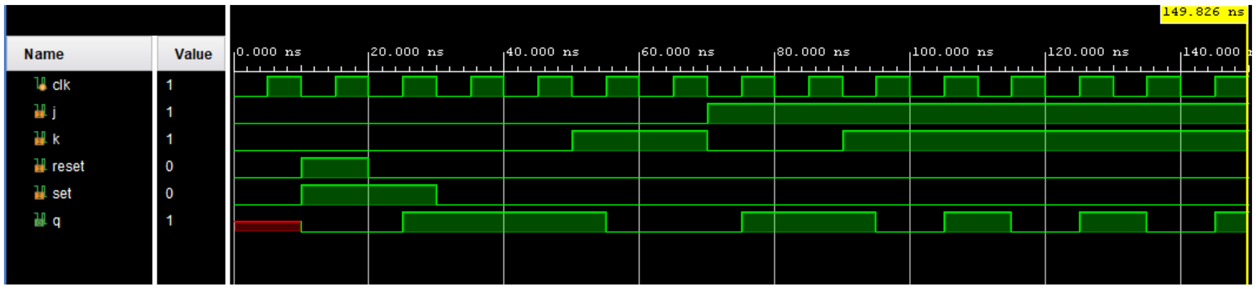
- **J_tb.sv**

```
module J_tb(J_inter.test1 t1);
    initial begin
        t1.reset=0;t1.set=0;
        t1.j=0;t1.k=0;
        #10 t1.reset=1;t1.set=1;
        t1.j=0;t1.k=0;
        #10 t1.reset=0;
        t1.j=0;t1.k=0;
        #10 t1.j=0;t1.k=0;t1.set=0;
        #20 t1.j=0;t1.k=1;
        #20 t1.j=1;t1.k=0;
        #20 t1.j=1;t1.k=1;
        #60 $finish;
    end
endmodule
```

- **J_top.sv**

```
`timescale 1ns / 1ps
module J_top;
    bit clk;
    J_inter i1(clk);
    JKFF Dut(.clk(clk),.j(i1.j),.k(i1.k),.reset(i1.reset),.set(i1.set)
                ,.q(i1.q));
    J_tb T1(i1);
    always #5 clk=~clk;
    initial
    begin
        $dumpfile("top.vcd");
        $dumpvars;
        $monitor("time=%d, clk=%b, reset=%b,set=%b, j=%b, k=%b, q=%b",
        $time, clk, i1.reset, i1.set,
        i1.j, i1.k, i1.q);
    end
endmodule
```

- **Simulation Waveforms**



3.2. Design a 4-bit ALU that has 3-bit opcode to decide the operation.

- **ALU.v**

```
`timescale 1ns / 1ps
module ALU(input clk, reset, input [3:0]
a,b , input [2:0] opcode, output reg [7:0]
c);
    always@(posedge clk, posedge reset)
    begin
    if(reset)
        c<=8'd0;
    else
        case(opcode)
            3'd0: c<=a+b;
            3'd1: c<=a-b;
            3'd2: c<=a*b;
            3'd3: begin
                if (b!=0) c<=a/b;
                else c<=8'dx; end
            3'd4: c<=a&b;
            3'd5: c<=a|b;
            3'd6: c<=a^b;
            3'd7: c<=~a;
            default: c<=8'dx;
        endcase
    end
endmodule
```

- **ALU_inter.sv**

```
interface ALU_inter( input bit clk);
    logic reset;
    logic [3:0] a,b;
    logic [2:0] opcode;
    logic [7:0] c;
    clocking cb@(posedge clk);
        default input #2ns output #3ns;
        output a, b, opcode;
    endclocking
    modport TB(clocking cb, output reset);
endinterface
```

