# Cross Coverage & SVA Assignment

## Assignment4

By: Mohamed Sayed Mohamed Soliman

# Question 1

## ➢ ALSU

Add the following cross coverage in the class of the ALSU of the last assignment:

1. When the ALU is addition or multiplication, A and B should have taken all permutations of maxpos, maxneg and zero.
2. When the ALU is addition, c_in should have taken 0 or 1
3. When the ALSU is shifting, then shift_in must take 0 or 1
4. When the ALSU is shifting or rotating, then direction must take 0 or 1
5. When the ALSU is OR or XOR and red_op_A is asserted, then A took all walking one patterns (001, 010, and 100) while B is taking the value 0
6. When the ALSU is OR or XOR and red_op_B is asserted, then B took all walking one patterns (001, 010, and 100) while A is taking the value 0
7. Covering the invalid case: reduction operation is activated while the opcode is not OR or XOR

1. Code Design

```
module ALSU(A, B, cin, serial_in, red_op_A, red_op_B, opcode, bypass_A, bypass_B, clk, rst, direction, leds, out);
parameter INPUT_PRIORITY = "A";
parameter FULL_ADDER = "ON";
input clk, rst, cin, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
input [2:0] opcode;
input signed [2:0] A, B;    // first bug --> we must put it signed
output reg [15:0] leds;
output reg signed [5:0] out;  // second bug --> we must put it signed

reg cin_reg, red_op_A_reg, red_op_B_reg, bypass_A_reg, bypass_B_reg, direction_reg, serial_in_reg;
reg [2:0] opcode_reg;
reg signed  [2:0]  A_reg, B_reg;
wire invalid_red_op, invalid_opcode, invalid;

//Invalid handling
assign invalid_red_op = (red_op_A_reg | red_op_B_reg) & (opcode_reg[1] | opcode_reg[2]);
assign invalid_opcode = opcode_reg[1] & opcode_reg[2];
assign invalid = invalid_red_op | invalid_opcode;

//Registering input signals
always @(posedge clk or posedge rst) begin
  if(rst) begin
      cin_reg <= 0;
      red_op_B_reg <= 0;
      red_op_A_reg <= 0;
      bypass_B_reg <= 0;
      bypass_A_reg <= 0;
      direction_reg <= 0;
      serial_in_reg <= 0;
      opcode_reg <= 0;
      A_reg <= 0;
      B_reg <= 0;
```

```
      end else begin
          cin_reg <= cin;
          red_op_B_reg <= red_op_B;
          red_op_A_reg <= red_op_A;
          bypass_B_reg <= bypass_B;
          bypass_A_reg <= bypass_A;
          direction_reg <= direction;
          serial_in_reg <= serial_in;
          opcode_reg <= opcode;
          A_reg <= A;
          B_reg <= B;
      end
  end

  //leds output blinking
  always @(posedge clk or posedge rst) begin
      if(rst) begin
          leds <= 0;
      end else begin
          if (invalid)
              leds <= ~leds;
          else
              leds <= 0;
      end
  end
```

```
//ALSU output processing
always @(posedge clk or posedge rst) begin
  if(rst) begin
    out <= 0;
  end
  else begin
     if (bypass_A_reg && bypass_B_reg)
      out <= (INPUT_PRIORITY == "A")? A_reg: B_reg;
    else if (bypass_A_reg)
      out <= A_reg;
    else if (bypass_B_reg)
      out <= B_reg;
      else  if (invalid)
        out <= 0;
    else begin
        case (opcode_reg) //  third bug is to used the opcode_reg  not the opcode
          3'h0: begin
            if (red_op_A_reg && red_op_B_reg)
              out <= (INPUT_PRIORITY == "A")? |A_reg: |B_reg;  // third bug is to replace AND with OR
            else if (red_op_A_reg)
              out <= |A_reg;
            else if (red_op_B_reg)
              out <= |B_reg;
            else
              out <= A_reg | B_reg;
          end
          3'h1: begin
            if (red_op_A_reg && red_op_B_reg)
              out <= (INPUT_PRIORITY == "A")? ^A_reg: ^B_reg;  // fourth bug is to replace OR with XOR
            else if (red_op_A_reg)
              out <= ^A_reg;
            else if (red_op_B_reg)
              out <= ^B_reg;
            else
              out <= A_reg ^ B_reg;
          end
          3'h2:begin
                if(FULL_ADDER == "ON")  // fifth bug to add Cin operation in case of full adde
                    out <= A_reg + B_reg + cin_reg ;
                else
                    out <= A_reg + B_reg ;
                end
          3'h3: out <= A_reg * B_reg;
          3'h4: begin
            if (direction_reg)
              out <= {out[4:0], serial_in_reg};
            else
              out <= {serial_in_reg, out[5:1]};
          end
```

```
                3'h5:  begin
                    if (direction_reg)
                      out <= {out[4:0], out[5]};
                    else
                      out <= {out[0], out[5:1]};
                end
            default : out <= 0 ;
             endcase
        end
      end
    end

    endmodule
```

## 2. Verification plan

| Label | Description | Stimulus Generation | Functional Coverage | Functionality Check |
|---|---|---|---|---|
| ALSU_1 | When the reset is asserted, the output value should be low | Directed at the start of the simulation | - | A checker in the testbench to make sure the output is correct |
| ALSU_2 | When the invalid is asserted, the output value should be low | Randomization | Cover all values of A | A checker in the testbench to make sure the output is correct |
| ALSU_3 | When the bypass_A is asserted, bypass_B is asserted and INPUT_PRIORITY = "A" the output count_out should take the A | Randomization | Cover all values of B | A checker in the testbench to make sure the output is correct |
| ALSU_4 | When the bypass_A is asserted and bypass_B is deasserted the output count_out should take the A | Randomization | Cover all values of opcode, and transition bin from 0=>1=>2=>3=>4=>5 | A checker in the testbench to make sure the output is correct |
| ALSU_5 | When the bypass_B is asserted and bypass_A is deasserted the output count_out should take the A | Randomization | - | A checker in the testbench to make sure the output is correct |
| ALSU_6 | Test all value of opcode and show the output in all case | Randomization | - | A checker in the testbench to make sure the output is correct |

## 3. ALSU Package

```
package pack_ALSU;
  typedef enum logic [2:0] {OR , XOR , ADD , MULT , SHIFT , ROTATE , INVALID_6 , INVALID_7 } reg_e ;
  typedef enum  {Or , Xor , Add , Mult , Shift ,Rotate} opcode_valid_e ;
  localparam MAXPOS = 3  ;
  localparam MAXNEG = -4 ;
  localparam zero = 0 ;

      class cla_ALSU ;
        rand  logic signed [2:0] A, B ;
        rand  reg_e opcode ;
        rand  logic       rst, cin, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
        bit clk ;
        rand opcode_valid_e opcodes_array[6];

        constraint rst_n {rst dist {0:/99 , 1:/1  } ; }

        constraint input_A {
          if ( (opcode == ADD ) || (opcode == MULT ) )
          {
            A dist { MAXPOS:/25 ,  MAXNEG:/25 , zero:/25 , {3'b010 , 3'b001 , 3'b111 , 3'b110 , 3'b101}:/25 } ;
          }
          else if (((opcode == XOR ) || (opcode == OR )) && (red_op_A == 1) && (red_op_B == 0)  )
          {
            B == 0 ;
            A dist {3'b001:/30 , 3'b010:/30 , 3'b100:/30 , 3'b000:/5 , 3'b011:/5 , 3'b101:/5 , 3'b110:/5 , 3'b111:/5 } ;
          }
          else
          {
            A inside { [MAXNEG : MAXPOS] } ;
          }
        }
```

```
constraint input_B {
 if ( (opcode == ADD ) || (opcode == MULT ) )
 {
   B dist { MAXPOS:/25 ,  MAXNEG:/25 , zero:/25 , {3'b010 , 3'b001 , 3'b111 , 3'b110 , 3'b101}:/25 } ;
 }
 else if (((opcode == XOR ) || (opcode == OR )) && (red_op_B == 1) && (red_op_A == 0) )
 {
   A == 0 ;
   B dist {3'b001:/30 , 3'b010:/30 , 3'b100:/30 , 3'b000:/5 , 3'b011:/5 , 3'b101:/5 , 3'b110:/5 , 3'b111:/5 } ;
 }
 else
 {
   B inside { [MAXNEG : MAXPOS]  } ;
 }
}

constraint input_opcode {opcode dist {[0:3]:/45 , [4:5]:/50 ,[6:7]:/1  } ; }

constraint input_bypass_A {bypass_A dist {1:=90 , 0:=10  } ; }
constraint input_bypass_B {bypass_B dist {1:=90 , 0:=10  } ; }

constraint input_red_op_A {red_op_A dist {1:/10 , 0:/90  } ; }
constraint input_red_op_B {red_op_B dist {1:/10 , 0:/90  } ; }
```

```
constraint c_fixed_array {
 foreach(opcodes_array[i])
 {
 foreach(opcodes_array[j])
 if(i != j)
 opcodes_array[i] != opcodes_array[j];
 }
 }

covergroup cvr_gp @(posedge clk) ;
 A_cp : coverpoint A {
     bins A_data_0 = {0} ;
     bins A_data_max = {MAXPOS} ;
     bins A_data_min = {MAXNEG} ;
     bins A_data_default = default ;
     bins A_data_walkingones[] = {1, 2, -4} iff (red_op_A && !red_op_B);
 }

 B_cp : coverpoint B {
     bins B_data_0 = {0} ;
     bins B_data_max = {MAXPOS} ;
     bins B_data_min = {MAXNEG} ;
     bins B_data_default = default ;
     bins B_data_walkingones[] = {1, 2, -4} iff (!red_op_A && red_op_B);
 }
 ALU_cp : coverpoint opcode {
     bins Bins_shift[] = {SHIFT , ROTATE} ;
     bins Bins_arith[] = {ADD , MULT} ;
     bins Bins_bitwise[] = {OR , XOR} ;
     illegal_bins Bins_invalid = {INVALID_6 , INVALID_7} ;
     bins Bins_trans = (OR => XOR => ADD => MULT => SHIFT => Rotate);
 }
 cin_cp : coverpoint cin {
     bins cin_data = {0 , 1} ;
 }
 direction_cp : coverpoint direction {
     bins direction_data = {0 , 1} ;
 }
 serial_in_cp : coverpoint serial_in {
     bins serial_in_data = {0 , 1} ;
 }
 red_op_A_cp : coverpoint red_op_A {
     bins red_op_A_LOW_data = {0} ;
     bins red_op_A_HIGH_data = {1} ;
 }
 red_op_B_cp : coverpoint red_op_B {
     bins red_op_B_LOW_data = {0} ;
     bins red_op_B_HIGH_data = {1} ;
 }
```

```
add_mult_cp1 : cross  A_cp , B_cp , ALU_cp
        {
            bins zero_A_add = binsof(ALU_cp.Bins_arith) && binsof(A_cp.A_data_0) && binsof(B_cp.B_data_0) ;
            bins max_nega_add = binsof(ALU_cp.Bins_arith) && binsof(A_cp.A_data_min) && binsof(B_cp.B_data_min) ;
            bins max_pos_add = binsof(ALU_cp.Bins_arith) && binsof(A_cp.A_data_max) && binsof(B_cp.B_data_max) ;
            option.cross_auto_bin_max = 0 ;
        }
opcode_cp2 : cross  cin_cp , direction_cp , serial_in_cp , ALU_cp ,red_op_A_cp , red_op_B_cp
        {
            bins cin_add = binsof(cin_cp.cin_data) && binsof(ALU_cp.Bins_arith) intersect {ADD} ;
            bins serialin_shift = binsof(serial_in_cp.serial_in_data) && binsof(ALU_cp) intersect {SHIFT} ;
            bins direction_shift_rota = binsof(direction_cp) && binsof(ALU_cp.Bins_shift) ;
            option.cross_auto_bin_max = 0 ;
        }
or_xor_cp3 : cross  A_cp , B_cp , ALU_cp , red_op_A_cp , red_op_B_cp
    {
        bins or_xor_data_A  = binsof(ALU_cp.Bins_bitwise) && binsof(A_cp.A_data_walkingones) && binsof(B_cp.B_data_0) && binsof(red_op_B_cp.red_op_B_LOW_data) && binso
        bins or_xor_data_B  = binsof(ALU_cp.Bins_bitwise) && binsof(B_cp.B_data_walkingones) && binsof(A_cp.A_data_0) && binsof(red_op_A_cp.red_op_A_LOW_data) && binso
        option.cross_auto_bin_max = 0 ;
    }
INVALID_cp4 : cross  ALU_cp , red_op_A_cp , red_op_B_cp
        {
            bins Bins_shift_data  = binsof(ALU_cp.Bins_shift) && ( binsof(red_op_B_cp.red_op_B_HIGH_data) || binsof(red_op_A_cp.red_op_A_HIGH_data) ) ;
            bins Bins_arith_data  = binsof(ALU_cp.Bins_arith) && ( binsof(red_op_B_cp.red_op_B_HIGH_data) || binsof(red_op_A_cp.red_op_A_HIGH_data) ) ;

            option.cross_auto_bin_max = 0 ;
        }
endgroup
function new() ;
  cvr_gp = new() ;
endfunction

function void print ();
 $display("A = 0h%0h  , B = 0h%0h , opcode = 0h%0h , rst = 0h%0h , cin = 0h%0h  " , this.A , this.B , this.opcode , this.rst , this.cin  ) ;
 endfunction

 endclass

endpackage
```

