MARIAM MOHAMED MARZOUK

ASSIGNMENT_2

Question 1

1) Write a module to test dynamic array data type and its predefined methods. Run Questasim to make sure the display statements are working as expected.

- declare two dynamic arrays dyn_arr1, dyn_arr2 of type int
- initialize dyn_arr2 array elements with (9,1,8,3,4,4)
- allocate six elements in array dyn_arr1
- initialize array dyn_arr1 with index as its value
- display dyn_arr1 and its size
    - Expected output: (0,1,2,3,4,5), 6
- delete array dyn_arr1
- reverse, sort, reverse sort and shuffle the array dyn_arr2 and display dyn_arr2 after using each method
    - Expected output: (4,4,3,8,1,9), (1,3,4,4,8,9), (9,8,4,4,3,1), <shuffled_array>

I. RTL Design

```
module dynamic_array;
int dyn_arr1[];
int dyn_arr2[] = '{9,1,8,3,4,4};
    initial begin
        dyn_arr1 = new[6] ;
        foreach (dyn_arr1[i]) begin
            dyn_arr1[i] = i;
        end
        $display("dyn_arr1 = %0p, and its size = %0d", dyn_arr1, $size(dyn_arr1));
        dyn_arr1. delete();
        dyn_arr2. reverse();
        $display("dyn_arr2 = %0p After Reverse", dyn_arr2);
        dyn_arr2. sort();
        $display("dyn_arr2 = %0p After Sort", dyn_arr2);
        dyn_arr2.rsort();
        $display("dyn_arr2 = %0p After Reverse Sort", dyn_arr2);
        dyn_arr2. shuffle();
        $display("dyn_arr2 = %0p After Shuffle", dyn_arr2);
        $stop;

    end
endmodule
```

II. Print

```
# dyn_arr1 = 0 1 2 3 4 5, and its size = 6
# dyn_arr2 = 4 4 3 8 1 9 After Reverse
# dyn_arr2 = 1 3 4 4 8 9 After Sort
# dyn_arr2 = 9 8 4 4 3 1 After Reverse Sort
# dyn_arr2 = 8 1 4 3 9 4 After Shuffle
```

Question 2

Q2. Verify the functionality of the following counter:

- **Parameters:**
  1. WIDTH: width of the data_load and count_out ports (Valid values: 4, 6, 8, default: 4)
- **Inputs**:
  1. clk
  2. rst_n (active low sync rst)
  3. load_n (active low load)
  4. up_down (When the input is high then increment counter, else decrement the counter)
  5. ce (enable signal to increment or decrement the counter depending on the up_down) 6. data_load (load data to count_out output when the load signal is asserted)
- **Outputs**:
  1. count_out (counter output)
  2. max_count (When the counter reaches the maximum value, this signal is high, else low)
  3. zero (When the counter reaches the minimum value, this signal is high, else low)

Requirements:

1. Create a verification plan document based on your verification plan items to support your verification planning, an example of the document can be found in the link here. Please copy this document to have your own version
2. Create a package that has a class with the following constraints
   a. Constraint the reset to be deactivated most of the time
   b. Constraint the load signal to be active 70% of the time
   c. Constraint the enable signal to be active 70% of the time
3. Create a testbench that randomize the data in a repeat block or for loop using the class object to use the above constraints. Make the testbench self-checking.