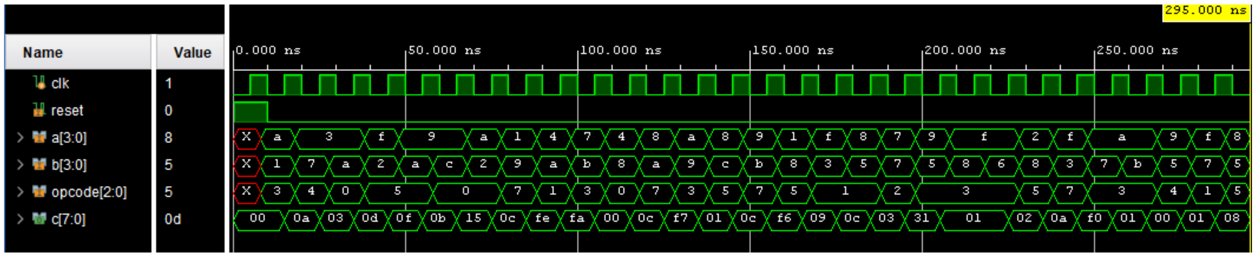
- **ALU_tb.sv**

```
class transactor;
    rand bit [3:0] a,b;
    rand bit [2:0] opcode;
    constraint C1 {opcode inside {[0:5],7};}
    constraint C2 {a inside {[1:10],15};}
    constraint C3 {b>0;b<14;}
endclass

class driver;
    transactor tx;
    virtual ALU_inter.TB il;
    function new(virtual ALU_inter.TB il_new);
        this.il=il_new;
        this.tx=new();
        $display ("%d : Driver: new method created", $time);
    endfunction
    task drive_data();
        begin
        repeat(30) @( il.cb)
        begin
        if((tx.randomize()))
        begin
            il.cb.a<=tx.a;
            il.cb.b<=tx.b;
            il.cb.opcode<=tx.opcode;
        end
        else
            $display("%d : Randomization error", $time);
        end
        end
    endtask
endclass

program ALU_tb(ALU_inter.TB itb);
    driver drv=new(itb);
    initial
    begin
    itb.reset<=1;
    #10 itb.reset<=0;
    end
    initial
    begin
    drv.drive_data();
    end
endprogram
```

- **ALU_top.sv**

```
module top;
    bit clk;
    ALU_inter Ia (clk);
    ALU DUT(clk, Ia.reset, Ia.a, Ia.b , Ia.opcode, Ia.c);
    ALU_tb TB1 (Ia.TB);
    always #5 clk=~clk;
    initial
    begin
        $dumpfile("top.vcd");
        $dumpvars;
        $monitor("time=%d, clk=%b, reset=%b, a=%b,b=%b,opcode=%b, c=%b",
                        $time, clk, Ia.reset,Ia.a,Ia.b,Ia.opcode, Ia.c);
    #500 $finish;
    end
endmodule
```

- **Simulation Waveforms**



3.3. Design a 4-bit shift register and randomize the direction and parallel load inputs. Test the design using driver and transactor class.

- **shift_reg.v**

```
`timescale 1ns / 1ps
module shift_reg(input [1:0] select, input [3:0] parallel_in,
            input serial_in, clk, reset, output reg [3:0] out);

  always @(posedge clk or posedge reset)
    begin
            if (reset == 1)
                out <= 4'b0000;
            else
              case(select)
                2'b00: begin //left shift
                    out[0] <= serial_in;
                    out[3:1] <= out[2:0];
                    end
                2'b01: begin //right shift
                    out[3] <= serial_in;
                    out[2:0] <= out[3:1];
                    end
                2'b10: out <= parallel_in; // parallel in
                default: out <= out; //Hold the previous value of output
              endcase
    end
endmodule
```

- **S_inter.sv**

```
interface S_inter( input bit clk);
    logic reset, serial_in;
    logic [3:0] parallel_in, out;
    logic [1:0] select;

    clocking cb@(posedge clk);
        default input #2ns output #3ns;
        output serial_in, parallel_in, select;
    endclocking
    modport TB(clocking cb, output reset, output out);
endinterface
```

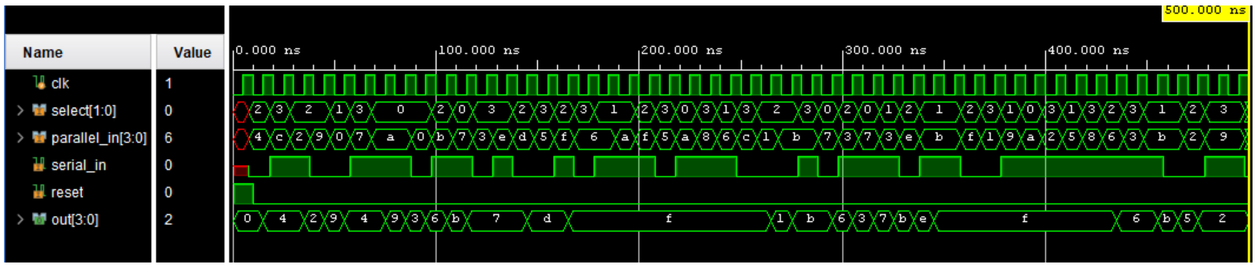
- **S_tb.sv**

```
class transactor;
    rand bit serial_in;
    rand bit [3:0] parallel_in;
    rand bit [1:0] select;
    constraint C1 {select inside {[0:3]};}
    constraint C2 {parallel_in inside {[0:15]};}
    constraint C3 {serial_in inside {0,1};}
endclass

class driver;
    transactor tx;
    virtual S_inter.TB il;
    function new(virtual S_inter.TB il_new);
        this.il=il_new;
        this.tx=new();
        $display ("%d : Driver: new method created", $time);
    endfunction
    task drive_data();
        begin
        repeat(100) @( il.cb)
        begin
        if((tx.randomize()))
        begin
            il.cb.serial_in <= tx.serial_in;
            il.cb.parallel_in <= tx.parallel_in;
            il.cb.select <= tx.select;
        end
        else
            $display("%d : Randomization error", $time);
        end
        end
    endtask
endclass

program S_tb(S_inter.TB itb);
    driver drv=new(itb);
    initial begin
        itb.reset<=1;
        #10 itb.reset<=0;
    end
    initial begin
        drv.drive_data();
    end
endprogram
```