## 4. ALSU Testbench

```
import pack_ALSU::*;

module ALSU_tb();

  parameter WIDTH = 3;
  parameter INPUT_PRIORITY = "A";
  parameter FULL_ADDER = "ON";

  logic signed [2:0] A, B ;
  logic [2:0] opcode ;
  logic       clk, rst, cin, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
  logic [15:0] leds;
  logic signed [5:0] out;

  integer i  , j;
  integer correct_counter = 0 ;
  integer error_counter = 0 ;
  logic [5:0] old_count_out ;
  logic invalid ;
  logic test ;

  opcode_valid_e opcodes_array_tb[6];

  ALSU #(.INPUT_PRIORITY(INPUT_PRIORITY) , .FULL_ADDER(FULL_ADDER)  ) dut (.*);

  cla_ALSU trans1 = new() ;
  initial begin
    clk = 0 ;
  forever
  begin
  #20 clk = ~clk ;
  trans1.clk =  clk ;
  end

end
```

```
  initial begin
    cin       = 0 ;
    red_op_A  = 0 ;
    red_op_B  = 0 ;
    bypass_A  = 0 ;
    bypass_B  = 0 ;
    direction = 0 ;
    serial_in = 0 ;
    A         = 0 ;
    B         = 0 ;
    opcode    = 0 ;
    rest ();
    #10 ;
    trans1.c_fixed_array.constraint_mode(0) ;
    repeat(10000) begin
    assert(trans1.randomize());

      A         = trans1.A         ;
      B         = trans1.B         ;
      cin       = trans1.cin       ;
      red_op_A  = trans1.red_op_A  ;
      red_op_B  = trans1.red_op_B  ;
      bypass_A  = trans1.bypass_A  ;
      bypass_B  = trans1.bypass_B  ;
      direction = trans1.direction ;
      serial_in = trans1.serial_in ;
      opcode    = trans1.opcode    ;
      rst       = trans1.rst       ;

      sample_data () ;
      golden_model ();

```

```
    end
    trans1.constraint_mode(0) ;
    trans1.c_fixed_array.constraint_mode(1) ;

    rst       = 0 ;
    red_op_A  = 0 ;
    red_op_B  = 0 ;
    bypass_A  = 0 ;
    bypass_B  = 0 ;

    repeat(10000) begin
     assert(trans1.randomize());

      A         = trans1.A         ;
      B         = trans1.B         ;
      cin       = trans1.cin       ;
      red_op_A  = trans1.red_op_A  ;
      red_op_B  = trans1.red_op_B  ;
      bypass_A  = trans1.bypass_A  ;
      bypass_B  = trans1.bypass_B  ;
      direction = trans1.direction ;
      serial_in = trans1.serial_in ;
     // opcode    = trans1.opcode    ;
      rst       = trans1.rst       ;
      for(int j = 0; j < 6 ; j =j+1) begin
        opcodes_array_tb[j] = trans1.opcodes_array[j];
        opcode = opcodes_array_tb[j];
        #45 sample_data();
        end
        sample_data();
        golden_model();

```

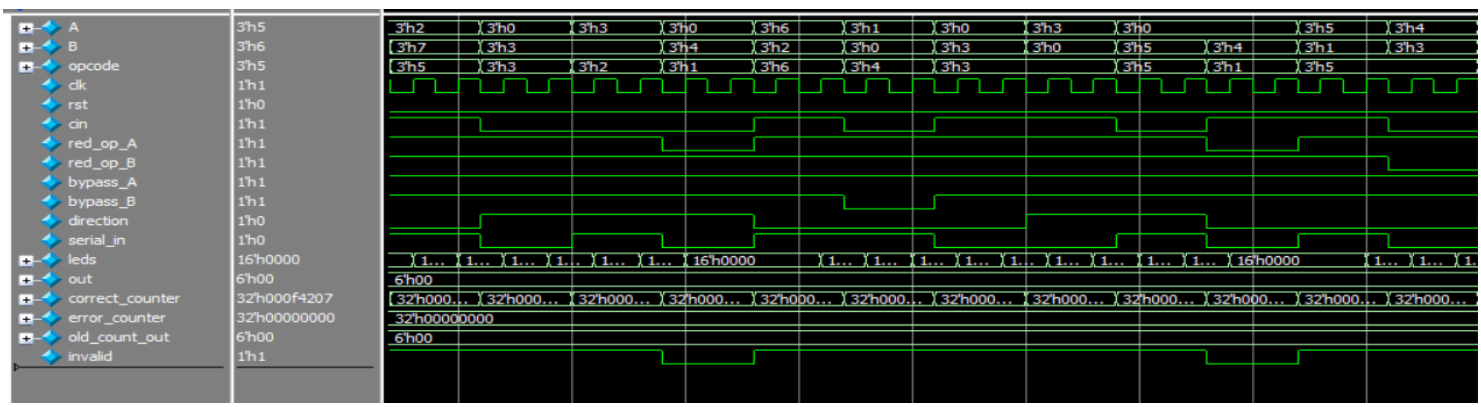
```
        end
        rst       = 0 ;
        red_op_A  = 0 ;
        red_op_B  = 0 ;
        bypass_A  = 0 ;
        bypass_B  = 0 ;

        trans1.cvr_gp.start();
        trans1.opcode = OR;
        opcode = trans1.opcode;
        #20
        trans1.cvr_gp.sample();
        #20
        trans1.opcode = XOR     ;
        opcode     = trans1.opcode     ;
        #20
        trans1.cvr_gp.sample();
        #20
        trans1.opcode = ADD     ;
        opcode     = trans1.opcode     ;
        #20
        trans1.cvr_gp.sample();
        #20
        trans1.opcode = MULT     ;
        opcode     = trans1.opcode     ;
        #20
        trans1.cvr_gp.sample();
        #20
        trans1.opcode = SHIFT     ;
        opcode     = trans1.opcode     ;
        #20
        trans1.cvr_gp.sample();
        #20
        trans1.opcode = ROTATE     ;
        opcode     = trans1.opcode     ;
        #20
        trans1.cvr_gp.sample();
        #40
      $display ("error_counter = %0d " ,error_counter );
      $display ("correct_counter = %0d " ,correct_counter );
      #100 ;

    $stop ;

end
task check_result (input logic signed [5:0] expected_result_out , input [15:0] expected_result_leds );
  @(negedge clk );
  @(negedge clk );
  if( (expected_result_out != out) && (expected_result_leds != leds) )
      begin
          $display (":error");
          old_count_out = out ;
          error_counter = error_counter +1 ;
          test = 1 ;

      end
      else
          begin
              correct_counter = correct_counter + 1 ;
              old_count_out = out ;
              test = 0 ;
          end
endtask
  task sample_data () ;
      if(rst || bypass_A || bypass_B)
      begin
        trans1.cvr_gp.stop();
      end
      else
      begin
        trans1.cvr_gp.start();
        trans1.cvr_gp.sample();
      end

  endtask
```

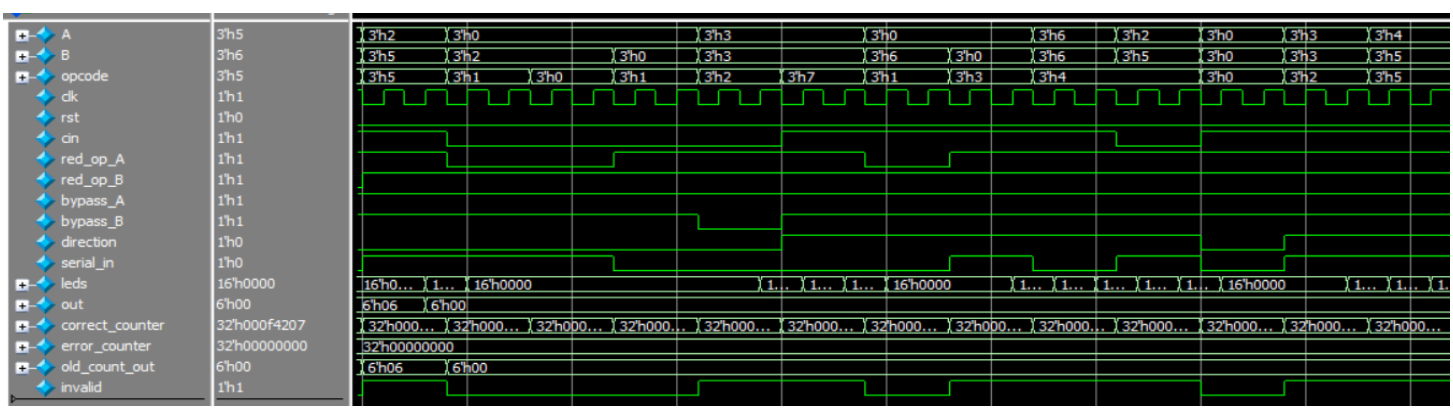
```
task golden_model ();
  invalid = (((red_op_A | red_op_B) & (opcode[1] | opcode[2])) | (opcode[1] & opcode[2]) ) ;
  if (rst)
  check_result(0,0);
  else if(bypass_A && bypass_B)
  begin
    if (INPUT_PRIORITY == "A")
    check_result(A,0);
      else
      check_result(B,0);
  end
  else if (bypass_A)
  check_result(A,0);
  else if (bypass_B)
  check_result(B,0);
  else if (invalid)
  check_result(0,'hffff);
  else begin
    case (opcode)
      3'h0: begin
        if (red_op_A && red_op_B)
        begin
          if (INPUT_PRIORITY == "A")
          check_result(|A,0);
            else
            check_result(|B,0);
        end
      3'h1: begin
        if (red_op_A && red_op_B)
        begin
          if (INPUT_PRIORITY == "A")
          check_result(^A,0);
            else
            check_result(^B,0);
        end
        else if (red_op_A)
        check_result(^A,0);
        else if (red_op_B)
        check_result(^B,0);
        else
        check_result(A^B,0);
      end
      3'h2: begin
        if(FULL_ADDER == "ON")
            check_result(A+B+cin,0);
        else
            check_result(A+B,0);
        end
      3'h3: check_result(A*B,0);
      3'h4: begin
        if (direction)
        check_result({old_count_out[4:0], serial_in},0);
        else
        check_result({serial_in, old_count_out[5:1]},0);
      end
      3'h5: begin
        if (direction)
        check_result({old_count_out[4:0], old_count_out[5]},0);
        else
        check_result({old_count_out[0], old_count_out[5:1]},0);
      end
    endcase
end
endtask
  task rest ();
    rst = 1 ;
    #20
    rst = 0 ;

endtask

endmodule
```

