P1) counter

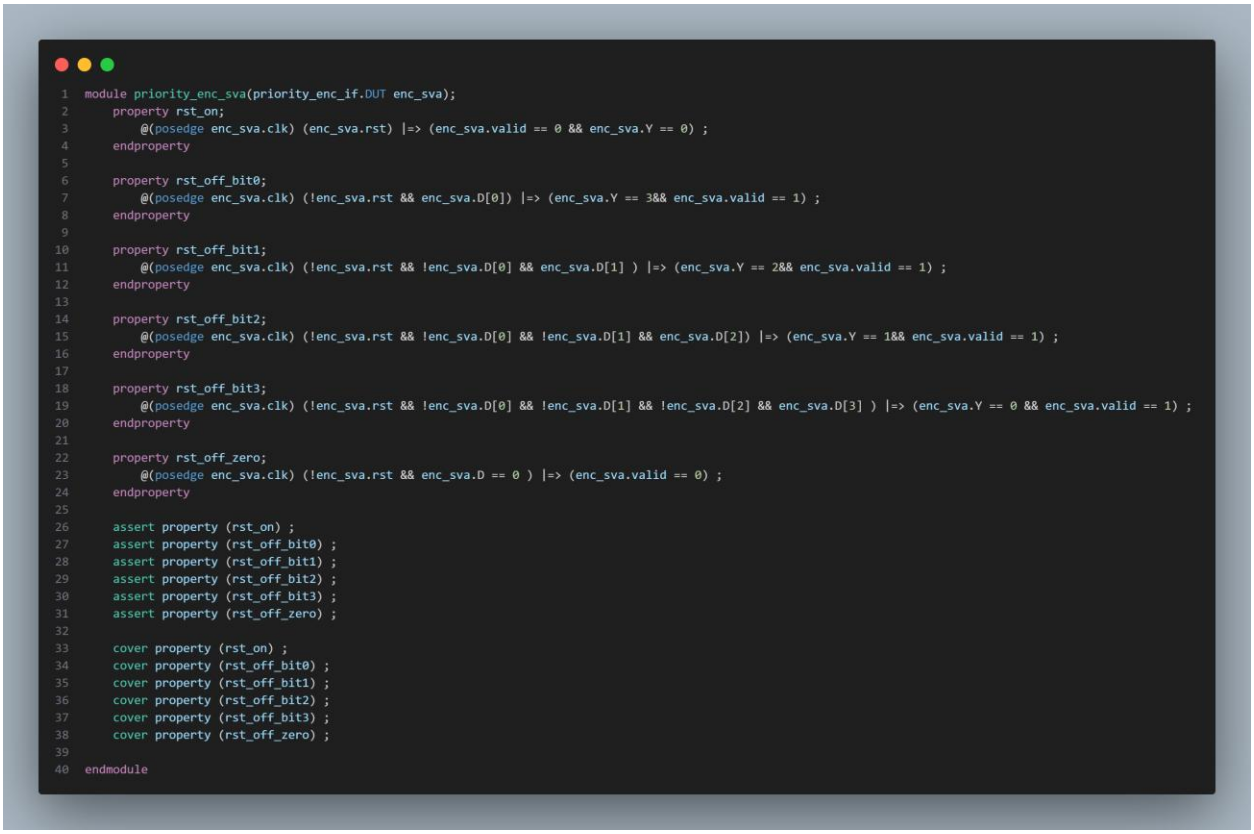
1. Testbench code

```
module priority_enc_tb(priority_enc_if.TEST enc_TEST);
    integer D_count ;

    initial begin
        enc_TEST.rst = 1 ;
        @(negedge enc_TEST.clk);
        // TEST_1
        enc_TEST.rst = 0 ;
        // TEST_2
        for (D_count = 0 ; D_count < 17 ; D_count = D_count + 1 ) begin
            @(posedge enc_TEST.clk);
            enc_TEST.D = D_count ;
            $display ("D = %b , Y = %b , enc_TEST.valid = %b" , enc_TEST.D , enc_TEST.Y , enc_TEST.valid) ;
        end
        $stop ;
    end
endmodule
```