- **S_top.sv**

```
module top;
bit clk;
S_inter Ia (clk);
shift_reg DUT(.clk(clk), .reset(Ia.reset), .serial_in(Ia.serial_in),
        .parallel_in(Ia.parallel_in), .select(Ia.select), .out(Ia.out));
S_tb TB1 (Ia.TB);
always #5 clk=~clk;
initial begin
    $dumpfile("top.vcd");
    $dumpvars;
    $monitor("time=%d, clk=%b, reset=%b, serial_in=%b, parallel_in=%b, select=%b, out=%b",
                $time, clk, Ia.reset, Ia.serial_in,
    Ia.parallel_in, Ia.select, Ia.out);
    #500 $finish;
end
endmodule
```

- **Simulation Waveforms**



3.4. Design a Moore FSM to detect the sequence 1001 with overlap and write the test bench with transactor to randomize the appropriate inputs, Driver class with cover group for functional coverage.

- **Moore_1001.v**

```
`timescale 1ns / 1ps
module Moore_1001(in,clk,reset,q,ps,ns); //module declaration
    input in,clk,reset;
    output q; //i/o port declaration
    reg q;
    output reg [2:0]ps,ns; // present and next state declaration

    parameter [2:0] a=3'b000, b=3'b001, c=3'b010, d=3'b011, e=3'b100; //declaring the parameters

    // behavioural logic to implement the present state logic
    always @(posedge clk, posedge reset)
        begin
            if(reset)
                ps<=a;
            else
                ps<=ns;
        end
    // behavioural logic to implement the combinational next state logic and detect the 1001 sequence
    always @(in or ps)
        begin
            case(ps)
                a:  ns = in ? b : a;
                b:  ns = in ? b : c;
                c:  ns = in ? b : d;
                d:  ns = in ? e : a;
                e:  ns = in ? b : c;
                default:    ns = a;
            endcase
        end

    //behavioural logic for output block
    always @(posedge clk, posedge reset)
        begin
            if(reset)
                q <= 0;
            else
                q <= ps==e;
        end

endmodule
```

- **inter.sv**

```
interface inter (input bit clk);
    bit in, reset;
    bit [2:0] ns,ps;
    bit q;

    clocking cbi @(posedge clk);
        default input #2ns output #3ns;
        input q;
        input ns,ps;
    endclocking

    modport mod_test(clocking cbi, input clk, in, output reset);

endinterface
```