5. Do File

```
vlib work
vlog ALSU.v ALSU_tb.sv +cover -covercells
vsim -voptargs=+acc work.ALSU_tb -cover
add wave *
coverage save ALSU_tb.ucdb -onexit
run -all
```

6. Waveform




7. Transcript

```
#
# error_counter = 0
# correct_counter = 1000000
```

```
#    Time: 3983980 ns  Iteration: 0  Region: /pack_ALSU::cla_ALSU::#cvr_gp#
# ** Error: (vsim-8565) Illegal state bin was hit at value=INVALID_6. The bin counter for the illegal bin '\/pack_ALSU::cla_ALSU::cvr_gp .ALU_cp.Bins_invalid' is 2467.
#    Time: 3986300 ns  Iteration: 0  Region: /pack_ALSU::cla_ALSU::#cvr_gp#
# ** Error: (vsim-8565) Illegal state bin was hit at value=INVALID_6. The bin counter for the illegal bin '\/pack_ALSU::cla_ALSU::cvr_gp .ALU_cp.Bins_invalid' is 2468.
#    Time: 3986340 ns  Iteration: 0  Region: /pack_ALSU::cla_ALSU::#cvr_gp#
# ** Error: (vsim-8565) Illegal state bin was hit at value=INVALID_6. The bin counter for the illegal bin '\/pack_ALSU::cla_ALSU::cvr_gp .ALU_cp.Bins_invalid' is 2469.
#    Time: 3986380 ns  Iteration: 0  Region: /pack_ALSU::cla_ALSU::#cvr_gp#
# ** Error: (vsim-8565) Illegal state bin was hit at value=INVALID_6. The bin counter for the illegal bin '\/pack_ALSU::cla_ALSU::cvr_gp .ALU_cp.Bins_invalid' is 2470.
#    Time: 3986420 ns  Iteration: 0  Region: /pack_ALSU::cla_ALSU::#cvr_gp#
# ** Error: (vsim-8565) Illegal state bin was hit at value=INVALID_6. The bin counter for the illegal bin '\/pack_ALSU::cla_ALSU::cvr_gp .ALU_cp.Bins_invalid' is 2471.
#    Time: 3986460 ns  Iteration: 0  Region: /pack_ALSU::cla_ALSU::#cvr_gp#
# ** Error: (vsim-8565) Illegal state bin was hit at value=INVALID_6. The bin counter for the illegal bin '\/pack_ALSU::cla_ALSU::cvr_gp .ALU_cp.Bins_invalid' is 2472.
#    Time: 3986500 ns  Iteration: 0  Region: /pack_ALSU::cla_ALSU::#cvr_gp#
# ** Error: (vsim-8565) Illegal state bin was hit at value=INVALID_6. The bin counter for the illegal bin '\/pack_ALSU::cla_ALSU::cvr_gp .ALU_cp.Bins_invalid' is 2473.
#    Time: 3986540 ns  Iteration: 0  Region: /pack_ALSU::cla_ALSU::#cvr_gp#
# ** Error: (vsim-8565) Illegal state bin was hit at value=INVALID_6. The bin counter for the illegal bin '\/pack_ALSU::cla_ALSU::cvr_gp .ALU_cp.Bins_invalid' is 2474.
#    Time: 3986580 ns  Iteration: 0  Region: /pack_ALSU::cla_ALSU::#cvr_gp#
# ** Error: (vsim-8565) Illegal state bin was hit at value=INVALID_6. The bin counter for the illegal bin '\/pack_ALSU::cla_ALSU::cvr_gp .ALU_cp.Bins_invalid' is 2475.
#    Time: 3989180 ns  Iteration: 0  Region: /pack_ALSU::cla_ALSU::#cvr_gp#
# ** Error: (vsim-8565) Illegal state bin was hit at value=INVALID_6. The bin counter for the illegal bin '\/pack_ALSU::cla_ALSU::cvr_gp .ALU_cp.Bins_invalid' is 2476.
#    Time: 3989220 ns  Iteration: 0  Region: /pack_ALSU::cla_ALSU::#cvr_gp#
# ** Error: (vsim-8565) Illegal state bin was hit at value=INVALID_6. The bin counter for the illegal bin '\/pack_ALSU::cla_ALSU::cvr_gp .ALU_cp.Bins_invalid' is 2477.
#    Time: 3989260 ns  Iteration: 0  Region: /pack_ALSU::cla_ALSU::#cvr_gp#
# ** Error: (vsim-8565) Illegal state bin was hit at value=INVALID_6. The bin counter for the illegal bin '\/pack_ALSU::cla_ALSU::cvr_gp .ALU_cp.Bins_invalid' is 2478.
#    Time: 3989300 ns  Iteration: 0  Region: /pack_ALSU::cla_ALSU::#cvr_gp#
# ** Error: (vsim-8565) Illegal state bin was hit at value=INVALID_6. The bin counter for the illegal bin '\/pack_ALSU::cla_ALSU::cvr_gp .ALU_cp.Bins_invalid' is 2479.
#    Time: 3989340 ns  Iteration: 0  Region: /pack_ALSU::cla_ALSU::#cvr_gp#
# ** Error: (vsim-8565) Illegal state bin was hit at value=INVALID_6. The bin counter for the illegal bin '\/pack_ALSU::cla_ALSU::cvr_gp .ALU_cp.Bins_invalid' is 2480.
#    Time: 3989380 ns  Iteration: 0  Region: /pack_ALSU::cla_ALSU::#cvr_gp#
# ** Error: (vsim-8565) Illegal state bin was hit at value=INVALID_6. The bin counter for the illegal bin '\/pack_ALSU::cla_ALSU::cvr_gp .ALU_cp.Bins_invalid' is 2481.
#    Time: 3989420 ns  Iteration: 0  Region: /pack_ALSU::cla_ALSU::#cvr_gp#
# ** Error: (vsim-8565) Illegal state bin was hit at value=INVALID_7. The bin counter for the illegal bin '\/pack_ALSU::cla_ALSU::cvr_gp .ALU_cp.Bins_invalid' is 2482.
#    Time: 3990100 ns  Iteration: 0  Region: /pack_ALSU::cla_ALSU::#cvr_gp#
```

⇨ All This Errors Resulting from Hitting the Illegal Bins.

## 8. Code Coverage

```
==============================================================================
=== File: ALSU.v
==============================================================================
Statement Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Stmts                           49        49         0     100.0

================================Statement Details================================

Statement Coverage for file ALSU.v --

    1                                           module ALSU(A, B, cin, serial_in, red_op_A, red_op_B, opcode, by
    2                                           parameter INPUT_PRIORITY = "A";
    3                                           parameter FULL_ADDER = "ON";
    4                                           input clk, rst, cin, red_op_A, red_op_B, bypass_A, bypass_B, dir
    5                                           input [2:0] opcode;
    6                                           input signed [2:0] A, B;
    7                                           output reg [15:0] leds;
    8                                           output reg signed [5:0] out;
    9
    10                                          reg cin_reg, red_op_A_reg, red_op_B_reg, bypass_A_reg, bypass_B_
    11                                          reg [2:0] opcode_reg, A_reg, B_reg;
    12
    13                                          wire invalid_red_op, invalid_opcode, invalid;
    14
    15                                          //Invalid handling
    16              1                    896366 assign invalid_red_op = (red_op_A_reg | red_op_B_reg) & (opcode_
    17              1                    856664 assign invalid_opcode = opcode_reg[1] & opcode_reg[2];
    18              1                    494649 assign invalid = invalid_red_op | invalid_opcode;
    19
    20                                          //Registering input signals
    21              1                   1997369 always @(posedge clk or posedge rst) begin
    22                                            if(rst) begin
```

```
Branch Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Branches                        30        30         0     100.0

================================Branch Details================================

Branch Coverage for file ALSU.v --

------------------------------------IF Branch------------------------------------
    22                                   1997369     Count coming in to IF
    22              1                      19911       if(rst) begin
    33              1                    1977458       end else begin
Branch totals: 2 hits of 2 branches = 100.0%

------------------------------------IF Branch------------------------------------
    49                                   2009908     Count coming in to IF
    49              1                      29921       if(rst) begin
    52              1                    1513995           if (invalid)
    54              1                     465992           else
Branch totals: 3 hits of 3 branches = 100.0%
```

```
FSM Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    FSMs                                                      100.0
        States                       0         0         0     100.0
        Transitions                  0         0         0     100.0
Toggle Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Toggle Bins                    118       118         0     100.0

================================Toggle Details================================

Toggle Coverage for File ALSU.v --

        Line                                    Node      1H->0L      0L->1H  "Coverage"
--------------------------------------------------------------------------------------------
            4                               serial_in           1           1      100.00
            4                                     rst           1           1      100.00
```

