ASSIGMENT 3

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
```

```
# number of correct =501 ,error=0
# ** Note: $finish    : counter_tb.sv(41)
#    Time: 5005 ns  Iteration: 1  Instance: /counter_tb
```

Tb

```
1   import all_item::*;
2
3   int error=0,correct =0;
4
5   module counter_tb();
6
7   parameter WIDTH = 4;
8
9   logic clk=1,rst_n,load_n,up_down,ce;
10  logic  [WIDTH-1:0] data_load;
11  logic  [WIDTH-1:0] count_out;
12  logic max_count,zero;
13  bit[3:0] last_value=0;
14
15  counter #(.WIDTH(WIDTH)) co (.*);
16
17  initial begin
18  forever begin#5 clk=!clk;
19  tr.clk=clk;
20  end
21  end
22
23  transaction tr=new();
24
```

```
initial begin
    tr.rst_n=0; rst_n=tr.rst_n;
    check_result(tr);
    repeat(500)begin
        assert (tr.randomize() );
        rst_n=tr.rst_n;
        load_n=tr.load_n;
        up_down=tr.up_down;
        ce=tr.ce;
        data_load=tr.data_load ;
        tr.count_out=count_out;
        check_result(tr);


    end
    $display("number of correct =%0d ,error=%0d",correct,error);
 $finish;
 end

task check_result(input transaction trans);
    //check reset
    golden_model(trans);
    @(negedge clk);
    if(trans.count_out!=count_out || trans.zero!=zero || trans.max_count!=max_count)begin
        $display("@%0t there is error count=%0d , zero=%0d ,max_count=%0d",$time,trans.count_out,trans.zero,trans.max_count);
        error++;
    end
    else
        correct++;

 endtask
```

```
task golden_model(input transaction tr_task);

if(!tr_task.rst_n)
tr_task.count_out=0;
else if(!load_n)
tr_task.count_out=tr_task.data_load;
else if(tr_task.ce)begin
    if(tr_task.up_down)
        tr_task.count_out=tr_task.count_out+1;
    else if(!tr_task.up_down)
        tr_task.count_out=tr_task.count_out-1;

end

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


endtask

endmodule
```

Package

```
package all_item;
    parameter WIDTH=4;
    class transaction;

    rand bit rst_n,load_n,up_down,ce;
    rand bit [WIDTH-1:0] data_load;
    bit clk;
    logic [WIDTH-1:0] count_out;
    bit zero,max_count;

    constraint x {
        rst_n dist {0:=5,1:=95};
        load_n dist {0:=70,1:=30};
        ce dist {1:=70,0:=30};
    }

    covergroup covgup @(posedge clk);
        coverpoint data_load iff(!load_n);
        CO1:coverpoint count_out iff(rst_n && ce && up_down){
            bins all_values_up[]={[0:15]};
            bins trans_zero=(15=>0);
        }
        CO2:coverpoint count_out iff(rst_n && ce && !up_down){
            bins all_values_down[]={[0:15]};
            bins trans_max=(0=>15);
        }
    endgroup

    function new();
        covgup=new();
    endfunction
```

```
    function new();
        covgup=new();
    endfunction

    function printing;
        $display("count_out=%0d  , max_count=%0d, zero=%0d",count_out,max_count,zero);
    endfunction

endclass
endpackage
```

Do file

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

vcover report counter_tb.ucdb -details -all -output coverage_report.txt
```

Verification plan

	Label	Description	Stimulus Generation	Functionality Check	functionality coverage
2	COUNTER_1	When the reset is asserted, the output counter value should be low then deassert reset	Directed at the start of the simulation then it randomized with cosntraint to be of high 95 % from time	A checker in the testbench to make sure the output is correct by test function	-
3	COUNTER_2	when load_n is asserted to low -> the output count out should take same value of load data	Randomization with constrain that Load_n should be low(active) for 70% of time,we can check this on time 75 ns	A checker in the testbench to make sure the output is correct by test function	
4	COUNTER_3	when load_n is asserted to low and load data is equal to 15 -> the output count out should take 15 and max count output should be 1	Randomization,we can check this on time 495 ns	A checker in the testbench to make sure the output is correct by test function	cover all values of load data when load_n asserted
5	COUNTER_4	when load_n is asserted to low and load data is equal to 0 -> the output count out should take 0 and zero output should be 1	Randomization,we can check this on time 535 ns	A checker in the testbench to make sure the output is correct by test function	
6	COUNTER_5	we assert load_n to high and we assert enaple input to high and count up -> the output count _out shoult increase every clock cycle	Randomization we can check this in most of simulation	A checker in the testbench to make sure the output is correct by test function in entire simulation	
7	COUNTER_6	we assert load_n to high and we assert enaple input to high and count down-> the output count _out shoult decrease every clock	Randomization under constrain that ec should be HIgh(active) for 70%	A checker in the testbench to make sure the output is correct by test function in entire simulation	we cover all values of count out in count up and transation from 15 to 0
7	COUNTER_6	we assert load_n to high and we assert enaple input to high and count down-> the output count _out shoult decrease every clock cycle	Randomization under constrain that ec should be HIgh(active) for 70% of time,	A checker in the testbench to make sure the output is correct by test function in entire simulation	we cover all values of count out in count up and transation from 15 to 0
8	COUNTER_7	we check on state when counter reach 15 through counting and check output max count if =1	Randomization	A checker in the testbench to make sure the output is correct by test function in entire simulation	
9	COUNTER_8	we check on state when counter reach 0 through counting and check output Zero count if =1	Randomization,we can check this on time 790 ns	A checker in the testbench to make sure the output is correct by test function in entire simulation	we cover all values of count out in count down and transation from 0 to 15
10	COUNTER_9	we check when reset is high and load_n is high and enaple is low the output shouldnt change and keep previous value	Randomization through simmulation	A checker in the testbench to make sure the output is correct by test function in entire simulation	