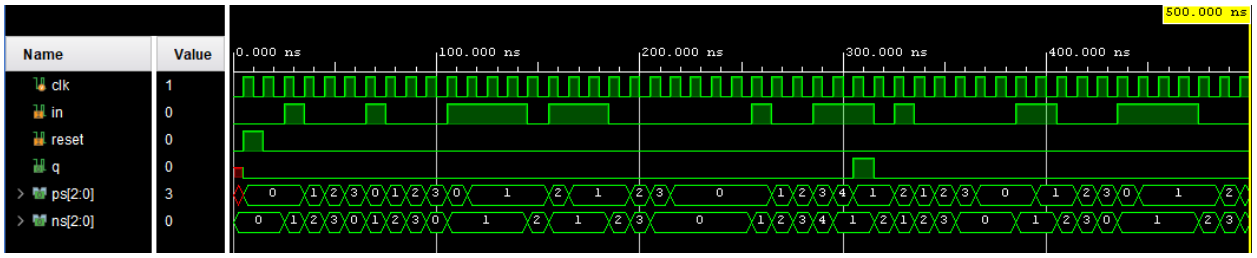
- **tb.sv**

```
class transactor;
    rand bit in;
    constraint c1 {in inside {0,1};}
endclass: transactor

class driver;
    transactor tx;
    virtual inter.mod_test intf_o;
    covergroup cg;
        in: coverpoint intf_o.in{
        bins t0={0};
        bins t1={1};}
        reset: coverpoint intf_o.reset{
        bins r1={0};}
        ns: coverpoint intf_o.cbi.ns{
        bins t2[]={0,1};
        bins t3[]={2,3};}
        ps: coverpoint intf_o.cbi.ps{
        bins t4[]={0,1};
        bins t5[]={2,3};}
        q: coverpoint intf_o.cbi.q;
    endgroup
    function new(virtual inter.mod_test intf_o_new);
        this.intf_o=intf_o_new;
        this.cg=new();
    endfunction: new
    task drive_data();
        tx=new();
        if(!(tx.randomize()))
            $display("%0d : Randomization Error", $time);
        else
        begin
            intf_o.in<=tx.in;
            cg.sample();
        end
    endtask: drive_data
endclass: driver

program test(inter.mod_test intf_o);
    driver drv;
    initial begin
        #5 intf_o.reset<=1;
        #10 intf_o.reset<=0;
    end
    initial begin
        drv=new(intf_o);
        repeat(100)@(intf_o.cbi)
            drv.drive_data();
    end
endprogram: test
```

- **top.sv**

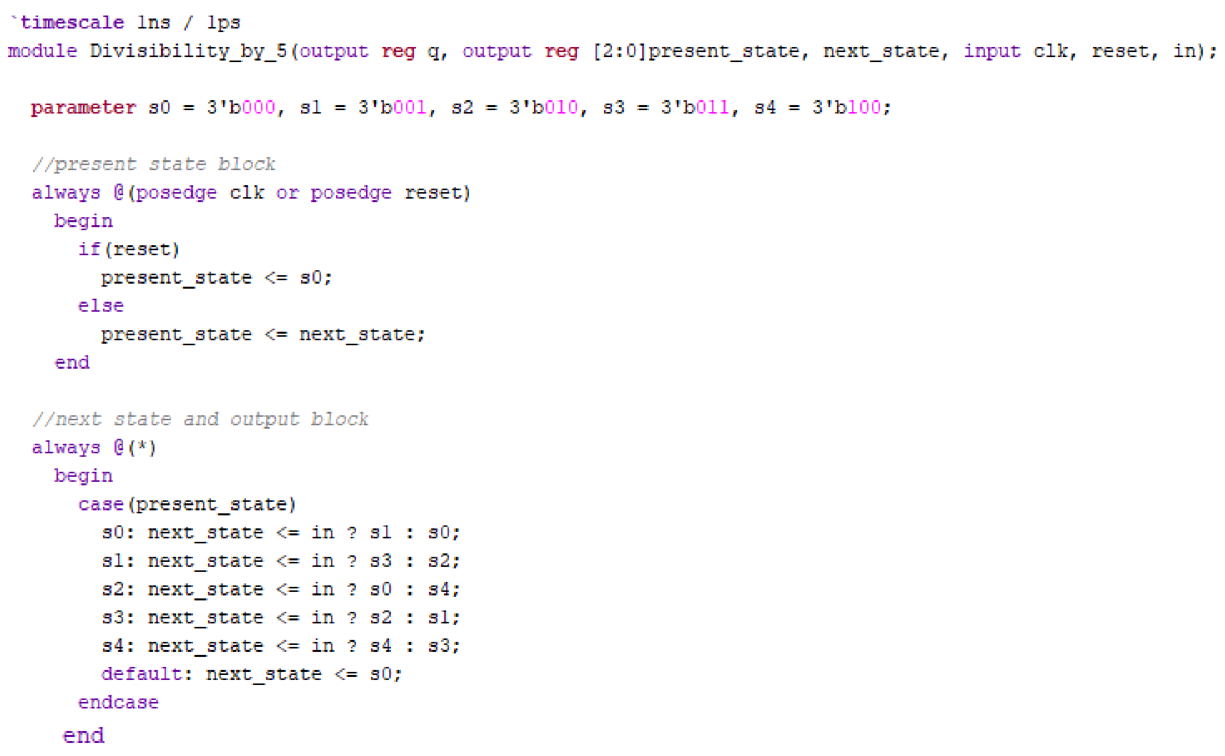
```
module top();
    bit clk;
    always #5 clk=~clk;
    inter intf(clk);
    Moore_1001 f1(.clk(clk), .reset(intf.reset), .in(intf.in),
                    .ns(intf.ns), .ps(intf.ps), .q(intf.q));
    test tb(intf);
    initial begin
        $dumpfile("top.vcd");
        $dumpvars;
        $monitor($time,"clk=%b, reset=%b, in=%b, ns=%b, ps=%b, q=%b",
                clk, intf.reset, intf.in, intf.ns, intf.ps, intf.q);
        #500 $finish;
    end
endmodule
```