## 9. function Coverage

```
Covergroup instance \/pack_ALSU::cla_ALSU::cvr_gp
                                                    100.0%        100    Covered
    covered/total bins:                                 36         36
    missing/total bins:                                  0         36
    % Hit:                                          100.0%        100
    Coverpoint A_cp                                 100.0%        100    Covered
        covered/total bins:                              6          6
        missing/total bins:                              0          6
        % Hit:                                      100.0%        100
        bin A_data_0                                  1408          1    Covered
        bin A_data_max                                1371          1    Covered
        bin A_data_min                                1290          1    Covered
        bin A_data_walkingones[-4]                     337          1    Covered
        bin A_data_walkingones[1]                      261          1    Covered
        bin A_data_walkingones[2]                      334          1    Covered
        default bin A_data_default                    3780               Occurred
    Coverpoint B_cp                                 100.0%        100    Covered
        covered/total bins:                              6          6
        missing/total bins:                              0          6
        % Hit:                                      100.0%        100
        bin B_data_0                                  1457          1    Covered
        bin B_data_max                                1373          1    Covered
        bin B_data_min                                1368          1    Covered
        bin B_data_walkingones[-4]                     364          1    Covered
        bin B_data_walkingones[1]                      368          1    Covered
        bin B_data_walkingones[2]                      320          1    Covered
        default bin B_data_default                    3797               Occurred
    Coverpoint ALU_cp                               100.0%        100    Covered
        covered/total bins:                              7          7
        missing/total bins:                              0          7
        % Hit:                                      100.0%        100
        illegal_bin Bins_invalid                      2482               Occurred
        bin Bins_shift[SHIFT]                         1291          1    Covered
        bin Bins_shift[ROTATE]                        1352          1    Covered
        bin Bins_arith[ADD]                           1444          1    Covered
        bin Bins_arith[MULT]                          1197          1    Covered
        bin Bins_bitwise[OR]                          1326          1    Covered
        bin Bins_bitwise[XOR]                         1367          1    Covered
        bin Bins_trans                                   1          1    Covered
    Coverpoint cin_cp                               100.0%        100    Covered
```

```
    Coverpoint cin_cp                                   100.0%        100    Covered
        covered/total bins:                                  1          1
        missing/total bins:                                  0          1
        % Hit:                                          100.0%        100
        bin cin_data                                     10459          1    Covered
    Coverpoint direction_cp                             100.0%        100    Covered
        covered/total bins:                                  1          1
        missing/total bins:                                  0          1
        % Hit:                                          100.0%        100
        bin direction_data                               10459          1    Covered
    Coverpoint serial_in_cp                             100.0%        100    Covered
        covered/total bins:                                  1          1
        missing/total bins:                                  0          1
        % Hit:                                          100.0%        100
        bin serial_in_data                               10459          1    Covered
    Coverpoint red_op_A_cp                              100.0%        100    Covered
        covered/total bins:                                  2          2
        missing/total bins:                                  0          2
        % Hit:                                          100.0%        100
        bin red_op_A_LOW_data                             5420          1    Covered
        bin red_op_A_HIGH_data                            5039          1    Covered
    Coverpoint red_op_B_cp                              100.0%        100    Covered
        covered/total bins:                                  2          2
        missing/total bins:                                  0          2
        % Hit:                                          100.0%        100
        bin red_op_B_LOW_data                             5468          1    Covered
        bin red_op_B_HIGH_data                            4991          1    Covered
    Cross add_mult_cp1                                  100.0%        100    Covered
        covered/total bins:                                  3          3
        missing/total bins:                                  0          3
        % Hit:                                          100.0%        100
        bin zero_A_add                                      47          1    Covered
        bin max_nega_add                                    45          1    Covered
        bin max_pos_add                                     56          1    Covered
    Cross opcode_cp2                                    100.0%        100    Covered
        covered/total bins:                                  3          3
        missing/total bins:                                  0          3
        % Hit:                                          100.0%        100
        bin cin_add                                       1444          1    Covered
        bin serialin_shift                                1291          1    Covered
        bin direction_shift_rota                          2643          1    Covered
    Cross or_xor_cp3                                    100.0%        100    Covered
        covered/total bins:                                  2          2
        missing/total bins:                                  0          2
        % Hit:                                          100.0%        100
        bin or_xor_data_A                                   64          1    Covered
        bin or_xor_data_B                                   60          1    Covered
    Cross INVALID_cp4                                   100.0%        100    Covered
        covered/total bins:                                  2          2
        missing/total bins:                                  0          2
        % Hit:                                          100.0%        100
        bin Bins_shift_data                               1842          1    Covered
        bin Bins_arith_data                               1978          1    Covered

TOTAL COVERGROUP COVERAGE: 100.0%  COVERGROUP TYPES: 1
```

# Question 2

```
Q2

1)
property prop1;
  @(posedge clk)
  disable iff (reset)
  a |-> ##2 b
endproperty

    dollar1_ass1:  assert property (prop1) ;
    dollar1_cover1: cover property (prop1) ;

2)
property prop2;
  @(posedge clk)
  disable iff (reset)
  a && b |-> ##[1:3] c
endproperty

    dollar1_ass2:  assert property (prop2) ;
    dollar1_cover2: cover property (prop2) ;

3)
property prop3;
  @(posedge clk)
  disable iff (reset)
  s11b |-> ##2 !b
endproperty

    dollar1_ass3:  assert property (prop3) ;
    dollar1_cover3: cover property (prop3) ;

4.1)

property decoder_output_one_bit_high;
 @(posedge clk)
 disable iff (reset)
(countones(Y) == 1);
endproperty

dollar1_ass4:  assert property (decoder_output_one_bit_high) ;
dollar1_cover4: cover property (decoder_output_one_bit_high) ;


4.2)
property priority_encoder_valid_low;
@(posedge clk) disable iff (reset) (D == 4'b0000) |=> (valid == 0);
endproperty


dollar1_ass5:  assert property (priority_encoder_valid_low) ;
dollar1_cover5: cover property (priority_encoder_valid_low) ;
```

# Question 3_1

- priority encoder

1. Code Design

```
module priority_enc (
input  clk,
input  rst,
input  [3:0] D,
output reg [1:0] Y,
output reg valid
);

always @(posedge clk) begin
  if (rst)
    begin
     Y <= 2'b0;
     valid <= 1'b0 ;
    end
  else
    begin
    casex (D)
        4'b1000: Y <= 0;
        4'bX100: Y <= 1;
        4'bXX10: Y <= 2;
        4'bXXX1: Y <= 3;
        default: Y <= 2'b0;
    endcase
    valid <= (~|D)? 1'b0: 1'b1;
    end
end
endmodule
```

## 2. Verification plan

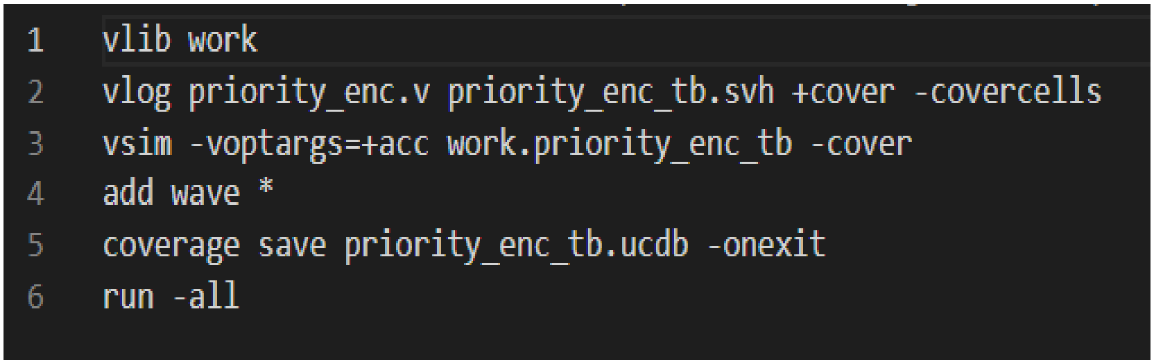
| LABEL | Description | Stimulus Generation | Function check |
|---|---|---|---|
| case_1 | when the reset is asserted . The output C value must be low && valid must be equal 0 | directed at the start of the simulation | A checker in the testbench to make sure the output is correct |
| case_2 | when D = 4'b1000 . The output C value must be low && valid must be equal 1 | directed during the simulation | A checker in the testbench to make sure the output is correct |
| case_3 | when D = 4'b0100 . The output C value must be equal 1 && valid must be equal 1 | directed during the simulation | A checker in the testbench to make sure the output is correct |
| case_4 | when D = 4'b1010 . The output C value must be equal 1 && valid must be equal 1 | directed during the simulation | A checker in the testbench to make sure the output is correct |
| case_5 | when D = 4'b0101 . The output C value must be equal 1 && valid must be equal 1 | directed during the simulation | A checker in the testbench to make sure the output is correct |
| case_6 | when D = 4'b1000 . The output C value must be low && valid must be equal 1 | directed during the simulation | A checker in the testbench to make sure the output is correct |
| case_7 | when D = 4'b0000 . The output C value must be low "default case" && valid must be equal 0 | directed during the simulation | A checker in the testbench to make sure the output is correct |

## 3. priority encoder Testbench

```
module priority_enc_tb ();
    logic        clk   ;
    logic        rst   ;
    logic  [3:0] D     ;
    logic  [1:0] Y     ;
    logic        valid ;
integer correct_counter = 0 ;
integer error_counter = 0 ;
priority_enc dut (
        .clk(clk),
        .rst(rst),
        .D(D),
        .Y(Y),
        .valid(valid)
    );
    always #20 clk = ~clk ;
    property prop1 ;
        @(posedge clk )
        disable iff (rst)
        ( D === 4'b1000 ) |=> (Y === 0) && valid;
    endproperty
    dollar1_ass:  assert property (prop1) ;
    dollar1_cover: cover property (prop1) ;
    property prop2 ;
        @(posedge clk )
        disable iff (rst)
        (( D === 4'b0100 ) || ( D === 4'b1100 ) )|=> (Y === 1)&& valid;
    endproperty

    dollar2_ass:  assert property (prop2) ;
    dollar2_cover: cover property (prop2) ;
```

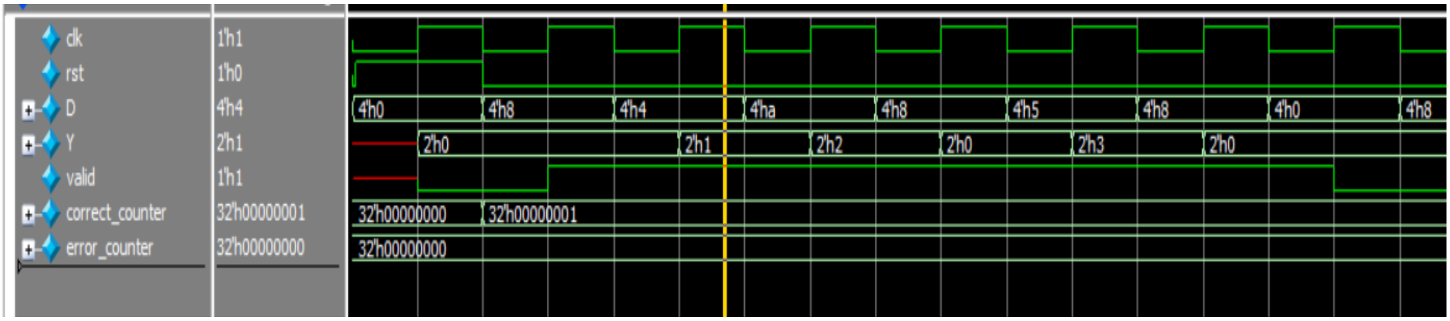
```
    initial
        begin
            clk = 0 ;
            D   = 0 ;
            rest ();

            D   = 4'b1000 ;
            @(negedge clk )

            D   = 4'b0100 ;
            @(negedge clk )

            D   = 4'b1010 ;
            @(negedge clk )

            D   = 4'b1000 ;
            @(negedge clk )

            D   = 4'b0101 ;
            @(negedge clk )

            D   = 4'b1000 ;
            @(negedge clk )


            D   = 4'b0000 ;
            @(negedge clk )

            D   = 4'b1000 ;
            @(negedge clk )

            D   = 4'b0100 ;
            @(negedge clk )

            D   = 4'b1100 ;
            @(negedge clk )

            D   = 4'b0010 ;
            @(negedge clk )

            D   = 4'b1010 ;
            @(negedge clk )

            D   = 4'b0110 ;
            @(negedge clk )

            D   = 4'b1110 ;
            @(negedge clk )



            D   = 4'b0001 ;
            @(negedge clk )

            D   = 4'b1001 ;
            @(negedge clk )

            D   = 4'b0101 ;
            @(negedge clk )

            D   = 4'b1101 ;
            @(negedge clk )

            D   = 4'b0011 ;
            @(negedge clk )

            D   = 4'b1011 ;
            @(negedge clk )

            D   = 4'b0111 ;
            @(negedge clk )

            D   = 4'b1111 ;
            @(negedge clk )

            $display ("error_counter = %0d " ,error_counter );
            $display ("correct_counter = %0d " ,correct_counter );
            $stop ;
        end
        task rest ();
            rst = 0 ;
            #1
            rst = 1 ;
            check_result(2'b0 , 0 );
            rst = 0 ;

        endtask

endmodule
```