2. Top module code

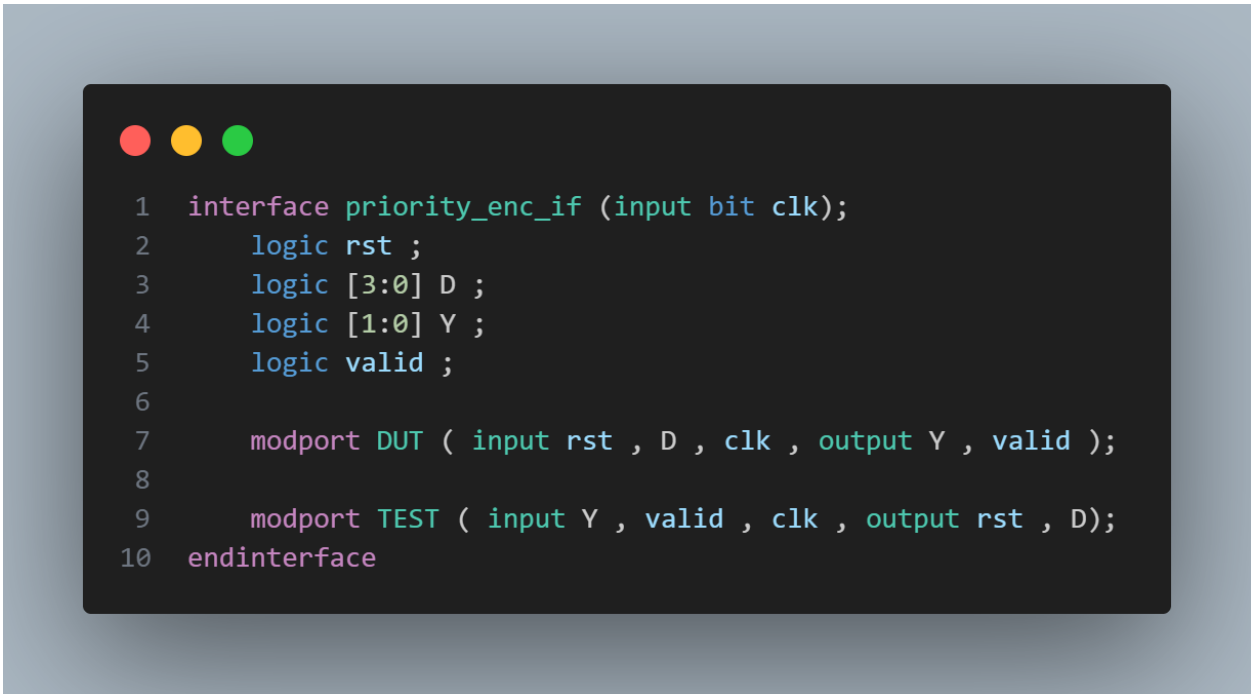
```
module priority_enc_top();
    bit clk ;
    initial begin
        clk = 0;
        forever begin
            #2 clk = ~clk;
        end
    end

    priority_enc_if inst_interface (clk) ;
    priority_enc inst_design (inst_interface) ;
    priority_enc_tb inst_tb (inst_interface) ;
    bind priority_enc priority_enc_sva inst_sva (.enc_sva(enc_DUT)) ;
endmodule
```

3. SVA module code



```
module priority_enc_sva(priority_enc_if.DUT enc_sva);
    property rst_on;
        @(posedge enc_sva.clk) (enc_sva.rst) |=> (enc_sva.valid == 0 && enc_sva.Y == 0) ;
    endproperty

    property rst_off_bit0;
        @(posedge enc_sva.clk) (!enc_sva.rst && enc_sva.D[0]) |=> (enc_sva.Y == 3&& enc_sva.valid == 1) ;
    endproperty

    property rst_off_bit1;
        @(posedge enc_sva.clk) (!enc_sva.rst && !enc_sva.D[0] && enc_sva.D[1] ) |=> (enc_sva.Y == 2&& enc_sva.valid == 1) ;
    endproperty

    property rst_off_bit2;
        @(posedge enc_sva.clk) (!enc_sva.rst && !enc_sva.D[0] && !enc_sva.D[1] && enc_sva.D[2]) |=> (enc_sva.Y == 1&& enc_sva.valid == 1) ;
    endproperty

    property rst_off_bit3;
        @(posedge enc_sva.clk) (!enc_sva.rst && !enc_sva.D[0] && !enc_sva.D[1] && !enc_sva.D[2] && enc_sva.D[3] ) |=> (enc_sva.Y == 0 && enc_sva.valid == 1) ;
    endproperty

    property rst_off_zero;
        @(posedge enc_sva.clk) (!enc_sva.rst && enc_sva.D == 0 ) |=> (enc_sva.valid == 0) ;
    endproperty

    assert property (rst_on) ;
    assert property (rst_off_bit0) ;
    assert property (rst_off_bit1) ;
    assert property (rst_off_bit2) ;
    assert property (rst_off_bit3) ;
    assert property (rst_off_zero) ;

    cover property (rst_on) ;
    cover property (rst_off_bit0) ;
    cover property (rst_off_bit1) ;
    cover property (rst_off_bit2) ;
    cover property (rst_off_bit3) ;
    cover property (rst_off_zero) ;

endmodule
```