Code Coverage report

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

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

Statement Coverage for file counter.v --

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

```
================================Branch Details================================

Branch Coverage for file counter.v --

------------------------------------IF Branch------------------------------------
    22                                    500     Count coming in to IF
    22              1                      21         if (!rst_n)
    24              1                     353         else if (!load_n)
    26              1                      82         else if (ce)
                                           44     All False Count
Branch totals: 4 hits of 4 branches = 100.0%

------------------------------------IF Branch------------------------------------
    27                                     82     Count coming in to IF
    27              1                      43             if (up_down)
    29              1                      39             else
Branch totals: 2 hits of 2 branches = 100.0%

------------------------------------IF Branch------------------------------------
    33                                    433     Count coming in to IF
    33              1                      21     assign max_count = (count_out == {WIDTH{1'b1}})? 1:0;
    33              2                     412     assign max_count = (count_out == {WIDTH{1'b1}})? 1:0;
Branch totals: 2 hits of 2 branches = 100.0%

------------------------------------IF Branch------------------------------------
    34                                    433     Count coming in to IF
    34              1                      45     assign zero = (count_out == 0)? 1:0;
    34              2                     388     assign zero = (count_out == 0)? 1:0;
Branch totals: 2 hits of 2 branches = 100.0%


Condition Coverage:
    Enabled Coverage            Active   Covered    Misses % Covered
    ----------------            ------      ----    ------ ---------
    FEC Condition Terms              0         0         0     100.0
Expression Coverage:
    Enabled Coverage            Active   Covered    Misses % Covered
```

```
Condition Coverage:
    Enabled Coverage            Active   Covered    Misses % Covered
    ----------------            ------      ----    ------ ---------
    FEC Condition Terms              0         0         0     100.0
Expression Coverage:
    Enabled Coverage            Active   Covered    Misses % Covered
    ----------------            ------      ----    ------ ---------
    FEC Expression Terms             0         0         0     100.0
FSM Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    FSMs                                                      100.0
        States                       0         0         0     100.0
        Transitions                  0         0         0     100.0
Toggle Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Toggle Bins                    30        30         0     100.0

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

Toggle Coverage for File counter.v --

       Line                                  Node      1H->0L      0L->1H  "Coverage"
--------------------------------------------------------------------------------------
         11                                   clk           1           1      100.00
         12                                 rst_n           1           1      100.00
         13                                load_n           1           1      100.00
         14                               up_down           1           1      100.00
         15                                    ce           1           1      100.00
         16                          data_load[3]           1           1      100.00
         16                          data_load[2]           1           1      100.00
         16                          data_load[1]           1           1      100.00
         16                          data_load[0]           1           1      100.00
         17                         count_out[3]           1           1      100.00
```