4. Do File

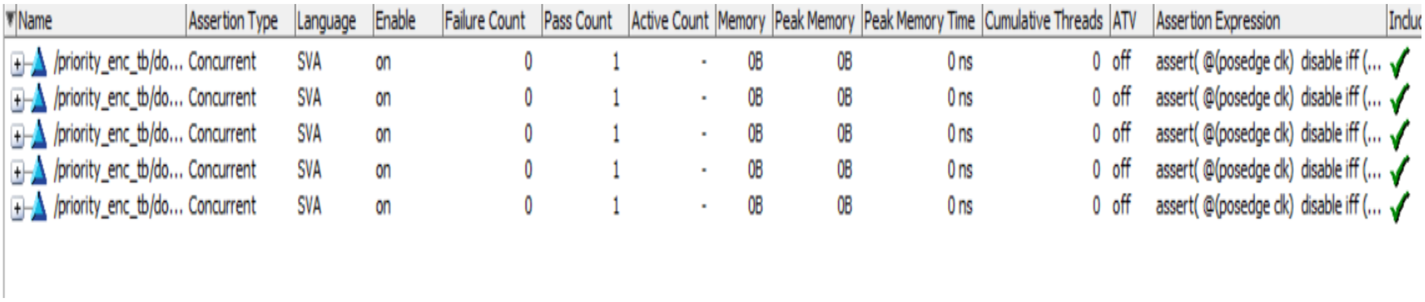


```
vlib work
vlog priority_enc.v priority_enc_tb.svh +cover -covercells
vsim -voptargs=+acc work.priority_enc_tb -cover
add wave *
coverage save priority_enc_tb.ucdb -onexit
run -all
```

5. Waveform

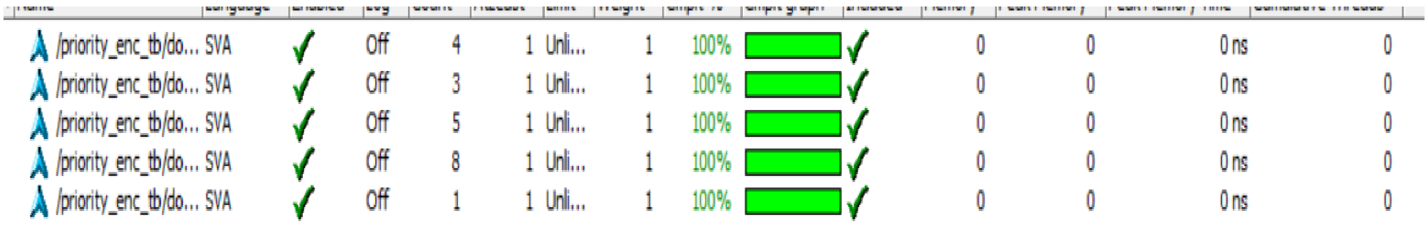


6. assertion



| Name | Assertion Type | Language | Enable | Failure Count | Pass Count | Active Count | Memory | Peak Memory | Peak Memory Time | Cumulative Threads | ATV | Assertion Expression | Induc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| /priority_enc_tb/do... | Concurrent | SVA | on | 0 | 1 | - | 0B | 0B | 0 ns | 0 | off | assert( @(posedge clk) disable iff (... | ✓ |
| /priority_enc_tb/do... | Concurrent | SVA | on | 0 | 1 | - | 0B | 0B | 0 ns | 0 | off | assert( @(posedge clk) disable iff (... | ✓ |
| /priority_enc_tb/do... | Concurrent | SVA | on | 0 | 1 | - | 0B | 0B | 0 ns | 0 | off | assert( @(posedge clk) disable iff (... | ✓ |
| /priority_enc_tb/do... | Concurrent | SVA | on | 0 | 1 | - | 0B | 0B | 0 ns | 0 | off | assert( @(posedge clk) disable iff (... | ✓ |
| /priority_enc_tb/do... | Concurrent | SVA | on | 0 | 1 | - | 0B | 0B | 0 ns | 0 | off | assert( @(posedge clk) disable iff (... | ✓ |

7. Assertion coverage



| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| /priority_enc_tb/do... | SVA | ✓ | Off | 4 | 1 | Unli... | 1 | 100% | 🟩 | ✓ | 0 | 0 | 0 ns | 0 |
| /priority_enc_tb/do... | SVA | ✓ | Off | 3 | 1 | Unli... | 1 | 100% | 🟩 | ✓ | 0 | 0 | 0 ns | 0 |
| /priority_enc_tb/do... | SVA | ✓ | Off | 5 | 1 | Unli... | 1 | 100% | 🟩 | ✓ | 0 | 0 | 0 ns | 0 |
| /priority_enc_tb/do... | SVA | ✓ | Off | 8 | 1 | Unli... | 1 | 100% | 🟩 | ✓ | 0 | 0 | 0 ns | 0 |
| /priority_enc_tb/do... | SVA | ✓ | Off | 1 | 1 | Unli... | 1 | 100% | 🟩 | ✓ | 0 | 0 | 0 ns | 0 |

## 8. Code Coverage

```
================================================================================
Statement Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Stmts                            9         9         0     100.0

=================================Statement Details================================

Statement Coverage for file priority_enc.v --

    1                                                   module priority_enc (
    2                                                   input  clk,
    3                                                   input  rst,
    4                                                   input  [3:0] D,
    5                                                   output reg [1:0] Y,
    6                                                   output reg valid
    7                                                   );
    8
    9               1                          23       always @(posedge clk) begin
    10                                                    if (rst)
    11                                                     begin
    12              1                           1            Y <= 2'b0;
    13              1                           1             valid <= 1'b0 ;
    14                                                     end
    15                                                    else
    16                                                     begin
    17                                                     casex (D)
    18              1                           4                 4'b1000: Y <= 0;
    19              1                           3                 4'bX100: Y <= 1;
    20              1                           5                 4'bXX10: Y <= 2;
    21              1                           9                 4'bXXX1: Y <= 3;
    22              1                           1                 default: Y <= 2'b0;
    23                                                     endcase
    24              1                          22          valid <= (~|D)? 1'b0: 1'b1;
    25                                                     end
    26                                                   end
```

```
Branch Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Branches                         7         7         0     100.0

================================Branch Details================================

Branch Coverage for file priority_enc.v --

------------------------------------IF Branch------------------------------------
    10                                         23     Count coming in to IF
    10              1                           1       if (rst)
    15              1                          22       else
Branch totals: 2 hits of 2 branches = 100.0%

-----------------------------------CASE Branch-----------------------------------
    17                                         22     Count coming in to CASE
    18              1                           4                 4'b1000: Y <= 0;
    19              1                           3                 4'bX100: Y <= 1;
    20              1                           5                 4'bXX10: Y <= 2;
    21              1                           9                 4'bXXX1: Y <= 3;
    22              1                           1                 default: Y <= 2'b0;
Branch totals: 5 hits of 5 branches = 100.0%
```

```
        Transitions                    0          0          0     100.0
Toggle Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Toggle Bins                     18        18         0     100.0

================================Toggle Details================================

Toggle Coverage for File priority_enc.v --

       Line                                   Node      1H->0L      0L->1H  "Coverage"
--------------------------------------------------------------------------------------
          2                                    clk           1           1      100.00
          3                                    rst           1           1      100.00
          4                                   D[3]           1           1      100.00
          4                                   D[2]           1           1      100.00
          4                                   D[1]           1           1      100.00
          4                                   D[0]           1           1      100.00
          5                                   Y[1]           1           1      100.00
          5                                   Y[0]           1           1      100.00
          6                                  valid           1           1      100.00
```

# Question 3_2

## ➢ Alu

1. Code Design

```
module ALU_4_bit (

    localparam          Add             = 3'b00; // A + B
    localparam          Sub             = 3'b01; // A - B
    localparam          Not_A           = 3'b10; // ~A
    localparam          ReductionOR_B   = 3'b11; // |B

    // Do the operation
    always @* begin
        case (Opcode)
            Add:            Alu_out = A + B;
            Sub:            Alu_out = A - B;
            Not_A:          Alu_out = ~A;
            ReductionOR_B:  Alu_out = |B;
            default:  Alu_out = 5'b0;
        endcase
    end // always @ *

    // Register output C
    always @(posedge clk or posedge reset) begin
        if (reset)
          C <= 5'b0;
        else
          C<= Alu_out;
    end

endmodule
```

## 2. Verification plan

| LABEL | Description | Stimulus Generation | Function check |
|---|---|---|---|
| case_1 | when the reset is asserted . The output C value must be low | directed at the start of the simulation | A checker in the testbench to make sure the output is correct |
| case_2 | verifing maximum negative value on A and maximum negative value on B && Opcode equal 0 | directed during the simulation | A checker in the testbench to make sure the output is correct |
| case_3 | verifing maximum negative value on A and ZERO value on B && Opcode equal 1 | directed during the simulation | A checker in the testbench to make sure the output is correct |
| case_4 | verifing maximum negative value on A and maximum positive value on B && Opcode equal 2 | directed during the simulation | A checker in the testbench to make sure the output is correct |
| case_5 | verifing ZERO value on A and maximum negative value on B && Opcode equal 3 | directed during the simulation | A checker in the testbench to make sure the output is correct |
| case_6 | verifing maximum positive value on A and ZERO value on B && Opcode equal 0 | directed during the simulation | A checker in the testbench to make sure the output is correct |
| case_7 | verifing ZERO value on A and ZERO value on B && Opcode equal 1 | directed during the simulation | A checker in the testbench to make sure the output is correct |
| case_8 | verifing maximum positive value on A and ZERO value on B && Opcode equal 2 | directed during the simulation | A checker in the testbench to make sure the output is correct |

## 3. Alu Testbench

```
module ALU_4_bit_tb ();
    localparam MAXPOS = 7  ;
    localparam MAXNEG = -8 ;
    logic clk ;
    logic reset ;
    logic   [1:0]  Opcode;
    logic signed [3:0] A ;
    logic signed [3:0] B ;
    logic signed [4:0] C ;
    localparam          Add            = 3'b00; // A + B
    localparam          Sub            = 3'b01; // A - B
    localparam          Not_A            = 3'b10; // ~A
    localparam          ReductionOR_B  = 3'b11; // |B

ALU_4_bit dut (
        .clk(clk),
        .reset(reset),
        .Opcode(Opcode),
        .A(A),
        .B(B),
        .C(C)
    );

    always #20 clk = ~clk ;

    property prop1_Add;
        @(posedge clk)
        disable iff (reset)
        (Opcode === Add)   |=>  (C === ($past(A)+$past(B)));
      endproperty

      assert_add: assert property (prop1_Add) ;
      cover_add: cover property (prop1_Add) ;
```

```
36      property prop1_sub;
37        @(posedge clk)
38        disable iff (reset)
39        (Opcode === Sub)   |=>  (C === ($past(A)-$past(B)));
40      endproperty
41
42      assert_sub: assert property (prop1_sub) ;
43      cover_sub: cover property (prop1_sub) ;
44
45      property prop1_not_A;
46        @(posedge clk)
47        disable iff (reset)
48        (Opcode === Not_A)   |=>  (C === (~$past(A)));
49      endproperty
50
51      assert_not_A: assert property (prop1_not_A) ;
52      cover_not_A: cover property (prop1_not_A) ;
53
54
55      property prop1_Reduc;
56        @(posedge clk)
57        disable iff (reset)
58        (Opcode === ReductionOR_B)   |=>  (C === (|$past(B)));
59      endproperty
60
61      assert_Reduc: assert property (prop1_Reduc) ;
62      cover_Reduc: cover property (prop1_Reduc) ;
```

```
64      property prop1_reset;
65        @(posedge clk)
66        reset   |=>   (C === 0);
67      endproperty
68
69      assert_reset: assert property (prop1_reset) ;
70      cover_reset: cover property (prop1_reset) ;
71
72    initial
73      begin
74        clk = 0 ;
75        A = 0 ;
76        B = 0 ;
77        Opcode = 0 ;
78        rest();
79
80        reset = 0 ;
81        A = MAXNEG ;
82        B = MAXNEG ;
83
84        @(negedge clk );
85
86        A = MAXNEG ;
87        B = MAXPOS ;
88
89        @(negedge clk );
90
91        B = MAXNEG ;
92        A = MAXPOS ;
93
94        @(negedge clk );
95
```

```
A = MAXPOS ;
B = MAXPOS ;

@(negedge clk );

A = MAXNEG ;
B = 0 ;

@(negedge clk );

A = 0 ;
B = 0 ;

@(negedge clk );

A = MAXPOS ;
B = 0 ;
@(negedge clk );

reset = 0 ;

@(negedge clk );
Opcode = 1 ;
A = MAXNEG ;
B = MAXNEG ;

@(negedge clk );
```

```

A = MAXNEG ;
B = MAXPOS ;

@(negedge clk );

B = MAXNEG ;
A = MAXPOS ;

@(negedge clk );

A = MAXPOS ;
B = MAXPOS ;

@(negedge clk );

A = MAXNEG ;
B = 0 ;

@(negedge clk );

A = 0 ;
B = 0 ;

@(negedge clk );

A = MAXPOS ;
B = 0 ;

@(negedge clk );
```

```
@(negedge clk );
Opcode = 2 ;
A = MAXNEG ;
B = MAXNEG ;

@(negedge clk );

A = MAXNEG ;
B = MAXPOS ;

@(negedge clk );

B = MAXNEG ;
A = MAXPOS ;

@(negedge clk );

A = MAXPOS ;
B = MAXPOS ;

@(negedge clk );

A = MAXNEG ;
B = 0 ;

@(negedge clk );

A = 0 ;
B = 0 ;

@(negedge clk );
```