4. Interface code



```
interface priority_enc_if (input bit clk);
    logic rst ;
    logic [3:0] D ;
    logic [1:0] Y ;
    logic valid ;

    modport DUT ( input rst , D , clk , output Y , valid );

    modport TEST ( input Y , valid , clk , output rst , D);
endinterface
```

5. Design code

```
module priority_enc (priority_enc_if.DUT enc_DUT);

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

6. DO file

```
vlib work
vlog *v +cover
vsim -voptargs=+acc priority_enc_top -cover -sv_seed random -l sim.log
add wave *
coverage save priority_enc.ucdb -onexit -du priority_enc
run -all
```

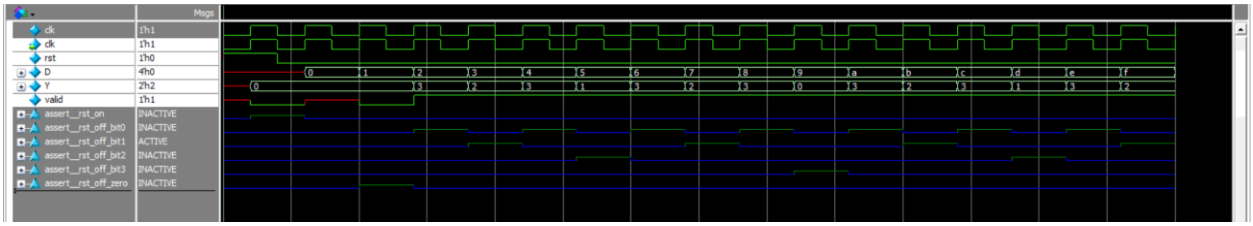
7. Code Coverage & assertion coverage

```
# ================================================================================
# === Instance: /\priority_enc_top#inst_design /inst_sva
# === Design Unit: work.priority_enc_sva
# ================================================================================
#
# Assertion Coverage:
#     Assertions                        6         6         0   100.00%
# --------------------------------------------------------------------
# Name                 File(Line)                    Failure      Pass
#                                                    Count        Count
# --------------------------------------------------------------------
# /\priority_enc_top#inst_design /inst_sva/assert__rst_off_zero
#                      priority_enc_sva.sv(31)             0           1
# /\priority_enc_top#inst_design /inst_sva/assert__rst_off_bit3
#                      priority_enc_sva.sv(30)             0           1
# /\priority_enc_top#inst_design /inst_sva/assert__rst_off_bit2
#                      priority_enc_sva.sv(29)             0           1
# /\priority_enc_top#inst_design /inst_sva/assert__rst_off_bit1
#                      priority_enc_sva.sv(28)             0           1
# /\priority_enc_top#inst_design /inst_sva/assert__rst_off_bit0
#                      priority_enc_sva.sv(27)             0           1
# /\priority_enc_top#inst_design /inst_sva/assert__rst_on
#                      priority_enc_sva.sv(26)             0           1
#
# Directive Coverage:
#     Directives                        6         6         0   100.00%
#
#
# Directive Coverage:
#     Directives                    6         6         0   100.00%
#
# DIRECTIVE COVERAGE:
# ---------------------------------------------------------------------------------------------
# Name                                  Design Design   Lang File(Line)      Hits Status
#                                       Unit   UnitType
# ---------------------------------------------------------------------------------------------
# /\priority_enc_top#inst_design /inst_sva/cover__rst_off_zero
#                                       priority_enc_sva Verilog  SVA  priority_enc_sva.sv(38)
#                                                                        1 Covered
# /\priority_enc_top#inst_design /inst_sva/cover__rst_off_bit3
#                                       priority_enc_sva Verilog  SVA  priority_enc_sva.sv(37)
#                                                                        1 Covered
# /\priority_enc_top#inst_design /inst_sva/cover__rst_off_bit2
#                                       priority_enc_sva Verilog  SVA  priority_enc_sva.sv(36)
#                                                                        2 Covered
# /\priority_enc_top#inst_design /inst_sva/cover__rst_off_bit1
#                                       priority_enc_sva Verilog  SVA  priority_enc_sva.sv(35)
#                                                                        3 Covered
# /\priority_enc_top#inst_design /inst_sva/cover__rst_off_bit0
#                                       priority_enc_sva Verilog  SVA  priority_enc_sva.sv(34)
#                                                                        7 Covered
# /\priority_enc_top#inst_design /inst_sva/cover__rst_on
#                                       priority_enc_sva Verilog  SVA  priority_enc_sva.sv(33)
#                                                                        1 Covered
```