Function Coverage report

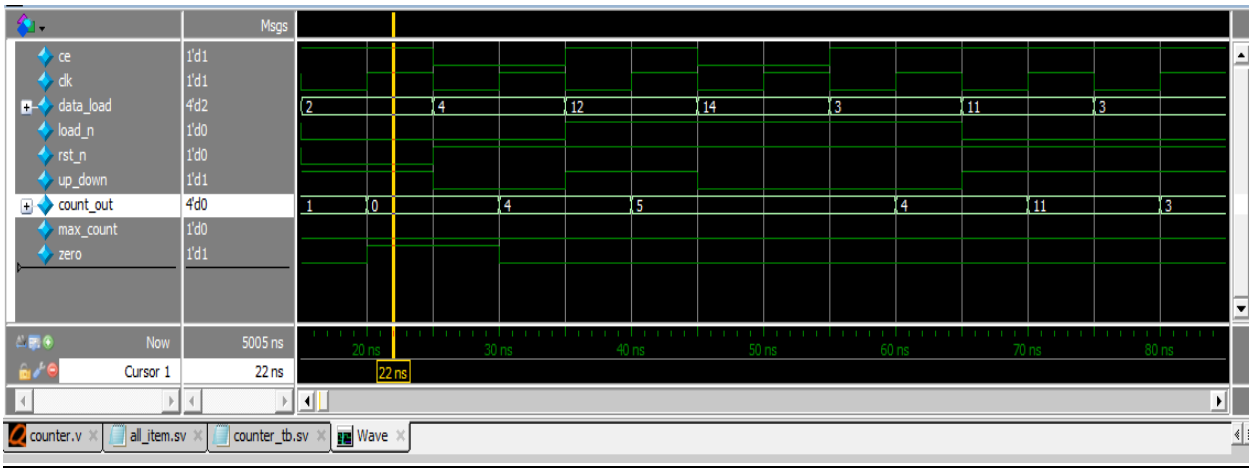
```
COVERGROUP COVERAGE:
----------------------------------------------------------------------------------------------
Covergroup                                              Metric       Goal    Status

----------------------------------------------------------------------------------------------
 TYPE /all_item/transaction/covgup                     100.0%        100    Covered
    covered/total bins:                                    50         50
    missing/total bins:                                     0         50
    % Hit:                                             100.0%        100
    Coverpoint covgup::data_load                       100.0%        100    Covered
        covered/total bins:                                16         16
        missing/total bins:                                 0         16
        % Hit:                                         100.0%        100
    Coverpoint covgup::CO1                             100.0%        100    Covered
        covered/total bins:                                17         17
        missing/total bins:                                 0         17
        % Hit:                                         100.0%        100
    Coverpoint covgup::CO2                             100.0%        100    Covered
        covered/total bins:                                17         17
        missing/total bins:                                 0         17
        % Hit:                                         100.0%        100
 CLASS transaction
 Covergroup instance \/all_item::transaction::covgup
                                                       100.0%        100    Covered
    covered/total bins:                                    50         50
    missing/total bins:                                     0         50
    % Hit:                                             100.0%        100
    Coverpoint data_load                               100.0%        100    Covered
        covered/total bins:                                16         16
        missing/total bins:                                 0         16
        % Hit:                                         100.0%        100
        bin auto[0]                                        30          1    Covered
        bin auto[1]                                        24          1    Covered
        bin auto[2]                                        24          1    Covered
        bin auto[3]                                        25          1    Covered
```

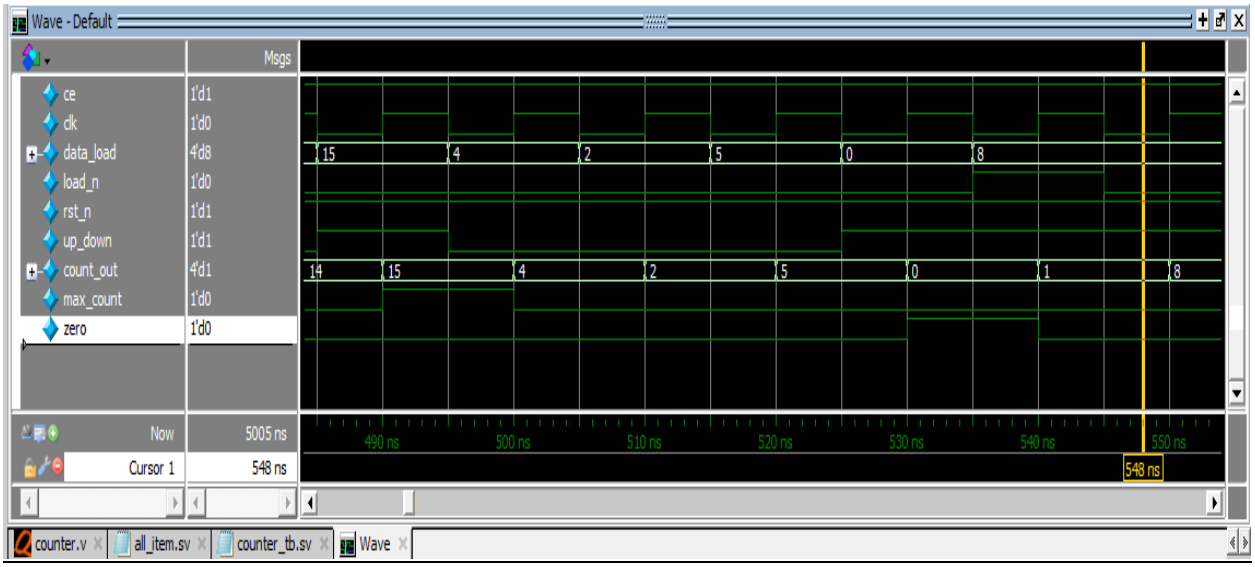
```
    % Hit:                                             100.0%        100
    Coverpoint data_load                               100.0%        100    Covered
        covered/total bins:                                16         16
        missing/total bins:                                 0         16
        % Hit:                                         100.0%        100
        bin auto[0]                                        30          1    Covered
        bin auto[1]                                        24          1    Covered
        bin auto[2]                                        24          1    Covered
        bin auto[3]                                        25          1    Covered
        bin auto[4]                                        23          1    Covered
        bin auto[5]                                        30          1    Covered
        bin auto[6]                                        29          1    Covered
        bin auto[7]                                        13          1    Covered
        bin auto[8]                                        22          1    Covered
        bin auto[9]                                        14          1    Covered
        bin auto[10]                                       27          1    Covered
        bin auto[11]                                       18          1    Covered
        bin auto[12]                                       25          1    Covered
        bin auto[13]                                       26          1    Covered
        bin auto[14]                                       21          1    Covered
        bin auto[15]                                       18          1    Covered
```

```
Coverpoint CO1                                  100.0%        100    Covered
    covered/total bins:                             17         17
    missing/total bins:                              0         17
    % Hit:                                       100.0%        100
    bin all_values_up[0]                            23          1    Covered
    bin all_values_up[1]                             9          1    Covered
    bin all_values_up[2]                             7          1    Covered
    bin all_values_up[3]                            11          1    Covered
    bin all_values_up[4]                            11          1    Covered
    bin all_values_up[5]                            15          1    Covered
    bin all_values_up[6]                            18          1    Covered
    bin all_values_up[7]                             8          1    Covered
    bin all_values_up[8]                             5          1    Covered
    bin all_values_up[9]                             9          1    Covered
    bin all_values_up[10]                            8          1    Covered
    bin all_values_up[11]                            7          1    Covered
    bin all_values_up[12]                           10          1    Covered
    bin all_values_up[13]                           11          1    Covered
    bin all_values_up[14]                           10          1    Covered
    bin all_values_up[15]                            7          1    Covered
    bin trans_zero                                   4          1    Covered
```

```
Coverpoint CO2                          100.0%        100    Covered
    covered/total bins:                     17         17
    missing/total bins:                      0         17
    % Hit:                               100.0%        100
    bin all_values_down[0]                  12          1    Covered
    bin all_values_down[1]                  16          1    Covered
    bin all_values_down[2]                  13          1    Covered
    bin all_values_down[3]                  10          1    Covered
    bin all_values_down[4]                   8          1    Covered
    bin all_values_down[5]                  11          1    Covered
    bin all_values_down[6]                  10          1    Covered
    bin all_values_down[7]                   6          1    Covered
    bin all_values_down[8]                   8          1    Covered
    bin all_values_down[9]                   7          1    Covered
    bin all_values_down[10]                 12          1    Covered
    bin all_values_down[11]                 10          1    Covered
    bin all_values_down[12]                 10          1    Covered
    bin all_values_down[13]                 10          1    Covered
    bin all_values_down[14]                  9          1    Covered
    bin all_values_down[15]                  9          1    Covered
    bin trans_max                            1          1    Covered

TOTAL COVERGROUP COVERAGE: 100.0%  COVERGROUP TYPES: 1

ASSERTION RESULTS:
```

