يوسف أحمد محمد ابراهيم

Assignment 2

Question 1:

1. code :

```
module dyn_arr();
    int dyn_array1[];
    int dyn_array2[];


    initial begin
        dyn_array2 = '{9, 1, 8, 3, 4, 4};
        dyn_array1 = new[6];

        foreach (dyn_array1[j]) dyn_array1[j] = j;

        $display("*** Array 1 : %p, size = %0d ***", dyn_array1, $size(dyn_array1));

        dyn_array1.delete();


        dyn_array2.reverse();
        $display("*** Reversed Array 2 %p ***", dyn_array2);
        dyn_array2.sort();
        $display("*** Sorted Array 2 : %p ***", dyn_array2);
        dyn_array2.rsort();
        $display("*** Reverse sorted Array 2 : %p ***", dyn_array2);
        dyn_array2.shuffle();
        $display("*** shuffled  Array 2 : %p ***", dyn_array2);

    end
endmodule
```

2. Results :

```
# *** Array 1 : '{0, 1, 2, 3, 4, 5}, size = 6 ***
# *** Reversed Array 2 '{4, 4, 3, 8, 1, 9} ***
# *** Sorted Array 2 : '{1, 3, 4, 4, 8, 9} ***
# *** Reverse sorted Array 2 : '{9, 8, 4, 4, 3, 1} ***
# *** shuffled  Array 2 : '{8, 1, 4, 3, 9, 4} ***
```

Question 2:

1. RTL design:

```
module counter (clk ,rst_n, load_n, up_down, ce, data_load, count_out, max_count, zero);
parameter WIDTH = 4;
input clk;
input rst_n;
input load_n;
input up_down;
input ce;
input [WIDTH-1:0] data_load;
output reg [WIDTH-1:0] count_out;
output max_count;
output zero;

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

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

endmodule
```

The design is working properly.

2. Verification plan:

Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
RESET_TEST	When the reset is asserted, outputs should be low.	Directed at the start of the simulation, and called in radomization step.	-	**check_result()** verifies correct output using golden model task **exp_out**.
RANDOMIZATION_TEST	Output of the dut design is equal to the output of the golden model	Randomized test using **rand_stimuls** class with constraints 90% deactive reset, 90% active clock enable, 70% low up_down, 70% active clock enable and 70% active load enable.	-	**check_result()** verifies correct output using golden model task **exp_out**.

3. packege code:

```
package counter_pkg;
    parameter WIDTH = 4;
    class rand_stimuls;
        rand bit             rst_n;
        rand bit             load_n;
        rand bit             up_down;
        rand bit             ce;
        rand bit [WIDTH-1:0] data_load;

        //no need for constructor, they will be initialized to 0

        constraint c1 {
            rst_n dist {1 := 9, 0 := 1};
            ce dist {1 := 9, 0 := 1};
            load_n dist {0 := 7, 1 := 3}; //load_n active 70%
        }
    endclass
endpackage
```

4. Testbench code:

```
import counter_pkg::*;

module counter_tb();
    parameter WIDTH = 4;
    bit             clk;
    bit             rst_n;
    bit             load_n;
    bit             up_down;
    bit             ce;
    bit [WIDTH-1:0] data_load;

    bit [WIDTH-1:0] count_out;
    bit             max_count;
    bit             zero;

    bit [WIDTH-1:0] count_out_exp;
    bit             max_count_exp;
    bit             zero_exp;

    rand_stimuls my_inputs;
    integer error_count, correct_count;

    counter dut(.*);

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

    initial begin
        error_count = 0;
        correct_count = 0;
        load_n = 0;
        up_down = 0;
        ce = 0;
        data_load = 0;

        //reset test
        rst_n = 0;
        @(negedge clk);
        check_result();

        //randomized test
        my_inputs = new();
        for(int i = 0; i < 50; i++) begin
            assert(my_inputs.randomize());
            rst_n = my_inputs.rst_n;
            load_n = my_inputs.load_n;
            up_down = my_inputs.up_down;
            ce = my_inputs.ce;
            data_load = my_inputs.data_load;

            check_result();
        end
        $display("*** ERROR count: %0d, CORRECT count: %0d", error_count, correct_count);
        $stop;
    end
```