- **Simulation Waveforms**

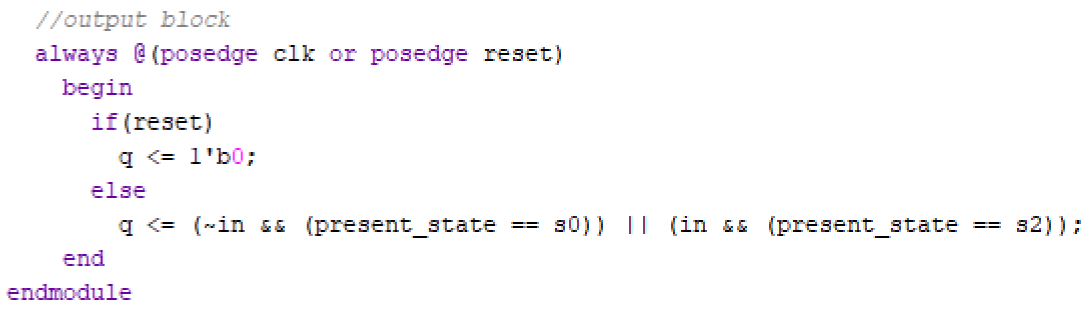


3.5. Design an FSM that detects if a number is divisible by 5 and write testbench to test the same using generator, transactor and driver class to drive the inputs. Use cover groups and cover points along with assertions to complete functional verification.

- **Divisibility_by_5.v**



```
`timescale 1ns / 1ps
module Divisibility_by_5(output reg q, output reg [2:0]present_state, next_state, input clk, reset, in);

  parameter s0 = 3'b000, s1 = 3'b001, s2 = 3'b010, s3 = 3'b011, s4 = 3'b100;

  //present state block
  always @(posedge clk or posedge reset)
    begin
      if(reset)
        present_state <= s0;
      else
        present_state <= next_state;
    end

  //next state and output block
  always @(*)
    begin
      case(present_state)
        s0: next_state <= in ? s1 : s0;
        s1: next_state <= in ? s3 : s2;
        s2: next_state <= in ? s0 : s4;
        s3: next_state <= in ? s2 : s1;
        s4: next_state <= in ? s4 : s3;
        default: next_state <= s0;
      endcase
    end
```



```
  //output block
  always @(posedge clk or posedge reset)
    begin
      if(reset)
        q <= 1'b0;
      else
        q <= (~in && (present_state == s0)) || (in && (present_state == s2));
    end
endmodule
```

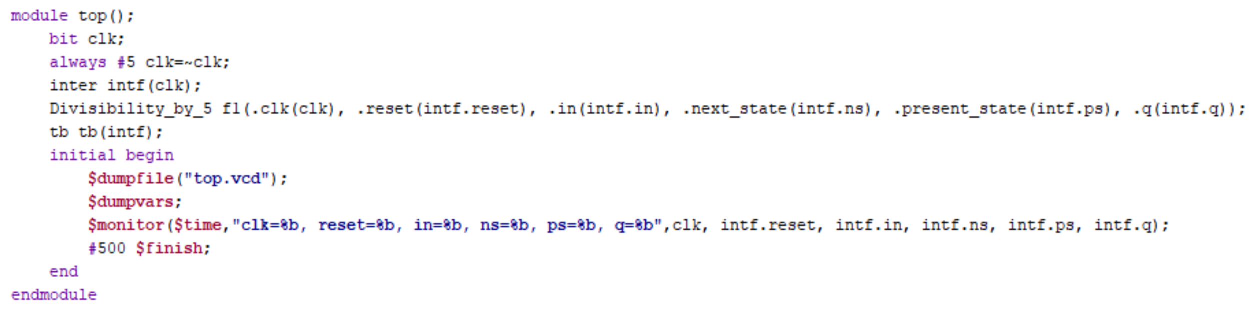
- **inter.sv**

```
interface inter (input bit clk);
    bit in, reset;
    bit [1:0] ns,ps;
    bit q;
    clocking cbi @(posedge clk);
        default input #2ns output #3ns;
        input q;
        input ns,ps;
    endclocking
    modport mod_test(clocking cbi, input clk, output reset, in);

    sequence s1;
        (reset==0) ##[0:$](q==1'b1);
    endsequence

    property p1;
        @(posedge clk) s1;
    endproperty:p1

    a1:assert property (p1)
        $display("output q=1 and input bitstream is divisible by 5, P1 passed");
        else $display("P1 failed");

     property p2;
         @(posedge clk) (reset==1)|-> (q==1'b0);
     endproperty:p2

     a2: assert property(p2)
     $display("fsm reset,P2 passed");
     else $display("P2 failed");

   endinterface
```