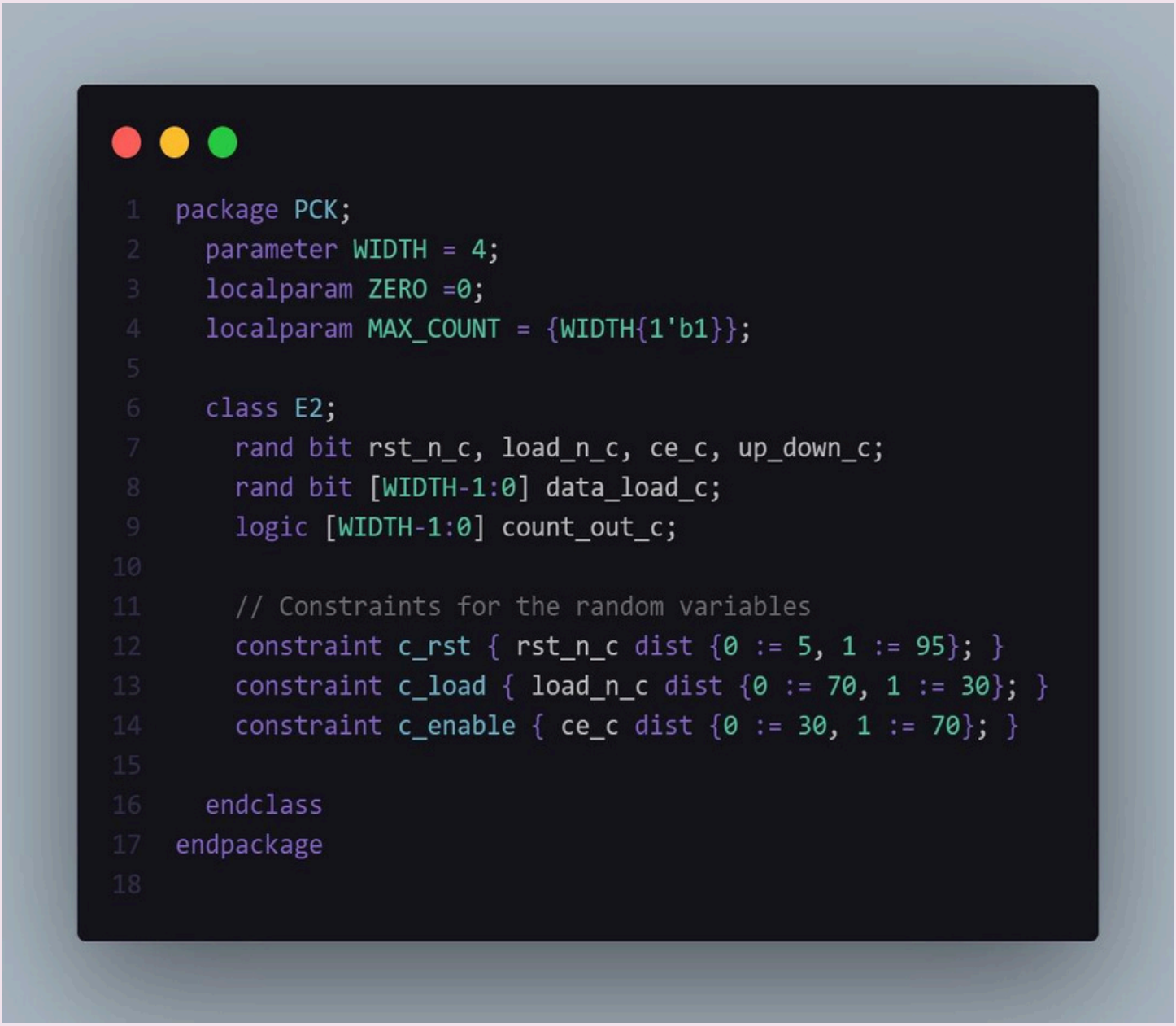
I. Verification plan

Label	Description	Stimulus Generation	Functionality Coverage	Functionality Check
Initialization	Set up the testbench environment and initialize signals.	Initialize rst_n and other variables. rst_n should be set to 1 (active-low reset) initially.	Ensure that the testbench initializes all signals properly before running tests.	Check initialization of all signals and clock.
Clock Generation	Generate a clock signal with a period of 50 time units.	Toggle clk every 25 time units (clock period = 50 time units).	Verify that the clock signal is stable and running at correct toggling rate.	Check the waveform or simulation log for correct clock toggling.
Randomization	Randomize the test object and apply values to DUT inputs.	Randomly assign values to rst_n, ce, up_down, load, data_load, and data_load inputs.	Ensure that all possible input conditions are covered.	Apply randomized values and check DUT output against expected results.
Reset Operation	When rst_n is low (reset condition), ensure the counter resets to 0.	Set rst_n to 0 and verify that count_out is reset to 0.	Verify that the counter resets to 0 when rst_n is asserted.	Compare count_out to 0 when rst_n is low.
Load Operation	When load is low (load operation), the counter should match data_load.	Set load to 0 and data_load to a random value, then check if count_out matches data_load.	Verify that count_out matches the value of data_load when load is low.	Compare count_out to data_load when load is low.
Up/Down Counting	Verify the counting operation based on up_down signal.	Set ce to 1 and up_down to a random value (0 or 1). When up_down is 1, count_out should increment; when 0, it should decrement.	Ensure the counter increments or decrements correctly based on the up_down signal.	Check count_out for correct increment/decrement behavior.
Wrap-Around Behavior	Ensure correct wrap-around behavior for the counter.	Test counting beyond the maximum value or below zero to ensure wrap-around behavior.	Verify wrap-around functionality at maximum and minimum values.	Check if count_out correctly wraps around from maximum to 0 or 0 to maximum.
Final Results	Display total counts of correct and incorrect results after all tests.	Record correct_count and error_count after the test sequence.	Ensure overall test results indicate success or failure.	Display final counts of correct and error results.
Self Check Task	Compare the actual count_out with the expected count_out_exp.	Use the self_check task to display and compare the DUT count_out with the expected value.	Ensure all DUT outputs match expected results for correctness.	Display and check count_out against count_out_exp for correctness.

II. RTL Design

```
////////////////////////////////////////////////////////////////////////////////
// Author: Kareem Waseem
// Course: Digital Verification using SV & UVM
//
// Description: Counter 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
```

III. Package



```
package PCK;
  parameter WIDTH = 4;
  localparam ZERO =0;
  localparam MAX_COUNT = {WIDTH{1'b1}};

  class E2;
    rand bit rst_n_c, load_n_c, ce_c, up_down_c;
    rand bit [WIDTH-1:0] data_load_c;
    logic [WIDTH-1:0] count_out_c;

    // Constraints for the random variables
    constraint c_rst { rst_n_c dist {0 := 5, 1 := 95}; }
    constraint c_load { load_n_c dist {0 := 70, 1 := 30}; }
    constraint c_enable { ce_c dist {0 := 30, 1 := 70}; }

  endclass
endpackage