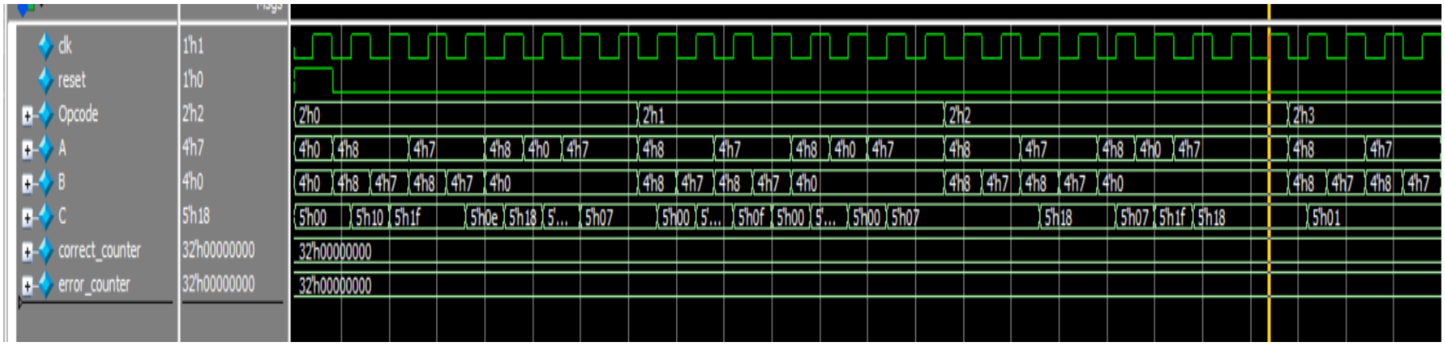
```
A = MAXPOS ;
B = 0 ;

@(negedge clk );

reset = 0 ;

@(negedge clk );

reset = 0 ;

@(negedge clk );
Opcode = 3 ;
A = MAXNEG ;
B = MAXNEG ;

@(negedge clk );

A = MAXNEG ;
B = MAXPOS ;

@(negedge clk );

B = MAXNEG ;
A = MAXPOS ;

@(negedge clk );

A = MAXPOS ;
B = MAXPOS ;
```

```
            A = MAXNEG ;
            B = 0 ;
            @(negedge clk );
            A = 0 ;
            B = 0 ;
            @(negedge clk );

            A = MAXPOS ;
            B = 0 ;

            @(negedge clk );
            Opcode = 0 ;
            @(negedge clk );
            $stop();
        end
        task rest ();
            reset = 0 ;
            #1
            reset = 1 ;
            #40;
            reset = 0 ;

        endtask


    endmodule
```

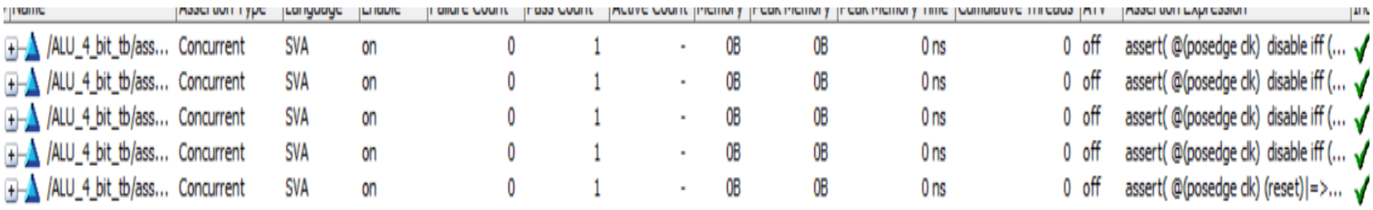
1. Do File

```
vlib work
vlog ALU.v alu_tb.sv +cover -covercells
vsim -voptargs=+acc work.ALU_4_bit_tb -cover
add wave *
coverage save alu_tb.ucdb -onexit
run -all
```

## 3. Waveform

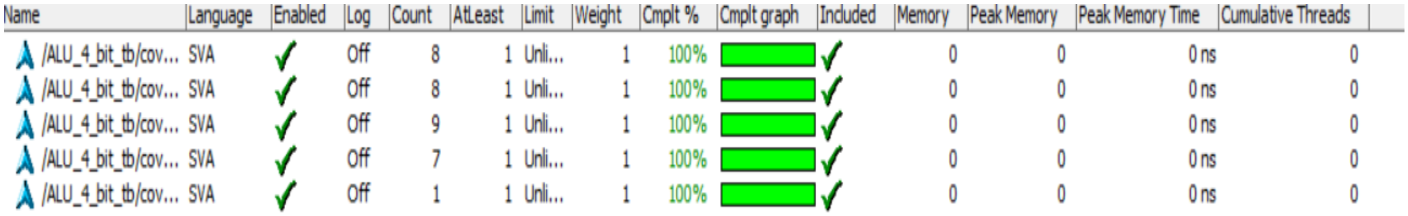


## 4. Assertion



| Name | Assertion Type | Language | Enable | Failure Count | Pass Count | Active Count | Memory | Peak Memory | Peak Memory Time | Cumulative Threads | ATV | Assertion Expression | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| /ALU_4_bit_tb/ass... | Concurrent | SVA | on | 0 | 1 | - | 0B | 0B | 0 ns | 0 | off | assert( @(posedge clk) disable iff (... | ✓ |
| /ALU_4_bit_tb/ass... | Concurrent | SVA | on | 0 | 1 | - | 0B | 0B | 0 ns | 0 | off | assert( @(posedge clk) disable iff (... | ✓ |
| /ALU_4_bit_tb/ass... | Concurrent | SVA | on | 0 | 1 | - | 0B | 0B | 0 ns | 0 | off | assert( @(posedge clk) disable iff (... | ✓ |
| /ALU_4_bit_tb/ass... | Concurrent | SVA | on | 0 | 1 | - | 0B | 0B | 0 ns | 0 | off | assert( @(posedge clk) disable iff (... | ✓ |
| /ALU_4_bit_tb/ass... | Concurrent | SVA | on | 0 | 1 | - | 0B | 0B | 0 ns | 0 | off | assert( @(posedge clk) (reset)\|=>... | ✓ |

## 5. Assertion coverage



| Name | Language | Enabled | Log | Count | AtLeast | Limit | Weight | Cmplt % | Cmplt graph | Included | Memory | Peak Memory | Peak Memory Time | Cumulative Threads |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| /ALU_4_bit_tb/cov... | SVA | ✓ | Off | 8 | 1 | Unli... | 1 | 100% | 🟩 | ✓ | 0 | 0 | 0 ns | 0 |
| /ALU_4_bit_tb/cov... | SVA | ✓ | Off | 8 | 1 | Unli... | 1 | 100% | 🟩 | ✓ | 0 | 0 | 0 ns | 0 |
| /ALU_4_bit_tb/cov... | SVA | ✓ | Off | 9 | 1 | Unli... | 1 | 100% | 🟩 | ✓ | 0 | 0 | 0 ns | 0 |
| /ALU_4_bit_tb/cov... | SVA | ✓ | Off | 7 | 1 | Unli... | 1 | 100% | 🟩 | ✓ | 0 | 0 | 0 ns | 0 |
| /ALU_4_bit_tb/cov... | SVA | ✓ | Off | 1 | 1 | Unli... | 1 | 100% | 🟩 | ✓ | 0 | 0 | 0 ns | 0 |

# 6. Code Coverage

```
Statement Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Stmts                            8         8         0     100.0

==============================Statement Details==============================

Statement Coverage for file ALU.v --

    1                                           module ALU_4_bit (
    2                                               input  clk,
    3                                               input  reset,
    4                                               input  [1:0] Opcode,   // The opcode
    5                                               input  signed [3:0] A, // Input data A in 2's complement
    6                                               input  signed [3:0] B, // Input data B in 2's complement
    7
    8                                               output reg signed [4:0] C // ALU output in 2's complement
    9
    10                                                      );
    11
    12                                             reg signed [4:0]           Alu_out; // ALU output in 2's complement
    13
    14                                             localparam                 Add            = 3'b00; // A + B
    15                                             localparam                 Sub            = 3'b01; // A - B
    16                                             localparam                 Not_A                = 3'b10; // ~A
    17                                             localparam                 ReductionOR_B  = 3'b11; // |B
    18
    19                                             // Do the operation
    20             1                       30      always @* begin
    21                                               case (Opcode)
    22             1                        9            Add:           Alu_out = A + B;
    23             1                        7            Sub:           Alu_out = A - B;
    24             1                        7            Not_A:         Alu_out = ~A;
    25             1                        7            ReductionOR_B: Alu_out = |B;
    26             1                        E          default:  Alu_out = 5'b0;
```

```
Branch Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Branches                         6         6         0     100.0

==============================Branch Details==============================

Branch Coverage for file ALU.v --

------------------------------------CASE Branch------------------------------------
    21                                      30     Count coming in to CASE
    22             1                         9            Add:           Alu_out = A + B;
    23             1                         7            Sub:           Alu_out = A - B;
    24             1                         7            Not_A:         Alu_out = ~A;
    25             1                         7            ReductionOR_B: Alu_out = |B;
    26             1                         E          default:  Alu_out = 5'b0;
Branch totals: 4 hits of 4 branches = 100.0%

------------------------------------IF Branch------------------------------------
    32                                      29     Count coming in to IF
    32             1                         2          if (reset)
    34             1                        27          else
Branch totals: 2 hits of 2 branches = 100.0%
```

```
Toggle Coverage:
    Enabled Coverage              Active      Hits    Misses % Covered
    ----------------              ------      ----    ------ ---------
    Toggle Bins                       44        44         0     100.0

================================Toggle Details================================

Toggle Coverage for File ALU.v --

        Line                                    Node      1H->0L      0L->1H  "Coverage"
--------------------------------------------------------------------------------------
           2                                     clk           1           1      100.00
           3                                   reset           1           1      100.00
           4                               Opcode[1]           1           1      100.00
           4                               Opcode[0]           1           1      100.00
           5                                    A[3]           1           1      100.00
           5                                    A[2]           1           1      100.00
           5                                    A[1]           1           1      100.00
           5                                    A[0]           1           1      100.00
           6                                    B[3]           1           1      100.00
           6                                    B[2]           1           1      100.00
           6                                    B[1]           1           1      100.00
           6                                    B[0]           1           1      100.00
           8                                    C[4]           1           1      100.00
           8                                    C[3]           1           1      100.00
           8                                    C[2]           1           1      100.00
           8                                    C[1]           1           1      100.00
           8                                    C[0]           1           1      100.00
          12                              Alu_out[4]           1           1      100.00
          12                              Alu_out[3]           1           1      100.00
          12                              Alu_out[2]           1           1      100.00
          12                              Alu_out[1]           1           1      100.00
          12                              Alu_out[0]           1           1      100.00
```

```
DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------------------
Name                                    Design Design   Lang File(Line)      Count Status
                                        Unit   UnitType
--------------------------------------------------------------------------------------------
/ALU_4_bit_tb/cover_add                 ALU_4_bit_tb Verilog  SVA  alu_tb.sv(41)       8 Covered
/ALU_4_bit_tb/cover_sub                 ALU_4_bit_tb Verilog  SVA  alu_tb.sv(51)       8 Covered
/ALU_4_bit_tb/cover_not_A               ALU_4_bit_tb Verilog  SVA  alu_tb.sv(60)       9 Covered
/ALU_4_bit_tb/cover_Reduc               ALU_4_bit_tb Verilog  SVA  alu_tb.sv(70)       7 Covered
/ALU_4_bit_tb/cover_reset               ALU_4_bit_tb Verilog  SVA  alu_tb.sv(78)       1 Covered

TOTAL DIRECTIVE COVERAGE: 100.0%  COVERS: 5
```