- **tb.sv**

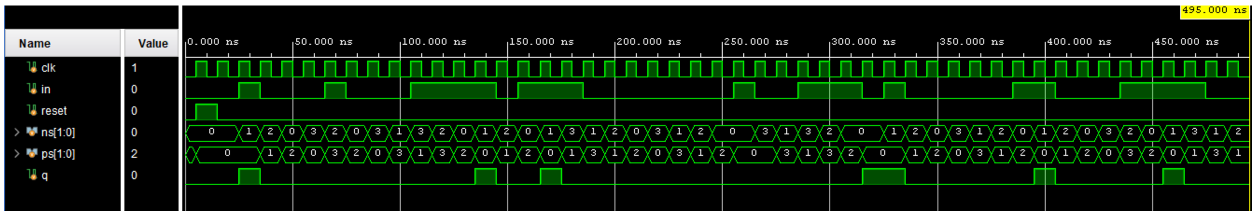
```
class transactor;
    rand bit in;
    constraint c1 {in inside {0,1};}
endclass: transactor

class driver;
    transactor tx;
    virtual inter.mod_test intf_o;
    covergroup cg;
        in: coverpoint intf_o.in{
        bins t0={0};
        bins t1={1};}
        reset: coverpoint intf_o.reset{
        bins r1={0};}
        ns: coverpoint intf_o.cbi.ns{
        bins t2[]={0,1};
        bins t3[]={2,3};}
        ps: coverpoint intf_o.cbi.ps{
        bins t4[]={0,1};
        bins t5[]={2,3};}
        q: coverpoint intf_o.cbi.q;
    endgroup
    function new(virtual inter.mod_test intf_o_new);
        this.intf_o=intf_o_new;
        this.cg=new();
    endfunction: new
    task drive_data();
        tx=new();
        if(!(tx.randomize()))
            $display("%0d : Randomization Error", $time);
        else
        begin
            intf_o.in<=tx.in;
            cg.sample();
        end
    endtask: drive_data
endclass: driver

program tb(inter.mod_test intf_o);
    driver drv;
    initial begin
        #5 intf_o.reset<=1;
        #10 intf_o.reset<=0;
    end
    initial begin
        drv=new(intf_o);
        repeat(50)@(intf_o.cbi)
            drv.drive_data();
    end
endprogram: tb
```

- **top.sv**



```
module top();
    bit clk;
    always #5 clk=~clk;
    inter intf(clk);
    Divisibility_by_5 f1(.clk(clk), .reset(intf.reset), .in(intf.in), .next_state(intf.ns), .present_state(intf.ps), .q(intf.q));
    tb tb(intf);
    initial begin
        $dumpfile("top.vcd");
        $dumpvars;
        $monitor($time,"clk=%b, reset=%b, in=%b, ns=%b, ps=%b, q=%b",clk, intf.reset, intf.in, intf.ns, intf.ps, intf.q);
        #500 $finish;
    end
endmodule
```