```

IV. Test bench

```
import PCK::*;
module counter_tb;
    // Parameter
    parameter WIDTH = 4;
    // Input and output declaration
    logic clk, rst_n, load_n, up_down, ce;
    logic [WIDTH-1:0] data_load;
    logic [WIDTH-1:0] count_out;
    logic max_count, zero;
    logic [WIDTH-1:0] count_out_exp; // Expected value for output count
    // Counters declaration
    int correct_count, error_count;
    // DUT Instantiation
    counter #(.WIDTH(WIDTH)) DUT (
        .clk(clk),
        .rst_n(rst_n),
        .load_n(load_n),
        .up_down(up_down),
        .ce(ce),
        .data_load(data_load),
        .count_out(count_out),
        .max_count(max_count),
        .zero(zero)
    );
    // Clock generation
    initial begin
        clk = 0;
        forever #25 clk = ~clk; // Clock period is 50 time units
    end
    // Define the randomization object
    E2 object1;
    initial begin
        // Initialize variables
        object1 = new();
        correct_count = 0;
        error_count = 0;
        count_out_exp = 0;
```

```
        // Run the test for a number of iterations
        repeat (1000) begin
            // Randomize the test object
            assert (object1.randomize());
            // Apply randomized values
            rst_n = object1.rst_n_c;
            load_n = object1.load_n_c;
            up_down = object1.up_down_c;
            ce = object1.ce_c;
            data_load = object1.data_load_c;
            // Calculate the expected value
            if (!rst_n) begin
                count_out_exp = 0;
            end else if (!load_n) begin
                count_out_exp = data_load;
            end else if (ce) begin
                // Update count_out_exp based on direction
                if (up_down) begin
                    if (count_out_exp == {WIDTH{1'b1}})
                        count_out_exp = 0; // Wrap around
                    else
                        count_out_exp = count_out_exp + 1;
                end else begin
                    if (count_out_exp == 0)
                        count_out_exp = {WIDTH{1'b1}}; // Wrap around
                    else
                        count_out_exp = count_out_exp - 1;
                end
            end
            #50; // Adjusted delay for stabilization
            // Check the output of the DUT
            self_check();
        end
        // Final results
        $display("Total Correct Count =
, Total $stop$ =
", correct_count, error_count);
    // Task to check the DUT output
    task self_check();
        // Display
        $display("DUT Output: count_out =
, Expected Output: count_out_exp) begin
", count_out$display(Error);Expected count_out =
, but got count_out++;
", count_out_exp,begin_out);
            correct_count++;
        end
    endtask
endmodule
```

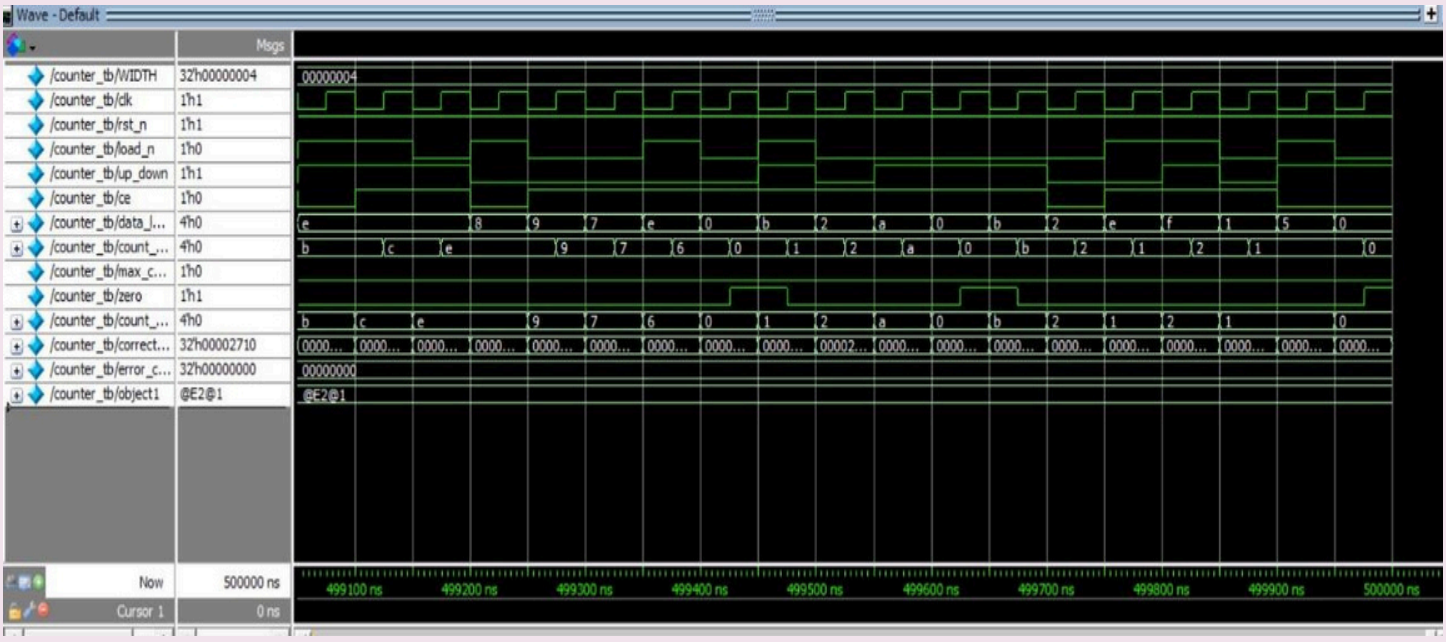
V. Counters

```
# Total Correct Count = 10000, Total Errors = 0
```

VI. Do File

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

VII. Waveform



VIII. Coverage Report

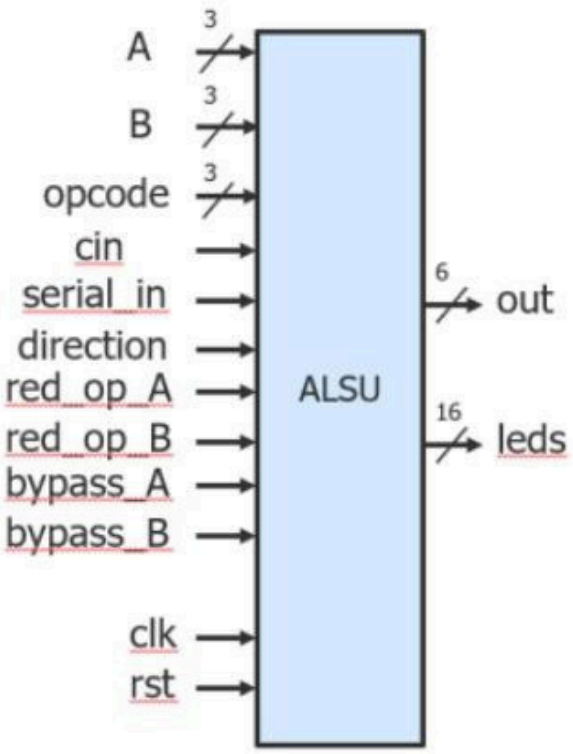
```
Coverage Report by instance with details

================================================================================
=== Instance: /\counter_tb#DUT
=== Design Unit: work.counter
================================================================================
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 counter.v
------------------------------------IF Branch------------------------------------
    21                                     9993     Count coming in to IF
    21             1                        533
    23             1                       6600
    25             1                       1999     |
                                            861     All False Count
Branch totals: 4 hits of 4 branches = 100.00%

------------------------------------IF Branch------------------------------------
    26                                     1999     Count coming in to IF
    26             1                       1014
    28             1                        985
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    32                                     8640     Count coming in to IF
    32             1                        548
    32             2                       8092
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    33                                     8640     Count coming in to IF
    33             1                        941
    33             2                       7699
Branch totals: 2 hits of 2 branches = 100.00%


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