```
#
# ================================================================================
# === Instance: /\priority_enc_top#inst_design
# === Design Unit: work.priority_enc
# ================================================================================
# Branch Coverage:
#     Enabled Coverage              Bins      Hits    Misses  Coverage
#     ----------------              ----      ----    ------  --------
#     Branches                         7         7         0   100.00%
#
# ================================Branch Details================================
#
# Branch Coverage for instance /\priority_enc_top#inst_design
#
#     Line         Item                      Count     Source
#     ----         ----                      -----     ------
#   File priority_enc.sv
# ------------------------------------IF Branch------------------------------------
#     4                                        18     Count coming in to IF
#     4               1                         1       if (enc_DUT.rst) begin
#
#     8               1                        17       else begin
#
# Branch totals: 2 hits of 2 branches = 100.00%
#
# ------------------------------------CASE Branch------------------------------------
#     9                                        17     Count coming in to CASE
#     10              1                         2         4'b1000: enc_DUT.Y <= 0;
#
#     11              1                         2         4'bX100: enc_DUT.Y <= 1;
#
#     12              1                         4         4'bXX10: enc_DUT.Y <= 2;
#
#     13              1                         8         4'bXXX1: enc_DUT.Y <= 3;
#
#                                               1     All False Count
# Branch totals: 5 hits of 5 branches = 100.00%
#
# Statement Coverage:
#     Enabled Coverage              Bins      Hits    Misses  Coverage
#     ----------------              ----      ----    ------  --------
#     Statements                       8         8         0   100.00%
#
# ================================Statement Details================================
#
# Statement Coverage for instance /\priority_enc_top#inst_design  --
#
#     Line         Item                      Count     Source
#     ----         ----                      -----     ------
#   File priority_enc.sv
#     1                                                module priority_enc (priority_enc_if.DUT enc_DUT);
#
#     2
#
#     3               1                        18     always @(posedge enc_DUT.clk) begin
#
#     4                                                 if (enc_DUT.rst) begin
#
#     5               1                         1      enc_DUT.Y <= 2'b0;
#
#     6               1                         1      enc_DUT.valid <= 0 ;
#
#     7                                                 end
#
#     8                                                 else begin
#
#     9                                                 casex (enc_DUT.D)
#
#     10              1                         2         4'b1000: enc_DUT.Y <= 0;
#
#     11              1                         2         4'bX100: enc_DUT.Y <= 1;
#
#     12              1                         4         4'bXX10: enc_DUT.Y <= 2;
#
#     13              1                         8         4'bXXX1: enc_DUT.Y <= 3;
#
#     14                                                endcase
#
#     15              1                        17        enc_DUT.valid <= (~|enc_DUT.D)? 1'b0: 1'b1;
#
#
#
```