- **Simulation Waveforms**



- **Simulation Output**

```
0clk=0, reset=0, in=0, ns=00, ps=00, q=0
                  5clk=1, reset=1, in=0, ns=00, ps=00, q=0
                 10clk=0, reset=1, in=0, ns=00, ps=00, q=0

Time: 15 ns Started: 15 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
P1 failed

Time: 15 ns Started: 15 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:29
fsm reset,P2 passed
                 15clk=1, reset=0, in=0, ns=00, ps=00, q=0
                 20clk=0, reset=0, in=0, ns=00, ps=00, q=0
                 25clk=1, reset=0, in=1, ns=01, ps=00, q=1
                 30clk=0, reset=0, in=1, ns=01, ps=00, q=1

Time: 35 ns Started: 5 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed
```

```
Time: 35 ns Started: 25 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed

Time: 35 ns Started: 35 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed
                 35clk=1, reset=0, in=0, ns=10, ps=01, q=0
                 40clk=0, reset=0, in=0, ns=10, ps=01, q=0
                 45clk=1, reset=0, in=0, ns=00, ps=10, q=0
                 50clk=0, reset=0, in=0, ns=00, ps=10, q=0
                 55clk=1, reset=0, in=0, ns=11, ps=00, q=0
                 60clk=0, reset=0, in=0, ns=11, ps=00, q=0
                 65clk=1, reset=0, in=1, ns=10, ps=11, q=0
                 70clk=0, reset=0, in=1, ns=10, ps=11, q=0
                 75clk=1, reset=0, in=0, ns=00, ps=10, q=0
                 80clk=0, reset=0, in=0, ns=00, ps=10, q=0
                 85clk=1, reset=0, in=0, ns=11, ps=00, q=0
                 90clk=0, reset=0, in=0, ns=11, ps=00, q=0
                 95clk=1, reset=0, in=0, ns=01, ps=11, q=0
                100clk=0, reset=0, in=0, ns=01, ps=11, q=0
                105clk=1, reset=0, in=1, ns=11, ps=01, q=0
                110clk=0, reset=0, in=1, ns=11, ps=01, q=0
                115clk=1, reset=0, in=1, ns=10, ps=11, q=0
                120clk=0, reset=0, in=1, ns=10, ps=11, q=0
                125clk=1, reset=0, in=1, ns=00, ps=10, q=0
                130clk=0, reset=0, in=1, ns=00, ps=10, q=0
                135clk=1, reset=0, in=1, ns=01, ps=00, q=1
                140clk=0, reset=0, in=1, ns=01, ps=00, q=1

Time: 145 ns Started: 45 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed

Time: 145 ns Started: 55 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed
```

```
Time: 145 ns Started: 65 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed

Time: 145 ns Started: 75 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed

Time: 145 ns Started: 85 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed

Time: 145 ns Started: 95 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed

Time: 145 ns Started: 105 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed

Time: 145 ns Started: 115 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed

Time: 145 ns Started: 125 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed

Time: 145 ns Started: 135 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed

Time: 145 ns Started: 145 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed
```

```
                145clk=1, reset=0, in=0, ns=10, ps=01, q=0
                150clk=0, reset=0, in=0, ns=10, ps=01, q=0
                155clk=1, reset=0, in=1, ns=00, ps=10, q=0
                160clk=0, reset=0, in=1, ns=00, ps=10, q=0
                165clk=1, reset=0, in=1, ns=01, ps=00, q=1
                170clk=0, reset=0, in=1, ns=01, ps=00, q=1

Time: 175 ns Started: 155 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed

Time: 175 ns Started: 165 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed

Time: 175 ns Started: 175 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed
                175clk=1, reset=0, in=1, ns=11, ps=01, q=0
                180clk=0, reset=0, in=1, ns=11, ps=01, q=0
                185clk=1, reset=0, in=0, ns=01, ps=11, q=0
                190clk=0, reset=0, in=0, ns=01, ps=11, q=0
                195clk=1, reset=0, in=0, ns=10, ps=01, q=0
                200clk=0, reset=0, in=0, ns=10, ps=01, q=0
                205clk=1, reset=0, in=0, ns=00, ps=10, q=0
                210clk=0, reset=0, in=0, ns=00, ps=10, q=0
                215clk=1, reset=0, in=0, ns=11, ps=00, q=0
                220clk=0, reset=0, in=0, ns=11, ps=00, q=0
                225clk=1, reset=0, in=0, ns=01, ps=11, q=0
                230clk=0, reset=0, in=0, ns=01, ps=11, q=0
                235clk=1, reset=0, in=0, ns=10, ps=01, q=0
                240clk=0, reset=0, in=0, ns=10, ps=01, q=0
                245clk=1, reset=0, in=0, ns=00, ps=10, q=0
                250clk=0, reset=0, in=0, ns=00, ps=10, q=0
                255clk=1, reset=0, in=1, ns=00, ps=00, q=0
                260clk=0, reset=0, in=1, ns=00, ps=00, q=0
                265clk=1, reset=0, in=0, ns=11, ps=00, q=0
                270clk=0, reset=0, in=0, ns=11, ps=00, q=0