Simulation




Wave - Default
Msgs
ce
clk
data_load
load_n
rst_n
up_down
count_out
max_count
zero
1'd1
1'd0
4'd8
1'd0
1'd1
1'd1
4'd1
1'd0
1'd0
15
4
2
5
0
8
14
15
4
2
5
0
1
8
Now
5005 ns
Cursor 1
548 ns
490 ns
500 ns
510 ns
520 ns
530 ns
540 ns
550 ns
548 ns
counter.v
all_item.sv
counter_tb.sv
Wave


Msgs
ce
clk
data_load
load_n
rst_n
up_down
count_out
max_count
zero
1'd0
1'd1
4'd3
1'd1
1'd1
1'd1
4'd5
1'd0
1'd0
15
0
Now
5005 ns

ALSU

Design

E: > study > kareem wassem > ASSIGMENTS > 3 > ALSU > ALSU.v

```
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;
reg signed [2:0] A_reg, B_reg; //change to signed
reg signed [5:0] out_next;
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;
```

```
  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
```

```
always @(posedge clk or posedge rst) begin

  if(rst) begin
    out <= 0;
  end
  else begin
    if (bypass_A_reg && bypass_B_reg)
      out <= (INPUT_PRIORITY == "A")? A_reg: B_reg;
    else if (bypass_A_reg)
      out <= A_reg;
    else if (bypass_B_reg)
      out <= B_reg;
    else if (invalid) // cahnge the priority of invalid bits after bypass_reg
        out <= 0;
    else begin
        case (opcode_reg)
          3'h0: begin //change Opcode to OR not AND
            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 // change opcode to XOR not OR
            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 //here we add condition to check full adder if ON or OFF
            if(FULL_ADDER == "ON")
              out <= A_reg + B_reg+cin_reg;
            else if(FULL_ADDER == "OFF")
              out <= A_reg + B_reg;

          end
          3'h3: out <= A_reg * B_reg;
          3'h4: begin

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

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

endmodule
```