```
#                                          Unit   UnitType
# ------------------------------------------------------------------------------------------
# /\priority_enc_top#inst_design /inst_sva/cover__rst_off_zero
#                                          priority_enc_sva Verilog  SVA  priority_enc_sva.sv(38)
#                                                                          1 Covered
# /\priority_enc_top#inst_design /inst_sva/cover__rst_off_bit3
#                                          priority_enc_sva Verilog  SVA  priority_enc_sva.sv(37)
#                                                                          1 Covered
# /\priority_enc_top#inst_design /inst_sva/cover__rst_off_bit2
#                                          priority_enc_sva Verilog  SVA  priority_enc_sva.sv(36)
#                                                                          2 Covered
# /\priority_enc_top#inst_design /inst_sva/cover__rst_off_bit1
#                                          priority_enc_sva Verilog  SVA  priority_enc_sva.sv(35)
#                                                                          3 Covered
# /\priority_enc_top#inst_design /inst_sva/cover__rst_off_bit0
#                                          priority_enc_sva Verilog  SVA  priority_enc_sva.sv(34)
#                                                                          7 Covered
# /\priority_enc_top#inst_design /inst_sva/cover__rst_on
#                                          priority_enc_sva Verilog  SVA  priority_enc_sva.sv(33)
#                                                                          1 Covered
#
# TOTAL DIRECTIVE COVERAGE: 100.00%  COVERS: 6
#
# ASSERTION RESULTS:
# ---------------------------------------------------------------------
# Name                 File(Line)                    Failure      Pass
#                                                    Count        Count
# ---------------------------------------------------------------------
# /\priority_enc_top#inst_design /inst_sva/assert__rst_off_zero
#                      priority_enc_sva.sv(31)             0          1
# /\priority_enc_top#inst_design /inst_sva/assert__rst_off_bit3
#                      priority_enc_sva.sv(30)             0          1
# /\priority_enc_top#inst_design /inst_sva/assert__rst_off_bit2
#                      priority_enc_sva.sv(29)             0          1
# /\priority_enc_top#inst_design /inst_sva/assert__rst_off_bit1
#                      priority_enc_sva.sv(28)             0          1
# /\priority_enc_top#inst_design /inst_sva/assert__rst_off_bit0
#                      priority_enc_sva.sv(27)             0          1
# /\priority_enc_top#inst_design /inst_sva/assert__rst_on
#                      priority_enc_sva.sv(26)             0          1
#
# Total Coverage By Instance (filtered view): 100.00%
#
# End time: 19:27:11 on Sep 18,2025, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
```

8. Questasim snippets



P1) ALU

1. Testbench code

```
import ALU_package::*;
module ALU_tb (ALU_if.TEST ALU_tb);
    localparam MAXPOS = 7 ;
    localparam MAXNEG = -8 ;

    initial begin
        ALU_class obj1 = new ;
        // TEST_1
        ALU_tb.reset = 1 ;
        @(negedge ALU_tb.clk );
        // TEST_2
        ALU_tb.reset = 0 ;
        repeat (200) begin
            @(posedge ALU_tb.clk );
            assert (obj1.randomize()) else $fatal("Randomization failed");
            ALU_tb.reset     = obj1.rst       ;
            ALU_tb.A         = obj1.A         ;
            ALU_tb.B         = obj1.B         ;
            ALU_tb.opcode    = obj1.opcode    ;
        end
        $stop ;
    end
endmodule
```

2. Top module code

```
import ALU_package::*;
module ALU_top ();
    bit clk ;
    initial begin
        clk = 0;
        forever begin
            #2 clk = ~clk;
        end
    end

    ALU_if inst_interface (clk) ;
    ALU_4_bit inst_design (inst_interface);
    ALU_tb inst_tb (inst_interface) ;
    bind ALU_4_bit ALU_sva inst_sva (.ALU_sva(ALU_DUT)) ;
endmodule
```

3. SVA module code

```
module ALU_sva (ALU_if.DUT ALU_sva);
    property reset_on;
        @(posedge ALU_sva.clk) (ALU_sva.reset) |-> (ALU_sva.C == 0) ;
    endproperty

    property reset_off_add;
        @(posedge ALU_sva.clk) (!ALU_sva.reset && ALU_sva.opcode == 0) |=> (ALU_sva.C == $past(ALU_sva.A) + $past(ALU_sva.B)) ;
    endproperty

    property reset_off_sub;
        @(posedge ALU_sva.clk) (!ALU_sva.reset && ALU_sva.opcode == 1) |=> (ALU_sva.C == $past(ALU_sva.A) - $past(ALU_sva.B)) ;
    endproperty