4. Testbench code:

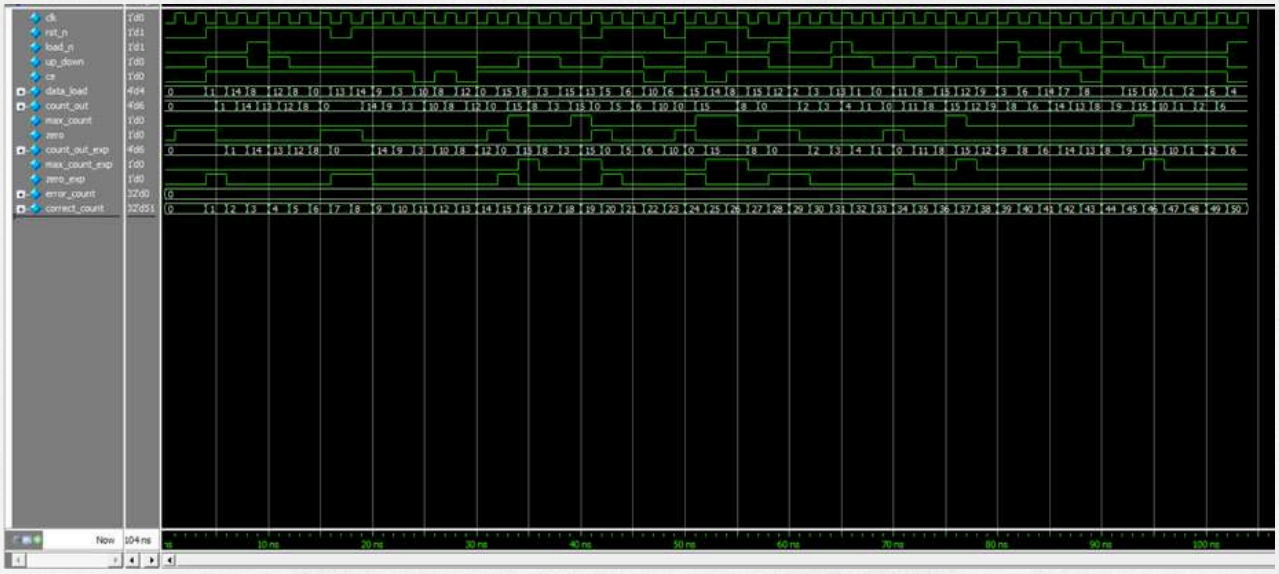
```
    task check_result();
        @(negedge clk);
        exp_out(); //calculate the correct outputs
        if(count_out != count_out_exp || max_count != max_count_exp || zero != zero_exp) begin
            $display("*** ERROR! at time %0t, output = %0d, max_count = %0d, zero = %0d | EXPECTED : %0d, %0d, %0d ***",
                $time, count_out, max_count, zero, count_out_exp, max_count_exp, zero_exp);
            error_count++;
        end
        else
            correct_count++;

    endtask


    task exp_out();
        if (!rst_n) begin
            count_out_exp = 0;
            max_count_exp = 0;
            zero_exp = 1;
        end
        else if(!load_n)
            count_out_exp = data_load;
        else if(ce)
            if(up_down)
                count_out_exp++;
            else
                count_out_exp--;

        if(count_out_exp == {WIDTH{1'b1}})
            max_count_exp = 1;
        else
            max_count_exp = 0;

        if(count_out_exp == 0)
            zero_exp = 1;
        else
            zero_exp = 0;
    endtask
endmodule
```

5. Do file :

```
vlib work
vlog 2_pkg.sv 2_counter.v 2_counter_tb.sv +cover -covercells
vsim -voptargs=+acc work.counter_tb -cover
add wave *
coverage save 2_counter_tb.ucdb -onexit -du work.counter
run -all
```

No exceptions needed.

6. Qesta sim wave snippets (unsigned radix)




```
** ERROR count: 0, CORRECT count: 51
** Note: $stop    : 2_counter_tb.sv(57)
   Time: 104 ns  Iteration: 1  Instance: /counter_tb
Break in Module counter_tb at 2_counter_tb.sv line 57
```

7. Statement coverage report :

```
Statement Coverage:
    Enabled Coverage             Bins      Hits    Misses  Coverage
    ----------------             ----      ----    ------  --------
    Statements                      7         7         0   100.00%

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

Statement Coverage for instance /\counter_tb#dut  --

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File 2_counter.v
    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

    20              1                         50     always @(posedge clk) begin

    21                                                   if (!rst_n)

    22              1                         15           count_out <= 0;

    23                                                   else if (!load_n)

    24              1                         28           count_out <= data_load;

    25                                                   else if (ce)

    26                                                       if (up_down)

    27              1                          1               count_out <= count_out + 1;

    28                                                       else

    29              1                          2               count_out <= count_out - 1;

    30                                               end

    31

    32              1                         39     assign max_count = (count_out == {WIDTH{1'b1}})? 1:0;

    33              1                         39     assign zero = (count_out == 0)? 1:0;
```