```
================================Condition Details================================

Condition Coverage for instance /\counter_tb#DUT  --

  File counter.v
----------------Focused Condition View-------------------
Line       32 Item    1  (count_out == {4{{1}}})
Condition totals: 1 of 1 input term covered = 100.00%

              Input Term   Covered  Reason for no coverage   Hint
             -----------  --------  ----------------------  --------------
  (count_out == {4{{1}}})         Y

     Rows:       Hits  FEC Target                Non-masking condition(s)
 ---------  ---------  --------------------      -------------------------
  Row   1:          1  (count_out == {4{{1}}})_0  -
  Row   2:          1  (count_out == {4{{1}}})_1  -

----------------Focused Condition View-------------------
Line       33 Item    1  (count_out == 0)
Condition totals: 1 of 1 input term covered = 100.00%

        Input Term   Covered  Reason for no coverage   Hint
       -----------  --------  ----------------------  --------------
  (count_out == 0)         Y

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
  Row   1:          1  (count_out == 0)_0    -
  Row   2:          1  (count_out == 0)_1    -


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 counter.v
    20             1                        9993
    22             1                         533
    24             1                        6600
    27             1                        1014
    29             1                         985
    32             1                        8641
    33             1                        8641
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

2) ALSU is a logic unit that can perform logical, arithmetic, and shift operations on input ports • Input ports A and B have various operations that can take place depending on the value of the opcode.

- Each input bit except for the clk and rst will be sampled at the rising edge before any processing so a D-FF is expected for each input bit at the design entry.
- The output of the ALSU is registered and is available at the rising edge of the clock.



Inputs

Each input bit except for the clk and rst will have a DFF in front of its port. Any processing will take place from the DFF output.

Invalid cases

1. Opcode bits are set to 110 or 111
2. red_op_A or red_op_B are set to high and the opcode is not OR or XOR operation

Output when invalid cases occurs

1. leds are blinking
2. out bits are set to low

Opcode	Operation
000	OR
001	XOR
010	ADD
011	MULT
100	SHIFT (Shift output by 1 bit)
101	ROTATE (Rotate output by 1 bit)
110	Invalid opcode
111	Invalid opcode

Input	Width	Description
clk	1	Input clock
rst	1	Active high asynchronous reset
A	3	Input port A
B	3	Input port B
cin	1	Carry in bit, only valid to be used if the parameter FULL_ADDER is "ON"
serial_in	1	Serial in bit, used in shift operations only
red_op_A	1	When set to high, this indicates that reduction operation would be executed on A rather than bitwise operations on A and B when the opcode indicates OR and XOR operations
red_op_B	1	When set to high, this indicates that reduction operation would be executed on B rather than bitwise operations on A and B when the opcode indicates OR and XOR operations
opcode	3	Opcode has a separate table to describe the different operations executed
bypass_A	1	When set to high, this indicates that port A will be registered to the output ignoring the opcode operation
bypass_B	1	When set to high, this indicates that port B will be registered to the output ignoring the opcode operation
direction	1	The direction of the shift or rotation operation is left when this input is set to high; otherwise, it is right.

Outputs and parameters

Output	Width	Description
leds	16	When an invalid operation occurs, all bits blink (bits turn on and then off with each clock cycle). Blinking serves as a warning; otherwise, if a valid operation occurs, it is set to low.
out	6	Output of the ALSU

Parameter	Default value	Description
INPUT_PRIORITY	A	Priority is given to the port set by this parameter whenever there is a conflict. Conflicts can occur in two scenarios, red_op_A and red_op_B are both set to high or bypass_A and bypass_B are both set to high. Legal values for this parameter are A and B
FULL_ADDER	ON	When this parameter has value "ON" then cin input must be considered in the addition operation between A and B. Legal values for this parameter are ON and OFF

Opcodes & Handling invalid cases

Testbench:

- You are required to verify the functionality of the ALSU under the default configuration only.

Requirements:

1. Create a verification plan document to support your verification planning, an example of the document can be found in the link here. Please copy this document to have your own version and fill the document with your verification requirements.
2. Add comments in your testbench and class with the labels taken from your verification plan document for easy tracking of the implementation of constraints.

Create a package that have a user defined enum opcode_e that takes the value of the opcode, you can name the invalid cases INVALID_6 and INVALID_7.

3. Create a class in the package to randomize the design inputs under the following constraints.
    1. Reset to be asserted with a low probability that you decide.
    2. Constraint for adder inputs (A, B) to take the values (MAXPOS, ZERO and MAXNEG) more often than the other values when the ALU is addition or multiplication.
    3. In case of ALU opcode OR or XOR and red_op_A is high, constraint the input A most of the time to have one bit high in its 3 bits while constraining the B to be low
    4. In case of ALU opcode OR or XOR and red_op_B is high, constraint the input B most of the time to have one bit high in its 3 bits while constraining the A to be low
    5. Invalid cases should occur less frequent than the valid cases
    6. bypass_A and bypass_B should be disabled most of the time
    7. Do not constraint the inputs A or B when the operation is shift or rotate

➢ In your testbench

- Loop to randomize the inputs applying the above constraints
- Check the code coverage report and modify the testbench as necessary to achieve 100% code coverage. Modify over the constraint blocks to reach 100% if necessery.