Tb

```
import pack_alsu::*;

module ALSU_tb ();
parameter INPUT_PRIORITY = "B";
parameter FULL_ADDER = "ON";

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



int error =0,correct=0;
bit invalid_t,x1,x2;
bit signed [5:0] last_out=0;

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


always #10 clk=!clk;
transaction tr=new();
```

```
 initial begin
    tr.rst=1'b1;
    init(tr);
    check_result(tr);

    tr.constraint_mode(0);
    tr.x.constraint_mode(1);
    repeat(200) begin
        assert(tr.randomize());
        init(tr);
        check_result(tr);
        sampling(tr);
    end

    tr.constraint_mode(0);
    tr.y.constraint_mode(1);
    tr.rst=0;tr.bypass_A=0;tr.bypass_B=0;tr.red_op_A=0;tr.red_op_B=0;
    tr.rst.rand_mode(0);tr.bypass_A.rand_mode(0);tr.bypass_B.rand_mode(0);tr.red_op_A.rand_mode(0);
    tr.red_op_B.rand_mode(0);
    init(tr);

    for(int i=0;i<100;i++)begin
        assert(tr.randomize());
        cin=tr.cin;direction=tr.direction;serial_in=tr.serial_in;A=tr.A;B=tr.B;
        if('{OR,XOR,ADD,MULT,SHIFT,ROTATE} ==tr.arr)$display("@%0t the wanted sequence is %p",$time,tr.arr);
        foreach(tr.arr[j])begin

```

```
47      for(int i=0;i<100;i++)begin
48          assert(tr.randomize());
49          cin=tr.cin;direction=tr.direction;serial_in=tr.serial_in;A=tr.A;B=tr.B;
50          if('{OR,XOR,ADD,MULT,SHIFT,ROTATE} ==tr.arr)$display("@%0t the wanted sequence is %p",$time,tr.arr);
51          foreach(tr.arr[j])begin
52
53              tr.opcode=tr.arr[j];
54              opcode=tr.arr[j];
55              tr.out=out;
56              tr.leds=leds;
57              check_result(tr);
58              sampling(tr);
59          end
60      end
61      $display("number of correct =%0d ,error=%0d",correct,error);
62  $stop;
63
64  end
```

```
5   task check_result(input transaction  ch);
6
7       golden_model(ch);
8          repeat(2) @(negedge clk);
9           if(ch.out != out && ch.leds != leds)begin
0               $display("@%t there is error out=%0b ,leds=%0b " ,$time ,tr.out,tr.leds);error++;
1           end
2           else
3               correct++;
4
5       if(ch.opcode ==ROTATE || ch.opcode ==SHIFT)begin
6           golden_model(ch);
7       end
8
9   endtask
```

```
83  task golden_model(input transaction  tr);
84          x1=tr.opcode==INVALID6 || tr.opcode == INVALID7;
85          x2=(tr.red_op_A==1'b1 | tr.red_op_B==1'b1) && (tr.opcode!= OR && tr.opcode!=XOR);
86          invalid_t= x1 || x2;
87
88      if(invalid_t)
89          tr.leds= ~tr.leds;
90
91      if(tr.rst)begin //check on reset
92          tr.out=0;
93          tr.leds= 0;
94      end
95      else begin
96          if(tr.bypass_A && tr.bypass_B)begin
97              if (INPUT_PRIORITY== "A")
98                  tr.out = tr.A;
99              else if (INPUT_PRIORITY== "B")
100                 tr.out = tr.B;
101         end
102         // check on pypass operations
103         else if(bypass_A) //check on bypass
104                 tr.out = tr.A;
105         else if(bypass_B)
106                 tr.out = tr.B;
107         //check on invalid_t condition output
108         else if(invalid_t) begin
109                 tr.out=0;
110         end
111         else begin
```

```
112             //here we check on OP code
113             case (tr.opcode)
114             OR:begin// check on priority first
115                 if(tr.red_op_A && tr.red_op_B && INPUT_PRIORITY== "A")
116                     tr.out = tr.A;
117                 else if(tr.red_op_A && tr.red_op_B&& INPUT_PRIORITY== "B")
118                     tr.out = tr.B;
119                 else if(tr.red_op_A)
120                     tr.out = (|tr.A);
121                 else if(tr.red_op_B)
122                     tr.out = (|tr.B);
123                 else
124                     tr.out = (tr.A|tr.B);
125             end
126             XOR:begin
127                 if(tr.red_op_A && tr.red_op_B && INPUT_PRIORITY== "A")
128                     tr.out = tr.A;
129                 else if(tr.red_op_A && tr.red_op_B&& INPUT_PRIORITY== "B")
130                     tr.out = tr.B;
131                 else if(tr.red_op_A)
132                     tr.out = (^tr.A);
133                 else if(tr.red_op_B)
134                     tr.out = (^tr.B);
135                 else
136                     tr.out = (tr.A^tr.B);
137             end
138             ADD:begin
139                 if(FULL_ADDER == "ON")
140                     tr.out= tr.A+tr.B+tr.cin;
```

```
141                 else if(FULL_ADDER == "OFF")
142                     tr.out= tr.A+tr.B;
143               end
144               MULT:begin
145                   tr.out= tr.A*tr.B;
146               end
147               SHIFT:begin
148                   if(tr.direction)
149                       tr.out = {tr.out[4:0],tr.serial_in};
150                   else if(!tr.direction)
151                       tr.out = {tr.serial_in,tr.out[5:1]};
152               end
153               ROTATE:begin
154                   if(tr.direction)
155                       tr.out = {tr.out[4:0],tr.out[5]};
156                   else if(!tr.direction)
157                       tr.out = {tr.out[0],tr.out[5:1]};
158               end
159               endcase
160
161           end
162
163       end
164       last_out=tr.out;
165       endtask
166
```

```
163     end
164     last_out=tr.out;
165     endtask
166
167     function void init(transaction in);
168         opcode=tr.opcode;A=tr.A;B=tr.B;
169         rst=tr.rst;cin=tr.cin; red_op_A=tr.red_op_A; red_op_B=tr.red_op_B; bypass_A=tr.bypass_A;
170         bypass_B=tr.bypass_B; direction=tr.direction; serial_in=tr.serial_in ;
171         tr.out=out;tr.leds=leds;
172     endfunction
173
174     function sampling(transaction tr);
175         if(rst ||bypass_A ||bypass_B)begin
176             tr.cvr_gp.stop();
177         end
178         else begin
179             tr.cvr_gp.start();
180             tr.cvr_gp.sample();
181         end
182     endfunction
183 endmodule
```