8. Branch coverage report :

```
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        10        10         0   100.00%

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

Branch Coverage for instance /\counter_tb#dut

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File 2_counter.v
------------------------------------IF Branch------------------------------------
    21                                        50     Count coming in to IF
    21              1                         15         if (!rst_n)

    23              1                         28         else if (!load_n)

    25              1                          3         else if (ce)

                                               4     All False Count
Branch totals: 4 hits of 4 branches = 100.00%

------------------------------------IF Branch------------------------------------
    26                                         3     Count coming in to IF
    26              1                          1             if (up_down)

    28              1                          2             else

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    32                                        38     Count coming in to IF
    32              1                          5     assign max_count = (count_out == {WIDTH{1'b1}})? 1:0;

    32              2                         33     assign max_count = (count_out == {WIDTH{1'b1}})? 1:0;

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    33                                        38     Count coming in to IF
    33              1                          9     assign zero = (count_out == 0)? 1:0;

    33              2                         29     assign zero = (count_out == 0)? 1:0;

Branch totals: 2 hits of 2 branches = 100.00%


Condition Coverage:
    Enabled Coverage              Bins   Covered    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Conditions                       2         2         0   100.00%
```

9. Toggle coverage report :

```
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         30        30         0   100.00%

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

Toggle Coverage for instance /\counter_tb#dut  --

                                                  Node      1H->0L      0L->1H  "Coverage"
                                                  ----------------------------------------
                                                    ce           1           1      100.00
                                                   clk           1           1      100.00
                                          count_out[3-0]         1           1      100.00
                                          data_load[0-3]         1           1      100.00
                                                 load_n          1           1      100.00
                                              max_count          1           1      100.00
                                                 rst_n           1           1      100.00
                                               up_down           1           1      100.00
                                                  zero           1           1      100.00

Total Node Count      =         15
Toggled Node Count    =         15
Untoggled Node Count =          0

Toggle Coverage       =     100.00% (30 of 30 bins)


Total Coverage By Instance (filtered view): 100.00%
```

Question 3:

1. RTL 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, cin, rst, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
input [2:0] opcode;
input signed [2:0] A, B;
output reg [15:0] leds;
output reg signed [5:0] out;

reg red_op_A_reg, red_op_B_reg, bypass_A_reg, bypass_B_reg, direction_reg, serial_in_reg;
reg signed [1:0] cin_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
```

1. RTL design:

```
61  //ALSU output processing
62  always @(posedge clk or posedge rst) begin
63    if(rst) begin
64      out <= 0;
65    end
66    else begin
67      if (bypass_A_reg && bypass_B_reg)
68        out <= (INPUT_PRIORITY == "A")? A_reg: B_reg;
69      else if (bypass_A_reg)
70        out <= A_reg;
71      else if (bypass_B_reg)
72        out <= B_reg;
73      else if (invalid)
74          out <= 0;
75      else begin
76          case (opcode_reg)
77            3'h0: begin
78              if (red_op_A_reg && red_op_B_reg)
79                out <= (INPUT_PRIORITY == "A")? |A_reg: |B_reg;
80              else if (red_op_A_reg)
81                out <= |A_reg;
82              else if (red_op_B_reg)
83                out <= |B_reg;
84              else
85                out <= A_reg | B_reg;
86            end
87            3'h1: begin
88              if (red_op_A_reg && red_op_B_reg)
89                out <= (INPUT_PRIORITY == "A")? ^A_reg: ^B_reg;
90              else if (red_op_A_reg)
91                out <= ^A_reg;
92              else if (red_op_B_reg)
93                out <= ^B_reg;
94              else
95                out <= A_reg ^ B_reg;
96            end
97            3'h2: out <= (FULL_ADDER)? (A_reg + B_reg + cin_reg) : A_reg + B_reg;
98            3'h3: out <= A_reg * B_reg;
99            3'h4: begin
100             if (direction_reg)
101               out <= {out[4:0], serial_in_reg};
102             else
103               out <= {serial_in_reg, out[5:1]};
104           end
105           3'h5: begin
106             if (direction_reg)
107               out <= {out[4:0], out[5]};
108             else
109               out <= {out[0], out[5:1]};
110           end
111           default : out <= 0;
112         endcase
113     end
114   end
115 end
116
117 endmodule
```

1. RTL design:

Found and fixed the following bugs:

1. opcode used in the case statement rather than opcode_reg.
2. adder case didn't use cin according to **FULL_ADDER** parameter.

2. Verification plan:

	Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
2	RESET_TEST	When the reset is asserted, outputs should be low.	Directed at the start of the simulation, and called in other radomization steps.	-	**check_result()** verifies correct output using a golden model().
3	RANDOMIZATION_TEST	Output of the dut design should equal to the output of the golden model with the rest of input randomized.	ALL inputs are randomized using the calss with the following constraints: **rst** is 90% low, any **OPCODE** have Invalid value to valid one is 1:3, and **A/B** have one-hot value to any other value is 2:1 when **red_op_A/red_op_B** is high and **OPCODE = OR/XOR.** And **A/B** have {MAXPOS, ZERO, MAXNG} value to any other value = 2 : 1 when **OPCODE** is MULT or ADD. **bypass_A/bypass_B** are 30% high. **red_op_A/red_op_B** are 30% low when **OPCODE** is not OR , XOR.	-	**check_result()** verifies correct output using a golden model().

3. packege code:

```
package ALSU_pkg;

    typedef enum bit[2:0] {
        OR,
        XOR,
        ADD,
        MULT,
        SHIFT,
        ROTATE,
        INVALID_6,
        INVALID_7
    } opcode_e;

    parameter MAXPOS = 3'b011;
    parameter ZERO = 3'b000;
    parameter MAXNEG = 3'b100;

    class rand_stimuls;
        rand bit  [2:0] A;
        rand bit  [2:0] B;
        rand bit        rst;
        rand bit        red_op_A;
        rand bit        red_op_B;
        rand bit        bypass_A;
        rand bit        bypass_B;
        rand bit        cin;
        rand bit        serial_in;
        rand bit        direction;
        rand opcode_e   opcode;

        //no need for constructor, they will be initialized to 0

        constraint c1 {
            rst dist {0 := 9, 1 := 1};


            opcode dist {[OR:ROTATE] := 3, [INVALID_6:INVALID_7] := 1};

            bypass_A dist {1 := 3, 0 := 7};
            bypass_B dist {1 := 3, 0 := 7};

```

3. packege code:

```
42              if(opcode == OR || opcode == XOR) {
43                  if(red_op_A) { //priority for A so red_op_B here doesn't matter
44                      A dist {
45                          [3'b000:3'b111] := 1,
46                          3'b001 := 2,
47                          3'b010 := 2,
48                          3'b100 := 2
49                          };
50                      B == 3'b000;
51                  }
52
53                  else if(red_op_B){
54                      A == 3'b000;
55                      B dist {
56                          [3'b000:3'b111] := 1,
57                          3'b001 := 2,
58                          3'b010 := 2,
59                          3'b100 := 2
60                          };
61                  }
62              }
63
64              else {
65                  red_op_A dist {0 := 7, 1 := 3};
66                  red_op_B dist {0 := 7, 1 := 3};
67              }
68
69              if(opcode == ADD || opcode == MULT) {
70                  A dist {[3'b001:3'b110] := 1, MAXPOS := 2, ZERO := 2, MAXNEG := 2};
71                  B dist {[3'b001:3'b110] := 1, MAXPOS := 2, ZERO := 2, MAXNEG := 2};
72              }
73          }
74      endclass
75  endpackage
```

4. Testbench code:

```
import ALSU_pkg::*;

module ALSU_tb();
    parameter INPUT_PRIORITY = "A";
    parameter FULL_ADDER = "ON";
    bit signed [2:0]  A;
    bit signed [2:0]  B;
    bit               cin;
    bit               serial_in;
    bit               red_op_A;
    bit               red_op_B;
    opcode_e          opcode;
    bit               bypass_A;
    bit               bypass_B;
    bit               clk;
    bit               rst;
    bit               direction;
    bit        [15:0] leds;
    bit signed [5:0]  out;

    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  [1:0] cin_reg;
    reg signed  [2:0] A_reg, B_reg;

    bit               invalid;

    rand_stimuls my_inputs; //handle

    bit [15:0]  leds_exp;
    bit [5:0]   out_exp;

    integer correct_count, error_count;

    ALSU dut(.*);

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


    initial begin
        correct_count = 0;
        error_count   = 0;
        A = 0;
        B = 0;
        cin = 0;
        serial_in = 0;
        red_op_A = 0;
        red_op_B = 0;
        bypass_A = 0;
        bypass_B = 0;
        direction = 0;
```

4. Testbench code:

```
        //reset test
        rst = 1;
        check_result();

        my_inputs = new();
        //random test
        for(int i = 0; i<1000; i++) begin
            assert(my_inputs.randomize());

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

            check_result();
        end

        $display("*** ERROR count: %0d, CORRECT count: %0d", error_count, correct_count);
        $stop;
    end

    //golden model
    assign invalid = (red_op_A_reg | red_op_B_reg) & (opcode_reg[1] | opcode_reg[2])
        ||(opcode_reg[1] & opcode_reg[2]);

    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
```

4. Testbench code:

```
115     always @(posedge clk or posedge rst) begin
116         if(rst) begin
117             leds_exp <= 0;
118         end else begin
119             if (invalid)
120                 leds_exp <= ~leds_exp;
121             else
122                 leds_exp <= 0;
123         end
124     end
125 
126     always @(posedge clk or posedge rst) begin
127         if(rst)
128             out_exp <= 0;
129         else begin
130             if (bypass_A_reg && bypass_B_reg)
131                 out_exp <= A_reg;
132             else if (bypass_A_reg)
133                 out_exp <= A_reg;
134             else if (bypass_B_reg)
135                 out_exp <= B_reg;
136             else if (invalid)
137                 out_exp <= 0;
138             else begin
139                 case (opcode_reg)
140                 3'h0: begin
141                     if (red_op_A_reg && red_op_B_reg)
142                         out_exp <= |A_reg;
143                     else if (red_op_A_reg)
144                         out_exp <= |A_reg;
145                     else if (red_op_B_reg)
146                         out_exp <= |B_reg;
147                     else
148                         out_exp <= A_reg | B_reg;
149                 end
150                 3'h1: begin
151                     if (red_op_A_reg && red_op_B_reg)
152                         out_exp <= ^A_reg;
153                     else if (red_op_A_reg)
154                         out_exp <= ^A_reg;
155                     else if (red_op_B_reg)
156                         out_exp <= ^B_reg;
157                     else
158                         out_exp <= A_reg ^ B_reg;
159                 end
```

4. Testbench code:

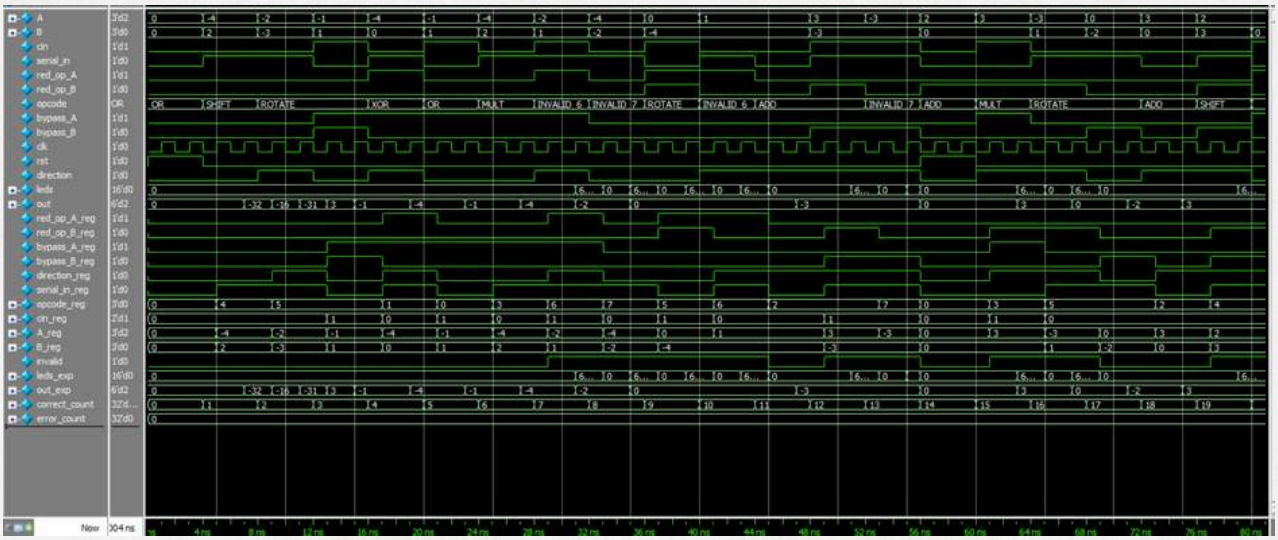
```
                3'h2: out_exp <= A_reg + B_reg + cin_reg;
                3'h3: out_exp <= A_reg * B_reg;
                3'h4: begin
                    if (direction_reg)
                        out_exp <= {out_exp[4:0], serial_in_reg};
                    else
                        out_exp <= {serial_in_reg, out_exp[5:1]};
                end
                3'h5: begin
                    if (direction_reg)
                        out_exp <= {out_exp[4:0], out_exp[5]};
                    else
                        out_exp <= {out_exp[0], out_exp[5:1]};
                end
                endcase
            end
        end
    end