I. Verification Plan

Label	Description	Stimulus Generation	Functionality Coverage	Functionality Check
Initialization	Set up the testbench environment and initialize signals.	Initialize rst to 1, then to 0 at the negative edge of the clock.	Ensure the testbench initializes properly before running tests.	rst signal transitions and initialization check.
Opcode Randomization	Randomly select values for the opcode from 0 to 5.	opcode is assigned a random value from 0 to 5 during each iteration.	Ensure all opcodes are covered during the test.	Output checked against the golden model after applying random opcode values.
Red Operation A and B Randomization	Randomly select red_op_A and red_op_B values (0 or 1).	red_op_A and red_op_B are randomly set to 0 or 1 in each iteration.	Included in coverage to verify how different red_op_A and red_op_B values affect the output.	Output checked against the golden model.
Bypass Input Randomization	Randomly select values for bypass_A and bypass_B (0 or 1).	bypass_A and bypass_B are randomly assigned values of 0 or 1.	Ensure different bypass configurations are covered.	Output checked against the golden model.
Direction Randomization	Randomly select the direction signal (0 or 1).	direction is randomly set to 0 or 1.	Validate how the direction signal affects the ALU output.	Output checked against the golden model.
Serial Input Randomization	Randomly select the serial_in signal (0 or 1).	serial_in is randomly assigned values of 0 or 1.	Ensure coverage of all serial input conditions.	Output checked against the golden model.
Input Values Randomization	Randomly select values for A and B (0 to 3) and cin (0 or 1).	A and B are assigned random values between 0 and 3, cin is assigned 0 or 1.	Validate functional coverage of various input values.	Output checked against the golden model.
Golden Model Check	Compare output against the expected values from the golden model.	Apply randomized inputs and compare out and leds to out_expected and leds_expected.	Included in coverage to validate correctness of the DUT implementation.	Errors and correct counts are displayed based on comparison with expected outputs.

II. 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, 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;
output reg [15:0] leds;
output reg signed [5:0] out;

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, 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 (invalid)
           out <= 0;
      else 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 begin
          case (opcode)
```

```
    3'h0: begin
      if (red_op_A_reg && red_op_B_reg)
        out <= (INPUT_PRIORITY == "A") ? |A_reg : |B_reg;
      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;
      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")
                 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
```

III. Golden Model

```
module ALSU_GM(A, B, cin, serial_in, red_op_A, red_op_B, opcode, bypass_A, bypass_B, clk, rst, direction, leds_expected, out_expected);
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;
output reg [15: 0] leds_expected;
output reg signed [5: 0] out_expected;
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, A_reg,B_reg;
wire invalid_red_op, invalid_opcode, invalid;
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;
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
always @(posedge clk or posedge rst) begin
if(rst) begin
leds_expected <= 0;
end else begin
if (invalid)
leds_expected <= ~leds_expected;
else
leds_expected <= 0;
end
end
always @(posedge clk or posedge rst) begin
if(rst) begin
```

```
out_expected <= 0;
end
else begin
if (invalid)
out_expected <= 0;
else if (bypass_A_reg && bypass_B_reg)
out_expected <= A_reg ;
else if (bypass_A_reg)
out_expected <= A_reg ;
else if (bypass_B_reg)
out_expected <= B_reg ;
else begin
case (opcode)
3'h0:
if (red_op_A_reg && red_op_B_reg)
out_expected = |A_reg;
else if (red_op_A_reg)
out_expected <= |A_reg;
else if (red_op_B_reg)
out_expected <= |B_reg;
else
out_expected <= A_reg | B_reg;
3'h1:
if (red_op_A_reg && red_op_B_reg)
out_expected <= ^A_reg;
else if (red_op_A_reg)
out_expected <= ^A_reg;
else if (red_op_B_reg)
out_expected <= ^B_reg;
else
out_expected <= A_reg ^ B_reg;
3'h2: out_expected <= A_reg + B_reg + cin_reg ;
3'h3: out_expected <= A_reg * B_reg;
3'h4:
if (direction_reg)
out_expected <= {out_expected[4: 0], serial_in_reg};
else
out_expected <= {serial_in_reg, out_expected[5: 1]};
3'h5:
if (direction_reg)
out_expected <= {out_expected[4: 0], out_expected[5]};
else
out_expected <= {out_expected[0], out_expected[5: 1]};
default: out_expected <= 0 ;
endcase
end
end
end
endmodule
```

IV. Package