Package

```
1   package pack_alsu;
2   typedef enum { OR=0,XOR,ADD,MULT,SHIFT,ROTATE,INVALID6,INVALID7 } Opcode_e;
3   typedef enum {MAXPOS=3,MAXNEG=-4,ZERO=0}corner_state_e;
4
5   class transaction;
6       rand bit  clk, rst, cin, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
7       rand Opcode_e opcode;
8       rand bit signed [2:0] A, B;
9       bit [2:0] ones_number={3'b001,3'b010,3'b100};
10      rand bit [2:0] found,notfound;
11      rand corner_state_e a_state;
12      rand bit [2:0] rem_numbers;
13      bit signed [5:0] out;
14      bit [15:0] leds;
15
16      rand Opcode_e arr[6];
17
18      constraint x {
19
20          rem_numbers!= MAXPOS||MAXNEG||ZERO;
21
22          rst dist {1:=5 , 0:=95};
23
24          found inside {ones_number};
25          !(notfound inside {ones_number});
26
27          if (opcode ==ADD || opcode== MULT){
28              A dist {a_state:=80,rem_numbers:=20};
29              B dist {a_state:=80,rem_numbers:=20};
30          }
```

```
        if (opcode ==OR || opcode== XOR ){
            if(red_op_A){
                A dist {found:=80,notfound:=20};
                B==3'b000;
            }
            else if (red_op_B){
                B dist {found:=80,notfound:=20};
                A==3'b000;
            }
        }


        opcode dist {[OR:ROTATE]:=80,[INVALID6:INVALID7]};

        bypass_A dist {0:=90,1:=10};
        bypass_B dist {0:=90,1:=10};
    }
    constraint y{
        unique{arr};
        foreach(arr[i])
            arr[i] inside {[OR:ROTATE]};
    }
```

```
    covergroup cvr_gp;
        CB1:coverpoint A{
            bins A_data_0={0};
            bins A_data_max={MAXPOS};
            bins A_data_min={MAXNEG};
            bins A_data_walkingones[] ={3'b001,3'b010,3'b100} iff (red_op_A);
            bins A_data_default=default;
        }
        CB2:coverpoint B{
            bins B_data_0={0};
            bins B_data_max={MAXPOS};
            bins B_data_min={MAXNEG};
            bins B_data_walkingones[] ={3'b001,3'b010,3'b100} iff (red_op_B);
            bins B_data_default=default;
        }
        CB3:coverpoint opcode{
            bins Bins_shift[]={SHIFT,ROTATE};
            bins Bins_arith[] ={ADD,MULT};
            bins Bins_bitwise[] ={OR,XOR};
            illegal_bins Bins_invalid ={INVALID6,INVALID7};
            bins Bins_trans=(OR=>XOR=>ADD=>MULT=>SHIFT=>ROTATE);
        }
    endgroup

    function new();
        cvr_gp=new();
    endfunction

endclass
endpackage
```

Do file

```
vlib work
vlog ALSU.v ALSU_tb.sv pack_alsu.sv +cover -covercells
vsim -voptargs=+acc work.ALSU_tb -cover
add wave *
coverage save ALSU_tb.ucdb -onexit
run -all
coverage exclude -src ALSU.v -line 120 -code b
coverage exclude -src ALSU.v -line 120 -code s
quit -sim

vcover report ALSU_tb.ucdb -details -all -output coverage_report.txt
```