    task check_result();
        @(negedge clk);
        @(negedge clk);
        if(out != out_exp || leds != leds_exp) begin
            $display("*** ERROR! at time %0t, out = %0d, Expected : %0d, leds = %0d, Expected : %0d ***"
                , $time, out, out_exp, leds, leds_exp);
            error_count++;
        end
        else
            correct_count++;
    endtask
endmodule
```

5. Do file :

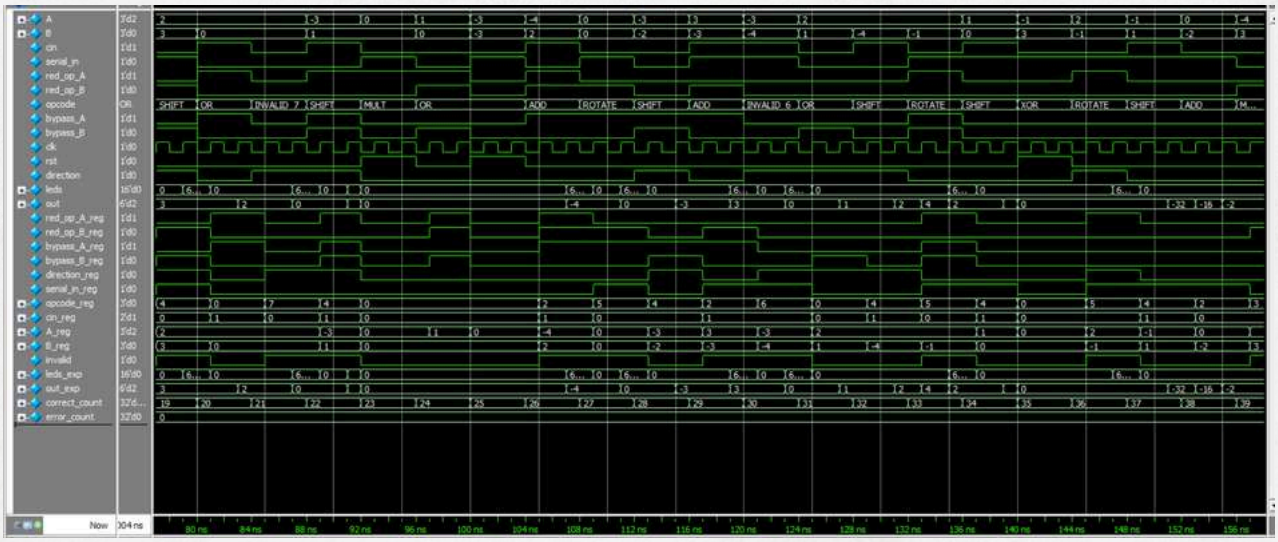
```
vlib work
vlog 3_pkg.sv 3_ALSU.v 3_ALSU_tb.sv +cover -covercells
vsim -voptargs=+acc work.ALSU_tb -cover
add wave *
coverage save 3_ALSU_tb.ucdb -onexit -du work.ALSU
coverage exclude -du ALSU -togglenode {cin_reg[1]}
coverage exclude -src 3_ALSU.v -line 111 -code b
coverage exclude -src 3_ALSU.v -line 111 -code s

run -all
```

6. Qesta sim wave snippets (decimal radix for A, B, OUT)

- first 80ns



- second 80ns



```
# *** ERROR count: 0, CORRECT count: 1001
# ** Note: $stop    : 3_ALSU_tb.sv(82)
#    Time: 4004 ns  Iteration: 1  Instance: /ALSU_tb
# Break in Module ALSU_tb at 3_ALSU_tb.sv line 82
```

7. Statement coverage report :

```
Statement Coverage:
    Enabled Coverage             Bins      Hits    Misses  Coverage
    ----------------             ----      ----    ------  --------
    Statements                     48        48         0   100.00%