# Question 4

## ➢ Counter

1. Code Design

```
////////////////////////////////////////////////////////////////////////////////
module counter(counter_inter.DUT  counter_if);

always @(posedge counter_if.clk , negedge counter_if.rst_n ) begin
    if (!counter_if.rst_n)
        counter_if.count_out <= 0;
    else if (!counter_if.load_n)
        counter_if.count_out <= counter_if.data_load;
    else if (counter_if.ce)
        if (counter_if.up_down)
            counter_if.count_out <= counter_if.count_out + 1;
        else
            counter_if.count_out <= counter_if.count_out - 1;
end

assign counter_if.max_count = (counter_if.count_out == {counter_if.WIDTH{1'b1}})? 1:0;
assign counter_if.zero = (counter_if.count_out == 0)? 1:0;

endmodule
```

2. Interface

```
interface counter_inter(clk) ;

    parameter WIDTH = 4;
    input clk;
    logic rst_n;
    logic load_n;
    logic up_down;
    logic ce;
    logic [WIDTH-1:0] data_load;
    logic [WIDTH-1:0] count_out;
    logic max_count;
    logic zero;

    modport TEST (output rst_n, load_n, up_down  , ce , data_load , input count_out , max_count , zero , clk ) ;

    modport DUT (input rst_n, load_n, up_down , clk , ce , data_load , output count_out , max_count , zero ) ;

    modport MONITOR (input rst_n, load_n, up_down , clk , ce , data_load , count_out , max_count , zero  ) ;

endinterface
```

3. Top module

```
module TOP_counter ();

    bit clk ;
    initial begin
        clk = 0;
        forever
          #1 clk = ~clk;
      end

      counter_inter  counter_if (clk) ;
      counter   DUT (counter_if);
      bind counter  sva_t  sva (counter_if);
      counter_tb   test (counter_if);


endmodule
```

4. Testbench

```
import pack_count::*;
module counter_tb(counter_inter.TEST  counter_if);
  integer i ;
  logic [counter_if.WIDTH:0] old_count_out ;
    cla_count trans1 = new()  ;
    initial begin
  forever
  begin
    #20 ;
  trans1.clk =  counter_if.clk ;
  end
  end
  initial begin
    rest ();
    @(posedge counter_if.clk );
    for(i=0; i<1000 ;i=i+1) begin
      trans1.randomize();
      //@(negedge counter_if.clk );
      //trans1.print() ;
      counter_if.data_load = trans1.data_load ;
      counter_if.ce = trans1.ce ;
      counter_if.up_down = trans1.up_down ;
      counter_if.load_n = trans1.load_n ;
      counter_if.rst_n = trans1.rst_n ;
      @(negedge counter_if.clk );
      trans1.count_out = counter_if.count_out ;
      end
    $stop ;
  end
  task rest ();
    counter_if.rst_n = 1 ;
    #20
    counter_if.rst_n = 0 ;
endtask
endmodule
```

5. Package

```
package pack_count;
    class cla_count ;
      parameter WIDTH = 4;
      rand  logic [WIDTH-1:0] data_load ;
      rand  logic             ce ;
      rand  logic             up_down ;
      rand  logic             load_n ;
      rand  logic             rst_n ;
      bit                     clk ;
      logic         [WIDTH-1:0] count_out ;
      parameter max_value = {{WIDTH{1'b1}}} ;
      parameter zero = 0 ;
      constraint enable {ce dist {1:/70 , 0:/30  } ; }
      constraint load {load_n dist {1:/30 , 0:/70  } ; }
      constraint rst {rst_n dist {1:/99 , 0:/1  } ; }
      covergroup covgr @(posedge clk) ;
        load_n_cp : coverpoint load_n {
          bins cove_asserted = {0} ;
      }
      count_out_cp1 : coverpoint count_out iff (rst_n&&ce&&up_down) ;
      count_out_cp2 : coverpoint count_out iff(rst_n && ce && up_down)
      {
        bins max_zero = (max_value => zero) ;
      }
      count_out_cp3 : coverpoint count_out iff(rst_n && ce && !up_down) ;
      count_out_cp4 : coverpoint count_out iff(rst_n && ce && !up_down)
      {
        bins zero_max = ( zero => max_value ) ;
      }
    endgroup
          function new() ;
            covgr  = new();
          endfunction
        endclass
endpackage
```

6. SVA

```
module sva_t(counter_inter.DUT  counter_if);

    property prop1 ;
        @(posedge counter_if.clk )
        disable iff (!counter_if.rst_n)
        (!counter_if.load_n) |=>  (counter_if.count_out === $past(counter_if.data_load)) ;
    endproperty
    dollar1_ass:  assert property (prop1) ;
    dollar1_cover: cover property (prop1) ;

    property prop2 ;
        @(posedge counter_if.clk )
        disable iff (!counter_if.rst_n)
        (counter_if.load_n && !counter_if.ce) |=> counter_if.count_out === $past(counter_if.count_out) ;
    endproperty
    dollar2_ass: assert property (prop2) ;
    dollar2_cover: cover property (prop2) ;

    property prop3 ;
        @(posedge counter_if.clk )
        disable iff (!counter_if.rst_n)
        (counter_if.load_n && counter_if.ce && counter_if.up_down) |=> (counter_if.count_out === $past(counter_if.count_out) + 1'b1 ) ;
    endproperty

    dollar3_ass: assert property (prop3) ;
    dollar3_cover: cover property (prop3) ;

    property prop4 ;
        @(posedge counter_if.clk )
        disable iff (!counter_if.rst_n)
        (counter_if.load_n && counter_if.ce && !counter_if.up_down) |=> (counter_if.count_out === $past(counter_if.count_out) - 1'b1 ) ;
    endproperty
    dollar4_ass:  assert property (prop4) ;

    property prop5;
    @(posedge counter_if.clk) (!counter_if.rst_n) |=>  (counter_if.count_out === 0)  ;
    endproperty

    dollar5_ass: assert property (prop5) ;
    dollar5_cover: cover property (prop5) ;

    property prop6;
    @(posedge counter_if.clk) ( counter_if.count_out === {counter_if.WIDTH{1'b1}}  )|-> counter_if.max_count ;
    endproperty

    dollar6_ass: assert property (prop6) ;
    dollar6_cover: cover property (prop6) ;

    property prop7;
    @(posedge counter_if.clk) ( counter_if.count_out === 0 )|-> counter_if.zero ;
    endproperty

    dollar7_ass: assert property (prop7) ;
    dollar7_cover: cover property (prop7) ;

endmodule
```

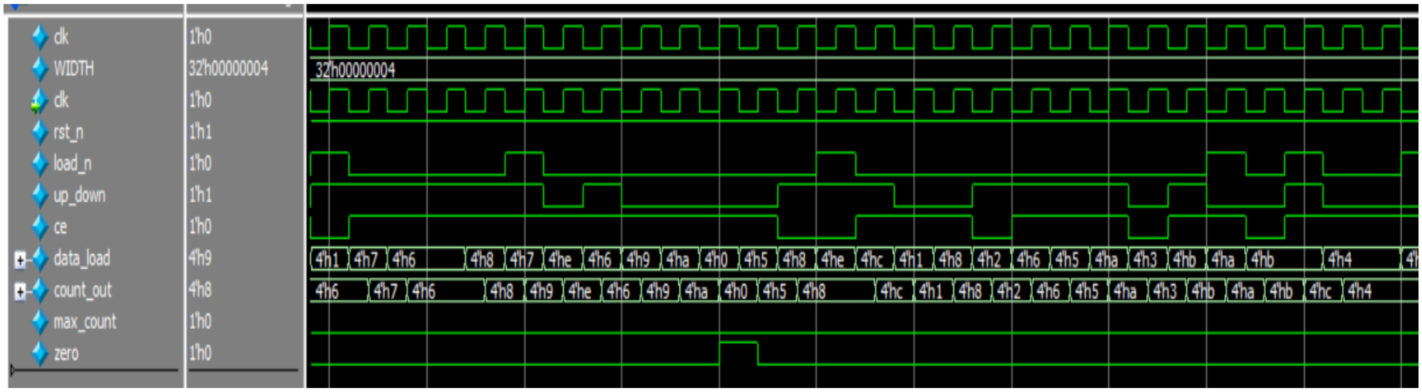
## 7. Do File

```
vlib work
vlog counter.sv counter_tb.sv sva_t.sv TOP_counter.sv counter_inter.sv pack_count.sv +cover -covercells
vsim -voptargs=+acc work.TOP_counter -cover
add wave *
coverage save TOP_counter.ucdb -onexit
add wave -position insertpoint  \
sim:/TOP_counter/DUT/counter_if/WIDTH \
sim:/TOP_counter/DUT/counter_if/clk \
sim:/TOP_counter/DUT/counter_if/rst_n \
sim:/TOP_counter/DUT/counter_if/load_n \
sim:/TOP_counter/DUT/counter_if/up_down \
sim:/TOP_counter/DUT/counter_if/ce \
sim:/TOP_counter/DUT/counter_if/data_load \
sim:/TOP_counter/DUT/counter_if/count_out \
sim:/TOP_counter/DUT/counter_if/max_count \
sim:/TOP_counter/DUT/counter_if/zero

run -all
```