```
package pck;
typedef enum bit [2: 0] {OR, XOR, ADD, MULT, SHIFT,ROTATE,INVALID_6,INVALID_7} opcode_e ;
localparam MAXPOS = 3 ;
localparam MAXNEG = -4 ;
localparam ZERO = 0 ;
class Z;
rand bit rst_c , red_op_A_c , red_op_B_c , cin_c , serial_in_c , direction_c , bypass_A_c , bypass_B_c ;
rand bit signed [2: 0] A_c , B_c ;
rand opcode_e opcode_c;
constraint c_rst {rst_c dist {0:/90 , 1:/10};}
constraint c_input_A {
  if (opcode_c == ADD || opcode_c == MULT) {
    A_c dist {
      MAXPOS / 25,
      MAXNEG / 25,
      ZERO / 25,
      {3'b001, 3'b010, 3'b101, 3'b110, 3'b111} / 25
    };} else if ((opcode_c == OR || opcode_c == XOR) && red_op_A_c == 1) {B_c == 0;
    A_c dist {
      {3'b001, 3'b010, 3'b100} := 80,
      {3'b000, 3'b011, 3'b101, 3'b110, 3'b111} := 20
    };} else {
    A_c inside {[MAXNEG: MAXPOS]};}}
constraint c_input_B {
  if (opcode_c == ADD || opcode_c == MULT) {
    B_c dist {
      MAXPOS / 25,
      MAXNEG / 25,
      ZERO / 25,
      {3'b001, 3'b010, 3'b101, 3'b110, 3'b111} / 25};
  } else if ((opcode_c == OR || opcode_c == XOR) && red_op_A_c == 1) {
    A_c == 0;
    B_c dist {
      {3'b001, 3'b010, 3'b100} := 80,
      {3'b000, 3'b011, 3'b101, 3'b110, 3'b111} := 20 };}
      else {B_c inside {[MAXNEG: MAXPOS]}; }}
constraint c_opcode { opcode_c dist { [0: 5]:/90 ,[6: 7]:/10 }; };
constraint c_bypass_signals {bypass_A_c dist { 0:/90 , 1:/10 };bypass_B_c dist { 0:/90 , 1:/10 };};
 // Additional constraints to ensure all branches are covered
  constraint all_branches {
    // Test all combinations for bypass signals and red_op signals
    if (opcode_c == ADD || opcode_c == MULT) {
      red_op_A_c dist {0:/50, 1:/50};
      red_op_B_c dist {0:/50, 1:/50};}
    // For OR and XOR operations
    if (opcode_c == OR || opcode_c == XOR) {
      // Test if red_op_A_c is true
      if (red_op_A_c == 1) {
        A_c dist { {3'b001, 3'b010, 3'b100} := 80, {3'b000, 3'b011, 3'b101, 3'b110, 3'b111} := 20 };B_c == 0;}
      // Test if red_op_B_c is true
      if (red_op_B_c == 1) {B_c dist { {3'b001, 3'b010, 3'b100} := 80, {3'b000, 3'b011, 3'b101, 3'b110, 3'b111} := 20 }; A_c == 0;}}}
endclass
endpackage

```

V. Test Bench

```
import pck::*;
module ALSU_TB;
//PARAMETER
  parameter INPUT_PRIORITY = "A";
  parameter FULL_ADDER = "ON";
// inputs/outputs declaration
   logic clk, rst, cin, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
   logic [2: 0] opcode;
   logic signed [2: 0] A,B;
   logic [15: 0] leds;
   logic signed [5: 0] out;
   logic signed [5: 0] out_expected;
   logic [15: 0] leds_expected;
//COUNTER
 integer error_count;
 integer correct_count;
//CLOCK GENERATION
 initial begin
    clk=0;
    forever
    #25 clk=~clk;
 end
Z obj; // create object from the class
ALSU #(.INPUT_PRIORITY(INPUT_PRIORITY), . FULL_ADDER(FULL_ADDER)) DUT_Design(.*) ; //INSTANTIATION
ALSU_GM DUTG(.*) ;
// Main program block
initial begin
obj = new();
error_count = 0 ;
correct_count = 0 ;
// Initialize inputs
    rst = 1;
    @(negedge clk);
    rst = 0;
repeat(10000) begin
        // Randomly select values for inputs
        opcode = $random % 6; // Randomly select an opcode from 0 to 5
        red_op_A = $random % 2; // Randomly select red_op_A (0 or 1)
        red_op_B = $random % 2; // Randomly select red_op_B (0 or 1)
        bypass_A = $random % 2; // Randomly select bypass_A (0 or 1)
        bypass_B = $random % 2; // Randomly select bypass_B (0 or 1)
        direction = $random % 2; // Randomly select direction (0 or 1)
        serial_in = $random % 2; // Randomly select serial_in (0 or 1)
        A = $random % 4; // Random value for A (0 to 3)
        B = $random % 4; // Random value for B (0 to 3)
        cin = $random % 2; // Random value for cin (0 or 1)
```