    property reset_off_NotA;
        @(posedge ALU_sva.clk) (!ALU_sva.reset && ALU_sva.opcode == 2) |=> (ALU_sva.C == ~($past(ALU_sva.A))) ;
    endproperty

    property reset_off_orB;
        @(posedge ALU_sva.clk) (!ALU_sva.reset && ALU_sva.opcode == 3) |=> (ALU_sva.C == |($past(ALU_sva.B))) ;
    endproperty

    property default_case;
        @(posedge ALU_sva.clk) (!ALU_sva.reset && (ALU_sva.opcode != 3) && (ALU_sva.opcode != 2) && (ALU_sva.opcode != 1) && (ALU_sva.opcode != 0) ) |=> (ALU_sva.C == 0) ;
    endproperty

    assert property (reset_on) ;
    assert property (reset_off_add) ;
    assert property (reset_off_sub) ;
    assert property (reset_off_NotA);
    assert property (reset_off_orB) ;
    assert property (default_case) ;

    cover property (reset_on) ;
    cover property (reset_off_add) ;
    cover property (reset_off_sub) ;
    cover property (reset_off_NotA);
    cover property (reset_off_orB) ;
    // cover property (default_case) ;
endmodule
```

4. Interface code

```
interface ALU_if(input bit clk);
    import ALU_package::*;
    logic reset;
    opcode_t opcode;    // The opcode
    logic signed [3:0] A;   // Input data A in 2's complement
    logic signed [3:0] B;   // Input data B in 2's complement
    logic signed [4:0] C ;

    modport DUT (
    input clk , reset , A ,B , opcode ,
    output C
    );

    modport TEST (
    input C ,
    output clk , reset , A ,B , opcode
    );
endinterface
```

5. Design code

```
module ALU_4_bit (ALU_if.DUT ALU_DUT);

   reg signed [4:0]      Alu_out; // ALU output in 2's complement

   // Do the operation
   always @* begin
      case (ALU_DUT.opcode)
         0:            Alu_out = ALU_DUT.A + ALU_DUT.B;
         1:            Alu_out = ALU_DUT.A - ALU_DUT.B;
         2:          Alu_out = ~ALU_DUT.A;
         3:  Alu_out = |ALU_DUT.B;
         default:  Alu_out = 5'b0;
      endcase
   end // always @ *

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

```

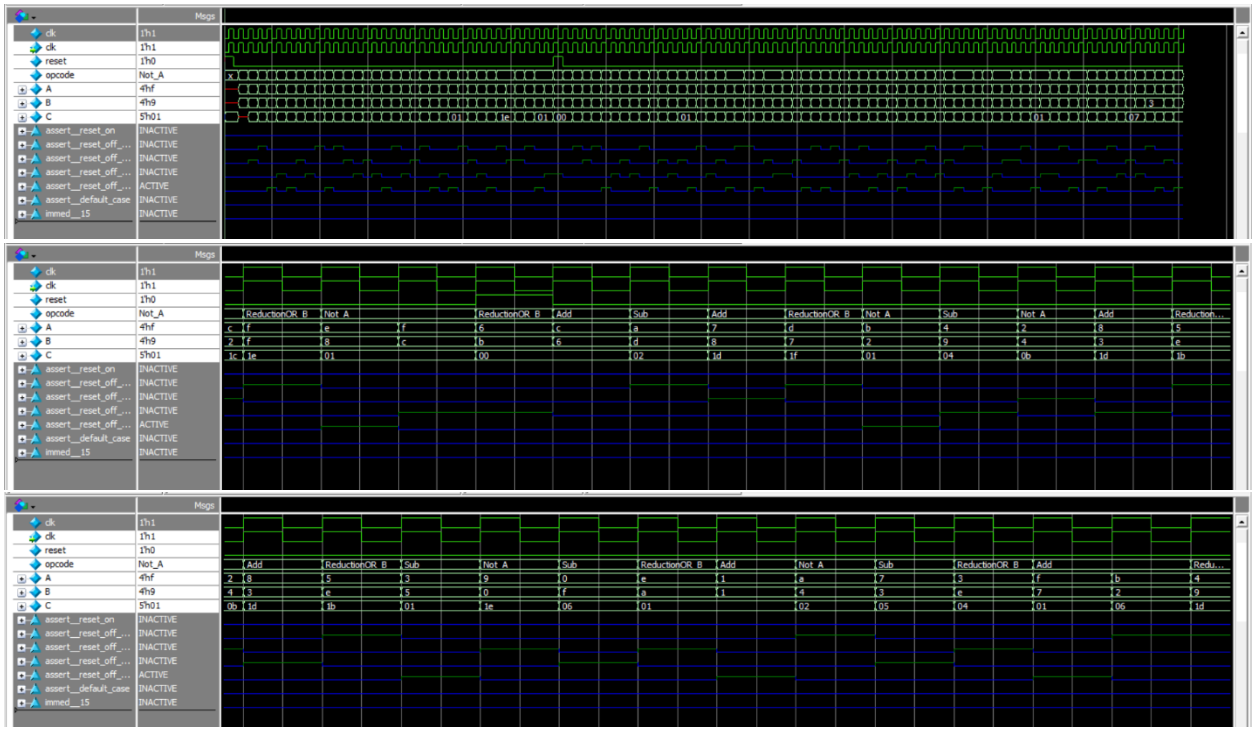
6. Package code