Verification plan

	Label	Description	Stimulus Generation	Functionality Check	Functionalit check
2	ALSU_1	we assert reset on start so OUT should be low and led should be low	Directed at the start of the simulation then it randomized under cosntraint to be of high 95 % from time	A checker in the testbench to make sure the output is correct by test function	-
3	ALSU_2	when the byPass_A is asserted OUT take value or reg A igonre Opcode and byPass_B is asserted OUT take value or reg B if Both high so out take input with HIGH priority	Randomized in class under constraint that make bypassA and bypass B is Low 90 % of time	A checker in the testbench to make sure the output is functionally correct by test function	-
4	ALSU_3	when Opcode= 6 or 7 its invalid and if red_op1 or red_op2 is high and OPcode is not or ,xor so its invalid case then Output is low	Randomized in class	A checker in the testbench to make sure the output is functionally correct by test function	we cover this in coverage group in illegal bins
5	ALSU_4	when opcode =OR so output is =|A when red_opA is high same output =|B if this red_op_b high if both high so output check priority and if both low output =A|B	Randomized in class under consraints that Opcode is valid most of simulation and also if OR operation so redA and RedB is high together most of simulation and also at least input if RedA is high only A should have at least 1 bit =1	A checker in the testbench to make sure the output is functionally correct by test function	we cover all Opcodes in from OR to ROTATE and put each one in Bin and and we cover transation from OR to ROTATE and we also we cover all corner casses of A ND B like MAXPOS and MAXNEG and ZERO and when RED_OP is high for values 1 2 4 and reamaining values
6	ALSU_5	when opcode =XOR so output is =^A when red_opA is high same output =^B if this red_op_b high if both high so output check priority and if both low output =A^B	Randomized in class under consraints if xOR operation so redA and RedB is high together most of simulation and also at least input if RedA is high only A should have at least 1 bit =1 and same for B	A checker in the testbench to make sure the output is functionally correct by test function	
7	ALSU_6	when opcode =ADD and full adder on so output =A+B+cin if full adder off out=A+B	Randomized in class	A checker in the testbench to make sure the output is functionally correct by test function	
8	ALSU_7	when opcode =mult so out=A*B	Randomized in class	A checker in the testbench to make sure the output is functionally correct by test function	
9	ALSU_8	when opcode =SHIFT and depend on Direction output will be left or right and serial in	Randomized in class	A checker in the testbench to make sure the output is functionally correct by test function	
10	ALSU_9	when opcode =ROTATE and depend on Direction output will be rotate left or right	Randomized in class	A checker in the testbench to make sure the output is functionally correct by test function	

```
# ** Warning: (vsim-WLF-5000) WLF file currently in use: vsim.wlf
#          File in use by: Marwan  Hostname: DESKTOP-QBO8P56  ProcessID: 17284
#          Attempting to use alternate WLF file "./wlft89vbwd".
# ** Warning: (vsim-WLF-5001) Could not open WLF file: vsim.wlf
#          Using alternate file: ./wlft89vbwd
# ** Warning: (vsim-8474) pack_alsu.sv(60): A higher value '4' is found in bin 'A_data_walkingones' of Coverpoint 'CB1'. It is invalid and will be ignored.
#    Time: 0 ns  Iteration: 0  Instance: /ALSU_tb
# ** Warning: (vsim-8474) pack_alsu.sv(67): A higher value '4' is found in bin 'B_data_walkingones' of Coverpoint 'CB2'. It is invalid and will be ignored.
#    Time: 0 ns  Iteration: 0  Instance: /ALSU_tb
# @10200 the wanted sequence is '{OR, XOR, ADD, MULT, SHIFT, ROTATE}
# @18840 the wanted sequence is '{OR, XOR, ADD, MULT, SHIFT, ROTATE}
# number of correct =801 ,error=0
# ** Note: $stop    : ALSU_tb.sv(62)
#    Time: 32040 ns  Iteration: 1  Instance: /ALSU_tb
# Break in Module ALSU_tb at ALSU_tb.sv line 62
```

Code Coverage report

```
Statement Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Stmts                           49        49         0     100.0

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

Statement Coverage for file ALSU.v --

    1                                           module ALSU(A, B, cin, serial_in, red_op_A, red_op_B, opcode, bypass_A, bypass_B, clk, rst, directi
    2                                           parameter INPUT_PRIORITY = "A";
    3                                           parameter FULL_ADDER = "ON";
    4                                           input  clk, rst, cin, red_op_A, red_op_B, bypass_A, bypass_B, 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 cin_reg, red_op_A_reg, red_op_B_reg, bypass_A_reg, bypass_B_reg, direction_reg, serial_in_reg;
    11                                          reg [2:0] opcode_reg;
    12                                          reg signed [2:0] A_reg, B_reg; //change to signed
    13                                          reg signed [5:0] out_next;
    14                                          wire invalid_red_op, invalid_opcode, invalid;
    15
    16                                          //Invalid handling
    17              1                    484    assign invalid_red_op = (red_op_A_reg | red_op_B_reg) & (opcode_reg[1] | opcode_reg[2]);
    18              1                    463    assign invalid_opcode = opcode_reg[1] & opcode_reg[2];
    19              1                    114    assign invalid = invalid_red_op | invalid_opcode;
    20
    21                                          //Registering input signals
    22              1                   1002    always @(posedge clk or posedge rst) begin
```

```
Branch Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Branches                        31        31         0     100.0

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

Branch Coverage for file ALSU.v --

------------------------------------IF Branch------------------------------------
    23                                      1002     Count coming in to IF
    23              1                         34       if(rst) begin
    35              1                        968       end else begin
Branch totals: 2 hits of 2 branches = 100.0%

------------------------------------IF Branch------------------------------------
    52                                      1019     Count coming in to IF
    52              1                         51       if(rst) begin
    54              1                        968       end else begin
Branch totals: 2 hits of 2 branches = 100.0%

------------------------------------IF Branch------------------------------------
    55                                       968     Count coming in to IF
    55              1                        190           if (invalid)
    57              1                        778           else
Branch totals: 2 hits of 2 branches = 100.0%

------------------------------------IF Branch------------------------------------
    65                                       912     Count coming in to IF
    65              1                         38       if(rst) begin
    68              1                        874       else begin
Branch totals: 2 hits of 2 branches = 100.0%

------------------------------------IF Branch------------------------------------
    69                                       874     Count coming in to IF
```