```
        // Apply values and check outputs
        @(negedge clk);
        assert(obj.randomize());
        rst = obj.rst_c;
        cin = obj.cin_c;
        red_op_A = obj.red_op_A_c;
        red_op_B = obj.red_op_B_c;
        bypass_A = obj.bypass_A_c;
        bypass_B = obj.bypass_B_c;
        direction = obj.direction_c;
        serial_in = obj.serial_in_c;
        opcode = obj.opcode_c;
        A = obj.A_c;
        B = obj.B_c;
        golden_model();
    end
    // Display results
    $display("Correct count =
, Error count =
",correct_count, error_count);
//task golden
task golden_model();
@(negedge clk);
if(out !== out_expected && leds !== leds_expected) begin
$display("incorrect result!!");
error_count ++;
end
else
correct_count ++;
endtask
endmodule
```

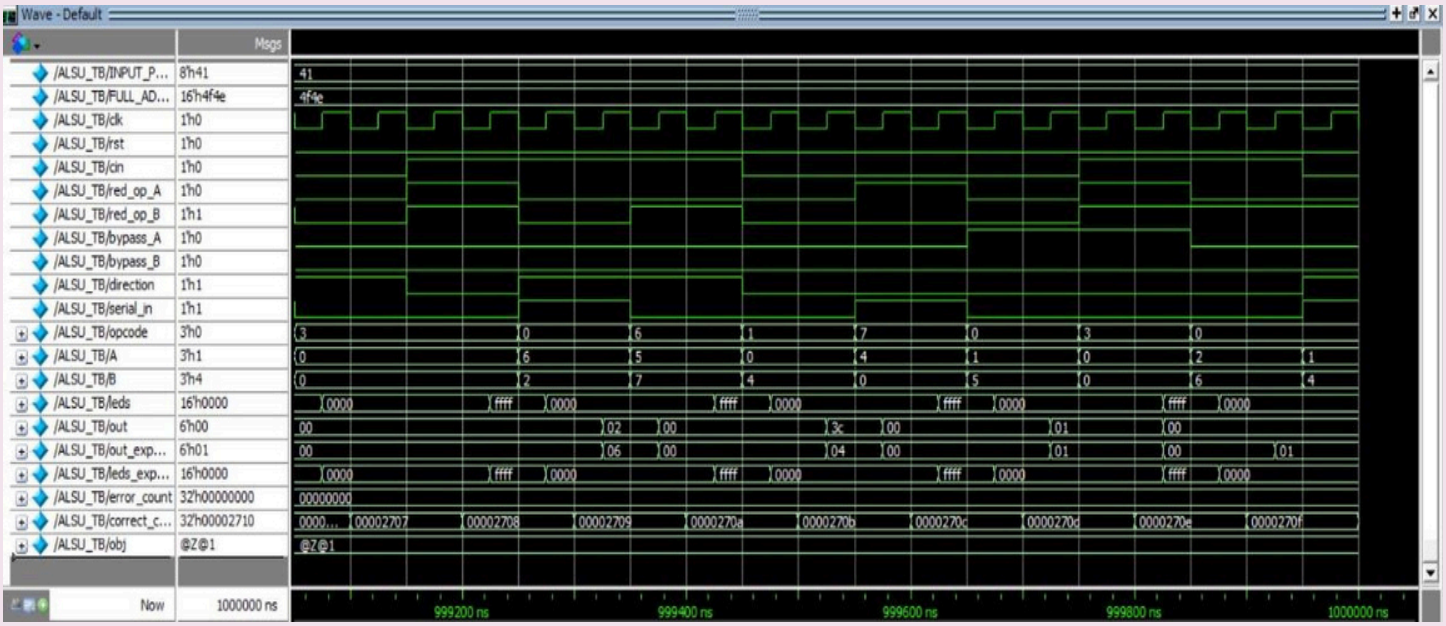
VI. Counters

```
correct counter = 10000 , error counter = 0
```

VII. Do File

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

VIII. Waveform



IX. Coverage Report

```
Coverage Report by instance with details

================================================================================
=== Instance: /\ALSU_TB#DUT_Design
=== Design Unit: work.ALSU
================================================================================
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        32        32         0   100.00%

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

Branch Coverage for instance /\ALSU_TB#DUT_Design

    Line          Item                      Count     Source
    ----          ----                      -----     ------
  File ALSU.v
-----------------------------------IF Branch-------------------------------------
    22                                      20955     Count coming in to IF
    22              1                        3077
    33              1                       17878
Branch totals: 2 hits of 2 branches = 100.00%

-----------------------------------IF Branch-------------------------------------
    49                                      20955     Count coming in to IF
    49              1                        3077
    51              1                       17878
Branch totals: 2 hits of 2 branches = 100.00%

-----------------------------------IF Branch-------------------------------------
    52                                      17878     Count coming in to IF
    52              1                        9705
    54              1                        8173
Branch totals: 2 hits of 2 branches = 100.00%

-----------------------------------IF Branch-------------------------------------
    61                                      20802     Count coming in to IF
    61              1                        2926
    64              1                       17876
Branch totals: 2 hits of 2 branches = 100.00%

-----------------------------------IF Branch-------------------------------------
    65                                      17876     Count coming in to IF
    65              1                        9705
    67              1                         804
    69              1                        1179
    71              1                        1143
    73              1                        5045
Branch totals: 5 hits of 5 branches = 100.00%

----------------------------------CASE Branch------------------------------------
    74                                       5045     Count coming in to CASE
    75              1                         822
    85              1                         542
    95              1                         521
    100             1                         820
    101             1                         803
    107             1                         820
    113             1                         717
Branch totals: 7 hits of 7 branches = 100.00%

-----------------------------------IF Branch-------------------------------------
    76                                       822     Count coming in to IF
    76              1                         18
    78              1                         13
    80              1                         20
    82              1                        771
Branch totals: 4 hits of 4 branches = 100.00%

-----------------------------------IF Branch-------------------------------------
    86                                       542     Count coming in to IF
    86              1                         16
    88              1                         16
    90              1                         20
    92              1                        490
Branch totals: 4 hits of 4 branches = 100.00%
```

```
------------------------------------IF Branch------------------------------------
    102                                     803     Count coming in to IF
    102             1                       324
    104             1                       479
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    108                                     820     Count coming in to IF
    108             1                       342
    110             1                       478
Branch totals: 2 hits of 2 branches = 100.00%


Condition Coverage:
    Enabled Coverage              Bins   Covered    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Conditions                       6         6         0   100.00%

================================Condition Details================================

Condition Coverage for instance /\ALSU_TB#DUT_Design  --

  File ALSU.v
----------------Focused Condition View-------------------
Line       67 Item    1  (bypass_A_reg && bypass_B_reg)
Condition totals: 2 of 2 input terms covered = 100.00%

    Input Term   Covered  Reason for no coverage   Hint
   -----------  --------  ----------------------  --------------
  bypass_A_reg         Y
  bypass_B_reg         Y

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
  Row   1:          1  bypass_A_reg_0        -
  Row   2:          1  bypass_A_reg_1        bypass_B_reg
  Row   3:          1  bypass_B_reg_0        bypass_A_reg
  Row   4:          1  bypass_B_reg_1        bypass_A_reg

----------------Focused Condition View-------------------
Line       76 Item    1  (red_op_A_reg && red_op_B_reg)
Condition totals: 2 of 2 input terms covered = 100.00%

    Input Term   Covered  Reason for no coverage   Hint
   -----------  --------  ----------------------  --------------
  red_op_A_reg         Y
  red_op_B_reg         Y