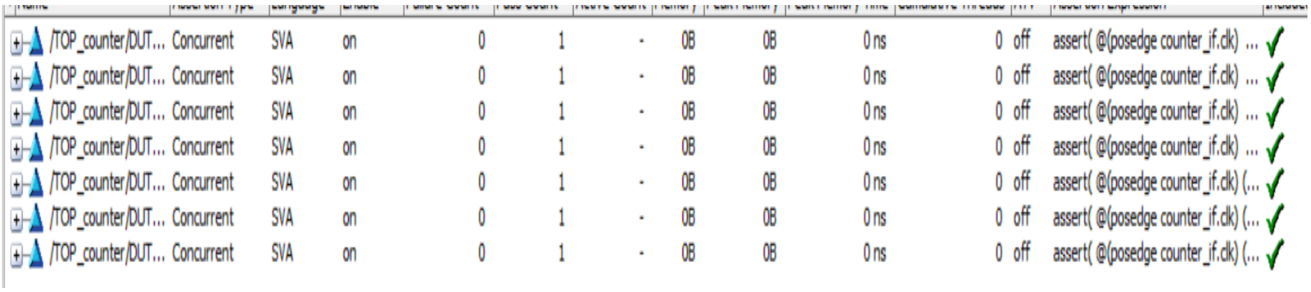
## 8. Verification plan

| Label | Design Requirement Description | Stimulus Generation | Functional Coverage | Functionality Check |
|---|---|---|---|---|
| COUNTER_1 | When the reset is asserted, the output counter value should be low | Directed at the start of the sim, then randomized with constraint that drive the reset to be off most of the simulation time. | - | A checker in the testbench to make sure the output is correct |
| COUNTER_2 | When the load is asserted, the output count_out should take the value of the load_data input | Randomization under constraints on the load signal to be off 70% of the time | Cover all values of load data | A checker in the testbench to make sure the output is correct |
| COUNTER_3 | When the load/reset is deasserted, the output count_out increment as long as the enable is active and up_down is high | Randomization for up_down, and randomization for enable to be 70% of the time | Cover all values of count_out, and transition bin from max to zero | A checker in the testbench to make sure the output is correct |
| COUNTER_4 | When the load/reset is deasserted, the output count_out increment as long as the enable is active and up_down is low | Randomization for up_down, and randomization for enable to be 70% of the time | Cover all values of count_out, and transition bin from zero to max | A checker in the testbench to make sure the output is correct |

## 9. Waveform

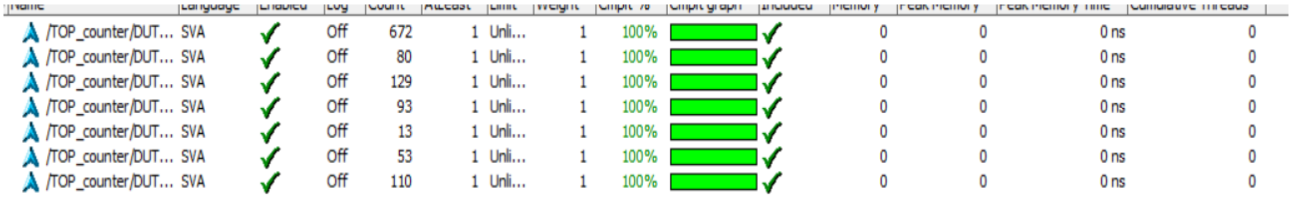


## 10. Assertion



| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| /TOP_counter/DUT... | Concurrent | SVA | on | 0 | 1 | - | 0B | 0B | 0 ns | 0 | off | assert( @(posedge counter_if.clk) ... | ✓ |
| /TOP_counter/DUT... | Concurrent | SVA | on | 0 | 1 | - | 0B | 0B | 0 ns | 0 | off | assert( @(posedge counter_if.clk) ... | ✓ |
| /TOP_counter/DUT... | Concurrent | SVA | on | 0 | 1 | - | 0B | 0B | 0 ns | 0 | off | assert( @(posedge counter_if.clk) ... | ✓ |
| /TOP_counter/DUT... | Concurrent | SVA | on | 0 | 1 | - | 0B | 0B | 0 ns | 0 | off | assert( @(posedge counter_if.clk) ... | ✓ |
| /TOP_counter/DUT... | Concurrent | SVA | on | 0 | 1 | - | 0B | 0B | 0 ns | 0 | off | assert( @(posedge counter_if.clk) (... | ✓ |
| /TOP_counter/DUT... | Concurrent | SVA | on | 0 | 1 | - | 0B | 0B | 0 ns | 0 | off | assert( @(posedge counter_if.clk) (... | ✓ |
| /TOP_counter/DUT... | Concurrent | SVA | on | 0 | 1 | - | 0B | 0B | 0 ns | 0 | off | assert( @(posedge counter_if.clk) (... | ✓ |

## 11. Assertion coverage



| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| /TOP_counter/DUT... | SVA | ✓ | Off | 672 | 1 | Unli... | 1 | 100% | 🟩 | ✓ | 0 | 0 | 0 ns | 0 |
| /TOP_counter/DUT... | SVA | ✓ | Off | 80 | 1 | Unli... | 1 | 100% | 🟩 | ✓ | 0 | 0 | 0 ns | 0 |
| /TOP_counter/DUT... | SVA | ✓ | Off | 129 | 1 | Unli... | 1 | 100% | 🟩 | ✓ | 0 | 0 | 0 ns | 0 |
| /TOP_counter/DUT... | SVA | ✓ | Off | 93 | 1 | Unli... | 1 | 100% | 🟩 | ✓ | 0 | 0 | 0 ns | 0 |
| /TOP_counter/DUT... | SVA | ✓ | Off | 13 | 1 | Unli... | 1 | 100% | 🟩 | ✓ | 0 | 0 | 0 ns | 0 |
| /TOP_counter/DUT... | SVA | ✓ | Off | 53 | 1 | Unli... | 1 | 100% | 🟩 | ✓ | 0 | 0 | 0 ns | 0 |
| /TOP_counter/DUT... | SVA | ✓ | Off | 110 | 1 | Unli... | 1 | 100% | 🟩 | ✓ | 0 | 0 | 0 ns | 0 |

# 12. Code Coverage

```
==============================================================================
=== File: counter.v
==============================================================================
Statement Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Stmts                            7         7         0     100.0

                                Statement Details

Statement Coverage for file counter.v --

    1                                           ////////////////////////////////////////////////////////////////////////////
    2                                           // Author: Kareem Waseem
    3                                           // Course: Digital Verification using SV & UVM
    4                                           //
    5                                           // Description: Counter Design
    6                                           //
    7                                           ////////////////////////////////////////////////////////////////////////////
    8                                           module counter (clk ,rst_n, load_n, up_down, ce, data_load, count_out, max_count, zero);
    9                                           parameter WIDTH = 4;
    10                                          input clk;
    11                                          input rst_n;
    12                                          input load_n;
    13                                          input up_down;
    14                                          input ce;
    15                                          input [WIDTH-1:0] data_load;
    16                                          output reg [WIDTH-1:0] count_out;
    17                                          output max_count;
    18                                          output zero;
    19
```

```
Branch Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Branches                        10        10         0     100.0

================================Branch Details================================

Branch Coverage for file counter.v --

------------------------------------IF Branch------------------------------------
    21                                      1000     Count coming in to IF
    21              1                          7         if (!rst_n)
    23              1                        707         else if (!load_n)
    25              1                        196         else if (ce)
                                              90     All False Count
Branch totals: 4 hits of 4 branches = 100.0%

------------------------------------IF Branch------------------------------------
    26                                       196     Count coming in to IF
    26              1                        116             if (up_down)
    28              1                         80             else
Branch totals: 2 hits of 2 branches = 100.0%

------------------------------------IF Branch------------------------------------
    32                                       906     Count coming in to IF
    32              1                          5     assign max_count = (count_out == {WIDTH{1'b1}})? 1:0;
    32              2                        901     assign max_count = (count_out == {WIDTH{1'b1}})? 1:0;
Branch totals: 2 hits of 2 branches = 100.0%

------------------------------------IF Branch------------------------------------
    33                                       906     Count coming in to IF
    33              1                         11     assign zero = (count_out == 0)? 1:0;
    33              2                        895     assign zero = (count_out == 0)? 1:0;
Branch totals: 2 hits of 2 branches = 100.0%
```

```
Toggle Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Toggle Bins                     46        46         0     100.0

================================Toggle Details================================

Toggle Coverage for File counter.v --

        Line                                    Node      1H->0L      0L->1H  "Coverage"
--------------------------------------------------------------------------------------------
          10                                     clk           1           1      100.00
          11                                   rst_n           1           1      100.00
          12                                  load_n           1           1      100.00
          13                                  up_down           1           1      100.00
          14                                       ce           1           1      100.00
          15                            data_load[7]           1           1      100.00
          15                            data_load[6]           1           1      100.00
          15                            data_load[5]           1           1      100.00
          15                            data_load[4]           1           1      100.00
          15                            data_load[3]           1           1      100.00
          15                            data_load[2]           1           1      100.00
          15                            data_load[1]           1           1      100.00
          15                            data_load[0]           1           1      100.00
          16                            count_out[7]           1           1      100.00
          16                            count_out[6]           1           1      100.00
          16                            count_out[5]           1           1      100.00
          16                            count_out[4]           1           1      100.00
          16                            count_out[3]           1           1      100.00
          16                            count_out[2]           1           1      100.00
          16                            count_out[1]           1           1      100.00
          16                            count_out[0]           1           1      100.00
          17                                max_count           1           1      100.00
          18                                     zero           1           1      100.00
```

## 13.function Coverage

```
        bin max_zero                                  3143          1    Covered
    Coverpoint count_out_cp3                         100.0%        100    Covered
        covered/total bins:                              16         16
        missing/total bins:                               0         16
        % Hit:                                       100.0%        100
        bin auto[0]                                   24240          1    Covered
        bin auto[1]                                   21882          1    Covered
        bin auto[2]                                   21698          1    Covered
        bin auto[3]                                   21219          1    Covered
        bin auto[4]                                   21348          1    Covered
        bin auto[5]                                   21608          1    Covered
        bin auto[6]                                   21505          1    Covered
        bin auto[7]                                   21246          1    Covered
        bin auto[8]                                   21160          1    Covered
        bin auto[9]                                   21547          1    Covered
        bin auto[10]                                  21228          1    Covered
        bin auto[11]                                  21532          1    Covered
        bin auto[12]                                  21783          1    Covered
        bin auto[13]                                  21157          1    Covered
        bin auto[14]                                  21535          1    Covered
        bin auto[15]                                  21905          1    Covered
    Coverpoint count_out_cp4                         100.0%        100    Covered
        covered/total bins:                               1          1
        missing/total bins:                               0          1
        % Hit:                                       100.0%        100
        bin zero_max                                   3320          1    Covered

  TOTAL COVERGROUP COVERAGE: 100.0%  COVERGROUP TYPES: 1
```

## 14.assertion coverage

```
DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------------------------
Name                                        Design Design   Lang File(Line)      Count Status
                                            Unit   UnitType
--------------------------------------------------------------------------------------------------
/TOP_counter/DUT/sva/dollar1_cover          sva_t  Verilog  SVA  sva_t.sv(9)       672 Covered
/TOP_counter/DUT/sva/dollar2_cover          sva_t  Verilog  SVA  sva_t.sv(19)       80 Covered
/TOP_counter/DUT/sva/dollar3_cover          sva_t  Verilog  SVA  sva_t.sv(29)      129 Covered
/TOP_counter/DUT/sva/dollar4_cover          sva_t  Verilog  SVA  sva_t.sv(38)       93 Covered
/TOP_counter/DUT/sva/dollar5_cover          sva_t  Verilog  SVA  sva_t.sv(45)       13 Covered
/TOP_counter/DUT/sva/dollar6_cover          sva_t  Verilog  SVA  sva_t.sv(52)       53 Covered
/TOP_counter/DUT/sva/dollar7_cover          sva_t  Verilog  SVA  sva_t.sv(59)      110 Covered

TOTAL DIRECTIVE COVERAGE: 100.0%  COVERS: 7
```