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

Statement Coverage for instance /\ALSU_tb#dut  --

    Line         Item                     Count     Source
    ----         ----                     -----     ------
  File 3_ALSU.v
    1                                               module ALSU(A, B, cin, serial_in, red_op_A, red_op_B, opcode, bypass_A, bypass_B, clk, rst, direction, leds, out);
    2                                               parameter INPUT_PRIORITY = "A";
    3                                               parameter FULL_ADDER = "ON";
    4                                               input clk, cin, rst, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
    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 red_op_A_reg, red_op_B_reg, bypass_A_reg, bypass_B_reg, direction_reg, serial_in_reg;
    11                                              reg signed [1:0] cin_reg;
    12                                              reg [2:0] opcode_reg;
    13                                              reg signed [2:0] A_reg, B_reg;
    14
    15                                              wire invalid_red_op, invalid_opcode, invalid;
    16
    17                                              //Invalid handling
    18              1                        929    assign invalid_red_op = (red_op_A_reg | red_op_B_reg) & (opcode_reg[1] | opcode_reg[2]);
    19              1                        844    assign invalid_opcode = opcode_reg[1] & opcode_reg[2];
    20              1                        495    assign invalid = invalid_red_op | invalid_opcode;
    21
    22                                              //Registering input signals
    23              1                       1989    always @(posedge clk or posedge rst) begin
    24                                                if(rst) begin
    25              1                        199        cin_reg <= 0;
    26              1                        199        red_op_B_reg <= 0;
    27              1                        199        red_op_A_reg <= 0;
    28              1                        199        bypass_B_reg <= 0;
```

7. Statement coverage report :

```
28          1                199          bypass_B_reg <= 0;
29          1                199          bypass_A_reg <= 0;
30          1                199          direction_reg <= 0;
31          1                199          serial_in_reg <= 0;
32          1                199          opcode_reg <= 0;
33          1                199          A_reg <= 0;
34          1                199          B_reg <= 0;
35                                      end else begin
36          1               1790          cin_reg <= cin;
37          1               1790          red_op_B_reg <= red_op_B;
38          1               1790          red_op_A_reg <= red_op_A;
39          1               1790          bypass_B_reg <= bypass_B;
40          1               1790          bypass_A_reg <= bypass_A;
41          1               1790          direction_reg <= direction;
42          1               1790          serial_in_reg <= serial_in;
43          1               1790          opcode_reg <= opcode;
44          1               1790          A_reg <= A;
45          1               1790          B_reg <= B;
46                                      end
47                                    end
48
49                                    //leds output blinking
50          1               2095    always @(posedge clk or posedge rst) begin
51                                      if(rst) begin
52          1                305          leds <= 0;
53                                      end else begin
54                                          if (invalid)
55          1                645            leds <= ~leds;
56                                          else
57          1               1145            leds <= 0;
58                                      end
59                                    end
```

7. Statement coverage report :

```
59                                  end
60
61                                  //ALSU output processing
62          1                1816   always @(posedge clk or posedge rst) begin
63                                    if(rst) begin
64          1                 186       out <= 0;
65                                    end
66                                    else begin
67                                      if (bypass_A_reg && bypass_B_reg)
68          1                 162         out <= (INPUT_PRIORITY == "A")? A_reg: B_reg;
69                                      else if (bypass_A_reg)
70          1                 336         out <= A_reg;
71                                      else if (bypass_B_reg)
72          1                 366         out <= B_reg;
73                                      else if (invalid)
74          1                 243           out <= 0;
75                                      else begin
76                                          case (opcode_reg)
77                                            3'h0: begin
78                                              if (red_op_A_reg && red_op_B_reg)
79          1                  15                out <= (INPUT_PRIORITY == "A")? |A_reg: |B_reg;
80                                              else if (red_op_A_reg)
81          1                  26                out <= |A_reg;
82                                              else if (red_op_B_reg)
83          1                  15                out <= |B_reg;
84                                              else
85          1                 151                out <= A_reg | B_reg;
86                                            end
87                                            3'h1: begin
88                                              if (red_op_A_reg && red_op_B_reg)
89          1                   6                out <= (INPUT_PRIORITY == "A")? ^A_reg: ^B_reg;
```

7. Statement coverage report :

90			else if (red_op_A_reg)
91	1	20	out <= ^A_reg;
92			else if (red_op_B_reg)
93	1	26	out <= ^B_reg;
94			else
95	1	65	out <= A_reg ^ B_reg;
96			end
97	1	49	3'h2: out <= (FULL_ADDER)? (A_reg + B_reg + cin_reg) : A_reg + B_reg;
98	1	50	3'h3: out <= A_reg * B_reg;
99			3'h4: begin
100			if (direction_reg)
101	1	21	out <= {out[4:0], serial_in_reg};
102			else
103	1	28	out <= {serial_in_reg, out[5:1]};
104			end
105			3'h5: begin
106			if (direction_reg)
107	1	26	out <= {out[4:0], out[5]};
108			else
109	1	25	out <= {out[0], out[5:1]};

8. Branch coverage report:

```
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        31        31         0   100.00%

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

Branch Coverage for instance /\ALSU_tb#dut

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File 3_ALSU.v
------------------------------------IF Branch------------------------------------
    24                                      1989     Count coming in to IF
    24              1                        199       if(rst) begin

    35              1                       1790       end else begin

  Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    51                                      2095     Count coming in to IF
    51              1                        305       if(rst) begin

    53              1                       1790       end else begin

  Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    54                                      1790     Count coming in to IF
    54              1                        645           if (invalid)

    56              1                       1145           else

  Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    63                                      1816     Count coming in to IF
    63              1                        186       if(rst) begin

    66              1                       1630       else begin

  Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    67                                      1630     Count coming in to IF
    67              1                        162         if (bypass_A_reg && bypass_B_reg)

    69              1                        336         else if (bypass_A_reg)

    71              1                        366         else if (bypass_B_reg)

    73              1                        243         else if (invalid)

    75              1                        523         else begin

  Branch totals: 5 hits of 5 branches = 100.00%
```

8. Branch coverage report:

```
------------------------------------CASE Branch------------------------------------
    76                                  523     Count coming in to CASE
    77              1                   207             3'h0: begin

    87              1                   117             3'h1: begin

    97              1                    49             3'h2: out <= (FULL_ADDER)? (A_reg + B_reg + cin_reg) : A_reg + B_reg;

    98              1                    50             3'h3: out <= A_reg * B_reg;

    99              1                    49             3'h4: begin

    105             1                    51             3'h5: begin

Branch totals: 6 hits of 6 branches = 100.00%

------------------------------------IF Branch------------------------------------
    78                                  207     Count coming in to IF
    78              1                    15                 if (red_op_A_reg && red_op_B_reg)

    80              1                    26                 else if (red_op_A_reg)

    82              1                    15                 else if (red_op_B_reg)

    84              1                   151                 else

Branch totals: 4 hits of 4 branches = 100.00%

------------------------------------IF Branch------------------------------------
    88                                  117     Count coming in to IF
    88              1                     6                 if (red_op_A_reg && red_op_B_reg)

    90              1                    20                 else if (red_op_A_reg)

    92              1                    26                 else if (red_op_B_reg)

    94              1                    65                 else

Branch totals: 4 hits of 4 branches = 100.00%

------------------------------------IF Branch------------------------------------
    100                                 49      Count coming in to IF
    100             1                   21                 if (direction_reg)

    102             1                   28                 else

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    106                                 51      Count coming in to IF
    106             1                   26                 if (direction_reg)

    108             1                   25                 else

Branch totals: 2 hits of 2 branches = 100.00%
```

9. Toggle coverage report:

```
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                        118       118         0   100.00%

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

Toggle Coverage for instance /\ALSU_tb#dut  --

                                    Node      1H->0L      0L->1H                          "Coverage"
                                    ----------------------------------------------------------------
                                   A[0-2]           1           1                              100.00
                                 A_reg[2-0]         1           1                              100.00
                                   B[0-2]           1           1                              100.00
                                 B_reg[2-0]         1           1                              100.00
                                  bypass_A          1           1                              100.00
                                bypass_A_reg        1           1                              100.00
                                  bypass_B          1           1                              100.00
                                bypass_B_reg        1           1                              100.00
                                    cin             1           1                              100.00
                                 cin_reg[0]         1           1                              100.00
                                    clk             1           1                              100.00
                                 direction          1           1                              100.00
                               direction_reg        1           1                              100.00
                                  invalid           1           1                              100.00
                              invalid_opcode        1           1                              100.00
                              invalid_red_op        1           1                              100.00
                                 leds[15-0]         1           1                              100.00
                                opcode[0-2]         1           1                              100.00
                              opcode_reg[2-0]       1           1                              100.00
                                   out[5-0]         1           1                              100.00
                                  red_op_A          1           1                              100.00
                                red_op_A_reg        1           1                              100.00
                                  red_op_B          1           1                              100.00
                                red_op_B_reg        1           1                              100.00
                                    rst             1           1                              100.00
                                 serial_in          1           1                              100.00
                               serial_in_reg        1           1                              100.00

Total Node Count     =         59
Toggled Node Count   =         59
Untoggled Node Count =          0

Toggle Coverage      =     100.00% (118 of 118 bins)
```

```
Total Coverage By Instance (filtered view): 100.00%
```