                275clk=1, reset=0, in=0, ns=01, ps=11, q=0
                280clk=0, reset=0, in=0, ns=01, ps=11, q=0
                285clk=1, reset=0, in=1, ns=11, ps=01, q=0
```

```
290clk=0, reset=0, in=1, ns=11, ps=01, q=0
295clk=1, reset=0, in=1, ns=10, ps=11, q=0
300clk=0, reset=0, in=1, ns=10, ps=11, q=0
305clk=1, reset=0, in=1, ns=00, ps=10, q=0
310clk=0, reset=0, in=1, ns=00, ps=10, q=0
315clk=1, reset=0, in=0, ns=00, ps=00, q=1
320clk=0, reset=0, in=0, ns=00, ps=00, q=1

Time: 325 ns Started: 185 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed

Time: 325 ns Started: 195 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed

Time: 325 ns Started: 205 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed

Time: 325 ns Started: 215 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed

Time: 325 ns Started: 225 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed

Time: 325 ns Started: 235 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed

Time: 325 ns Started: 245 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed
```

```
Time: 325 ns Started: 255 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed

Time: 325 ns Started: 265 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed

Time: 325 ns Started: 275 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed

Time: 325 ns Started: 285 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed

Time: 325 ns Started: 295 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed

Time: 325 ns Started: 305 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed

Time: 325 ns Started: 315 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed

Time: 325 ns Started: 325 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed
          325clk=1, reset=0, in=1, ns=01, ps=00, q=1
          330clk=0, reset=0, in=1, ns=01, ps=00, q=1
```

```
Time: 335 ns Started: 335 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed
          335clk=1, reset=0, in=0, ns=10, ps=01, q=0
          340clk=0, reset=0, in=0, ns=10, ps=01, q=0
          345clk=1, reset=0, in=0, ns=00, ps=10, q=0
          350clk=0, reset=0, in=0, ns=00, ps=10, q=0
          355clk=1, reset=0, in=0, ns=11, ps=00, q=0
          360clk=0, reset=0, in=0, ns=11, ps=00, q=0
          365clk=1, reset=0, in=0, ns=01, ps=11, q=0
          370clk=0, reset=0, in=0, ns=01, ps=11, q=0
          375clk=1, reset=0, in=0, ns=10, ps=01, q=0
          380clk=0, reset=0, in=0, ns=10, ps=01, q=0
          385clk=1, reset=0, in=1, ns=00, ps=10, q=0
          390clk=0, reset=0, in=1, ns=00, ps=10, q=0
          395clk=1, reset=0, in=1, ns=01, ps=00, q=1
          400clk=0, reset=0, in=1, ns=01, ps=00, q=1

Time: 405 ns Started: 345 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed

Time: 405 ns Started: 355 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed

Time: 405 ns Started: 365 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed

Time: 405 ns Started: 375 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed

Time: 405 ns Started: 385 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed
```

```
Time: 405 ns Started: 395 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed

Time: 405 ns Started: 405 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed
        405clk=1, reset=0, in=0, ns=10, ps=01, q=0
        410clk=0, reset=0, in=0, ns=10, ps=01, q=0
        415clk=1, reset=0, in=0, ns=00, ps=10, q=0
        420clk=0, reset=0, in=0, ns=00, ps=10, q=0
        425clk=1, reset=0, in=0, ns=11, ps=00, q=0
        430clk=0, reset=0, in=0, ns=11, ps=00, q=0
        435clk=1, reset=0, in=1, ns=10, ps=11, q=0
        440clk=0, reset=0, in=1, ns=10, ps=11, q=0
        445clk=1, reset=0, in=1, ns=00, ps=10, q=0
        450clk=0, reset=0, in=1, ns=00, ps=10, q=0
        455clk=1, reset=0, in=1, ns=01, ps=00, q=1
        460clk=0, reset=0, in=1, ns=01, ps=00, q=1

Time: 465 ns Started: 415 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed

Time: 465 ns Started: 425 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed

Time: 465 ns Started: 435 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed

Time: 465 ns Started: 445 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed
```

```
Time: 465 ns Started: 455 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed

Time: 465 ns Started: 465 ns Scope: /top/intf File: D:/Semester Materials/VI
Semester/SV/AAT/AAT 2/Divisibility by 5/Divisibility by 5.srcs/sim_1/new/inter.sv
Line:21
output q=1 and input bitstream is divisible by 5, P1 passed
            465clk=1, reset=0, in=1, ns=11, ps=01, q=0
            470clk=0, reset=0, in=1, ns=11, ps=01, q=0
            475clk=1, reset=0, in=0, ns=01, ps=11, q=0
            480clk=0, reset=0, in=0, ns=01, ps=11, q=0
            485clk=1, reset=0, in=0, ns=10, ps=01, q=0
            490clk=0, reset=0, in=0, ns=10, ps=01, q=0
$finish called at time : 495 ns :
```