```
package ALU_package ;
    localparam MAXPOS =  7 ;
    localparam MAXNEG = -8 ;
    typedef enum logic [1:0] { Add , Sub , Not_A , ReductionOR_B  } opcode_t ;
    class ALU_class;
        randc logic signed [3:0] A ,B ;
        rand logic rst ;
        randc opcode_t opcode ;

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

    endclass
endpackage
```

7. DO file

```
vlib work
vlog *v +cover
vsim -voptargs=+acc ALU_top -cover -sv_seed random -l sim.log
add wave *
coverage save ALU_4_bit.ucdb -onexit -du ALU_4_bit
run -all
```

8. Questasim snippets



9. Code Coverage & assertion coverage

```
# ================================================================================
# === Instance: /\ALU_top#inst_design
# === Design Unit: work.ALU_4_bit
# ================================================================================
# Branch Coverage:
#     Enabled Coverage              Bins      Hits    Misses  Coverage
#     ----------------              ----      ----    ------  --------
#     Branches                         6         6         0   100.00%
#
# ================================Branch Details================================
#
# Branch Coverage for instance /\ALU_top#inst_design
#
#     Line         Item                      Count     Source
#     ----         ----                      -----     ------
#   File ALU.sv
# ------------------------------------CASE Branch------------------------------------
#     7                                        99     Count coming in to CASE
#     8               1                        25            0:            Alu_out = ALU_DUT.A + ALU_DUT.B;
#
#     9               1                        25            1:            Alu_out = ALU_DUT.A - ALU_DUT.B;
#
#     10              1                        24            2:          Alu_out = ~ALU_DUT.A;
#
#     11              1                        25              3:  Alu_out = |ALU_DUT.B;
#
# Branch totals: 4 hits of 4 branches = 100.00%
#
# ------------------------------------IF Branch------------------------------------
#     18                                      103     Count coming in to IF
#     18              1                         4           if (ALU_DUT.reset)
#
#     20              1                        99           else
#
# Branch totals: 2 hits of 2 branches = 100.00%
#
#
```

```
#
# Statement Coverage for instance /\ALU_top#inst_design  --
#
#     Line         Item                      Count     Source
#     ----         ----                      -----     ------
#   File ALU.sv
#     1                                                module ALU_4_bit (ALU_if.DUT ALU_DUT);
#
#     2
#
#     3                                                   reg signed [4:0]      Alu_out; // ALU output in 2's complement
#
#     4
#
#     5                                                   // Do the operation
#
#     6               1                        99         always @* begin
#
#     7                                                      case (ALU_DUT.opcode)
#
#     8               1                        25             0:            Alu_out = ALU_DUT.A + ALU_DUT.B;
#
#     9               1                        25             1:            Alu_out = ALU_DUT.A - ALU_DUT.B;
#
#     10              1                        24             2:          Alu_out = ~ALU_DUT.A;
#
#     11              1                        25               3:  Alu_out = |ALU_DUT.B;
#
#     12                                                       default:  Alu_out = 5'b0;
#
#     13                                                    endcase
#
#     14                                                 end // always @ *
#
#     15
#
#     16                                                 // Register output C
#
#     17              1                       103         always @(posedge ALU_DUT.clk or posedge ALU_DUT.reset) begin
#
```