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
  Row   1:          1  red_op_A_reg_0        -
  Row   2:          1  red_op_A_reg_1        red_op_B_reg
  Row   3:          1  red_op_B_reg_0        red_op_A_reg
  Row   4:          1  red_op_B_reg_1        red_op_A_reg

----------------Focused Condition View-------------------
Line       86 Item    1  (red_op_A_reg && red_op_B_reg)
Condition totals: 2 of 2 input terms covered = 100.00%

    Input Term   Covered  Reason for no coverage   Hint
   -----------  --------  ----------------------  --------------
  red_op_A_reg         Y
  red_op_B_reg         Y

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
  Row   1:          1  red_op_A_reg_0        -
  Row   2:          1  red_op_A_reg_1        red_op_B_reg
  Row   3:          1  red_op_B_reg_0        red_op_A_reg
  Row   4:          1  red_op_B_reg_1        red_op_A_reg


Expression Coverage:
    Enabled Coverage              Bins   Covered    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Expressions                      8         8         0   100.00%
```

```
================================Expression Details================================

Expression Coverage for instance /\ALSU_TB#DUT_Design  --

  File ALSU.v
----------------Focused Expression View-----------------
Line       16 Item    1  ((red_op_A_reg | red_op_B_reg) & (opcode_reg[1] | opcode_reg[2]))
Expression totals: 4 of 4 input terms covered = 100.00%

     Input Term   Covered  Reason for no coverage   Hint
    -----------  --------  ----------------------  --------------
   red_op_A_reg         Y
   red_op_B_reg         Y
  opcode_reg[1]         Y
  opcode_reg[2]         Y

     Rows:       Hits  FEC Target           Non-masking condition(s)
 ---------  ---------  -------------------  -------------------------
  Row   1:          1  red_op_A_reg_0       ((opcode_reg[1] | opcode_reg[2]) && ~red_op_B_reg)
  Row   2:          1  red_op_A_reg_1       ((opcode_reg[1] | opcode_reg[2]) && ~red_op_B_reg)
  Row   3:          1  red_op_B_reg_0       ((opcode_reg[1] | opcode_reg[2]) && ~red_op_A_reg)
  Row   4:          1  red_op_B_reg_1       ((opcode_reg[1] | opcode_reg[2]) && ~red_op_A_reg)
  Row   5:          1  opcode_reg[1]_0      ((red_op_A_reg | red_op_B_reg) && ~opcode_reg[2])
  Row   6:          1  opcode_reg[1]_1      ((red_op_A_reg | red_op_B_reg) && ~opcode_reg[2])
  Row   7:          1  opcode_reg[2]_0      ((red_op_A_reg | red_op_B_reg) && ~opcode_reg[1])
  Row   8:          1  opcode_reg[2]_1      ((red_op_A_reg | red_op_B_reg) && ~opcode_reg[1])

----------------Focused Expression View-----------------
Line       17 Item    1  (opcode_reg[1] & opcode_reg[2])
Expression totals: 2 of 2 input terms covered = 100.00%

     Input Term   Covered  Reason for no coverage   Hint
    -----------  --------  ----------------------  --------------
  opcode_reg[1]         Y
  opcode_reg[2]         Y

     Rows:       Hits  FEC Target           Non-masking condition(s)
 ---------  ---------  -------------------  -------------------------
  Row   1:          1  opcode_reg[1]_0      opcode_reg[2]
  Row   2:          1  opcode_reg[1]_1      opcode_reg[2]
  Row   3:          1  opcode_reg[2]_0      opcode_reg[1]
  Row   4:          1  opcode_reg[2]_1      opcode_reg[1]

----------------Focused Expression View-----------------
Line       18 Item    1  (invalid_red_op | invalid_opcode)
Expression totals: 2 of 2 input terms covered = 100.00%

      Input Term   Covered  Reason for no coverage   Hint
     -----------  --------  ----------------------  --------------
  invalid_red_op         Y
  invalid_opcode         Y

     Rows:       Hits  FEC Target           Non-masking condition(s)
 ---------  ---------  -------------------  -------------------------
  Row   1:          1  invalid_red_op_0     ~invalid_opcode
  Row   2:          1  invalid_red_op_1     ~invalid_opcode
  Row   3:          1  invalid_opcode_0     ~invalid_red_op
  Row   4:          1  invalid_opcode_1     ~invalid_red_op


Statement Coverage:
    Enabled Coverage             Bins      Hits    Misses  Coverage
    ----------------             ----      ----    ------  --------
    Statements                     49        49         0   100.00%

================================Statement Details================================
```

```
================================Statement Details================================

Statement Coverage for instance /\ALSU_TB#DUT_Design  --

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File ALSU.v
    16              1                      18117
    17              1                      16234
    18              1                      11528
    21              1                      20955
    23              1                       3077
    24              1                       3077
    25              1                       3077
    26              1                       3077
    27              1                       3077
    28              1                       3077
    29              1                       3077
    30              1                       3077
    31              1                       3077
    32              1                       3077
    34              1                      17878
    35              1                      17878
    36              1                      17878
    37              1                      17878
    38              1                      17878
    39              1                      17878
    40              1                      17878
    41              1                      17878
    42              1                      17878
    43              1                      17878
    48              1                      20955
    50              1                       3077
    53              1                       9705
    55              1                       8173
    60              1                      20802
    62              1                       2926
    66              1                       9705
    68              1                        804
    70              1                       1179
    72              1                       1143
    77              1                         18
    79              1                         13
    81              1                         20
    83              1                        771
    87              1                         16
    89              1                         16
    91              1                         20
    93              1                        490
    97              1                        521
    100             1                        820
    103             1                        324
    105             1                        479
    109             1                        342
    111             1                        478
    113             1                        717
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                        118       118         0   100.00%
```

```
================================Toggle Details================================

Toggle Coverage for instance /\ALSU_TB#DUT_Design  --

                                          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          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%
```