```
# ================================Toggle Details================================
#
# Toggle Coverage for instance /\ALU_top#inst_design  --
#
#                                          Node      1H->0L      0L->1H                            "Coverage"
#                                          ----------------------------------------------------------------
#                                    Alu_out[4-0]           1           1                               100.00
#
# Total Node Count     =          5
# Toggled Node Count   =          5
# Untoggled Node Count =          0
#
# Toggle Coverage      =     100.00% (10 of 10 bins)
#
#
#
# DIRECTIVE COVERAGE:
# ---------------------------------------------------------------------------------------------------
# Name                                     Design Design   Lang File(Line)      Hits Status
#                                          Unit   UnitType
# ---------------------------------------------------------------------------------------------------
# /\ALU_top#inst_design /inst_sva/cover__reset_off_orB
#                                          ALU_sva Verilog  SVA  ALU_sva.sv(37)    23 Covered
# /\ALU_top#inst_design /inst_sva/cover__reset_off_NotA
#                                          ALU_sva Verilog  SVA  ALU_sva.sv(36)    24 Covered
# /\ALU_top#inst_design /inst_sva/cover__reset_off_sub
#                                          ALU_sva Verilog  SVA  ALU_sva.sv(35)    25 Covered
# /\ALU_top#inst_design /inst_sva/cover__reset_off_add
#                                          ALU_sva Verilog  SVA  ALU_sva.sv(34)    24 Covered
# /\ALU_top#inst_design /inst_sva/cover__reset_on
#                                          ALU_sva Verilog  SVA  ALU_sva.sv(33)     2 Covered
#
# TOTAL DIRECTIVE COVERAGE: 100.00%  COVERS: 5
#
# ASSERTION RESULTS:
# ---------------------------------------------------------------------
# Name                  File(Line)                    Failure      Pass
#                                                     Count        Count
# ---------------------------------------------------------------------
# /\ALU_top#inst_design /inst_sva/assert__default_case
#                       ALU_sva.sv(31)                0          0
# /\ALU_top#inst_design /inst_sva/assert__reset_off_orB
#                       ALU_sva.sv(30)                0          1
# /\ALU_top#inst_design /inst_sva/assert__reset_off_NotA
#                       ALU_sva.sv(29)                0          1
# /\ALU_top#inst_design /inst_sva/assert__reset_off_sub
#                       ALU_sva.sv(28)                0          1
# /\ALU_top#inst_design /inst_sva/assert__reset_off_add
#                       ALU_sva.sv(27)                0          1
# /\ALU_top#inst_design /inst_sva/assert__reset_on
#                       ALU_sva.sv(26)                0          1
#
# Total Coverage By Instance (filtered view): 96.66%
#
# End time: 19:52:50 on Sep 18,2025, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
```

P2)

```
module P2();
    bit requst ;
    bit grant ;
    bit clk ;
    bit frame ;
    bit irdy ;
    bit request ;

    property assert1;
        @(posedge clk) $rose(request) |=> ##[2:5] (grant) ;
    endproperty

    property assert2;
        @(posedge clk) $rose(grant) |-> ($fell(frame) && $fell(irdy)) ;
    endproperty

    property assert3;
        @(posedge clk) ($rose(frame) && $rose(irdy)) |=> ($fell(grant)) ;
    endproperty
endmodule
```

```
VSIM(paused)>
# Compile of module.sv was successful.
```

P3)

```
module P3 ();
    enum logic [1:0] {IDLE , GEN_BLK_ADDR , WAITO} cs ;
    bit clk ;
    bit get_data;

    property cs_property;
        @(posedge clk) $onehot(cs) ;
    endproperty
    // here cs nust be GEN_BLK_ADDR just one clock after (cs == IDLE && get_data) and after 64 clock it must be WAITO
    property cs_IDLE;
        @(posedge clk) (cs == IDLE && get_data ) |=> (cs == GEN_BLK_ADDR) ##64 (cs == WAITO) ;
    endproperty
    //  here cs nust be GEN_BLK_ADDR just one clock after (cs == IDLE && get_data) and stay the same for 64 clock and after that must be WAITO
    property cs_IDLE2;
        @(posedge clk) (cs == IDLE && get_data) |=> (cs == GEN_BLK_ADDR[*64] ##1 (cs == WAITO));
    endproperty

endmodule
```