Toggle Coverage:

Enabled Coverage	Active	Hits	Misses	% Covered
Toggle Bins	130	130	0	100.0

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

Toggle Coverage for File ALSU.v --

Line	Node	1H->0L	0L->1H	"Coverage"
4	serial_in	1	1	100.00
4	rst	1	1	100.00
4	red_op_B	1	1	100.00
4	red_op_A	1	1	100.00
4	direction	1	1	100.00
4	clk	1	1	100.00
4	cin	1	1	100.00
4	bypass_B	1	1	100.00
4	bypass_A	1	1	100.00
5	opcode[2]	1	1	100.00
5	opcode[1]	1	1	100.00
5	opcode[0]	1	1	100.00
6	B[2]	1	1	100.00
6	B[1]	1	1	100.00
6	B[0]	1	1	100.00
6	A[2]	1	1	100.00
6	A[1]	1	1	100.00
6	A[0]	1	1	100.00
7	leds[9]	1	1	100.00
7	leds[8]	1	1	100.00
7	leds[7]	1	1	100.00
7	leds[6]	1	1	100.00
7	leds[5]	1	1	100.00

Function Coverage report

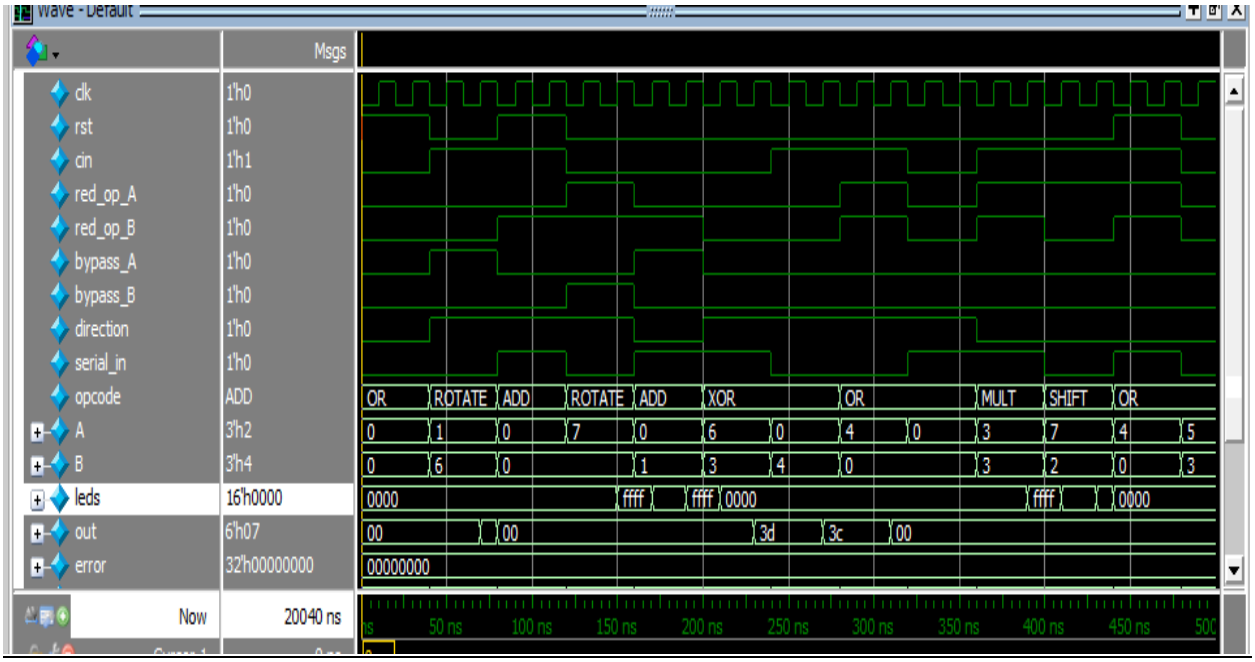
COVERGROUP COVERAGE:

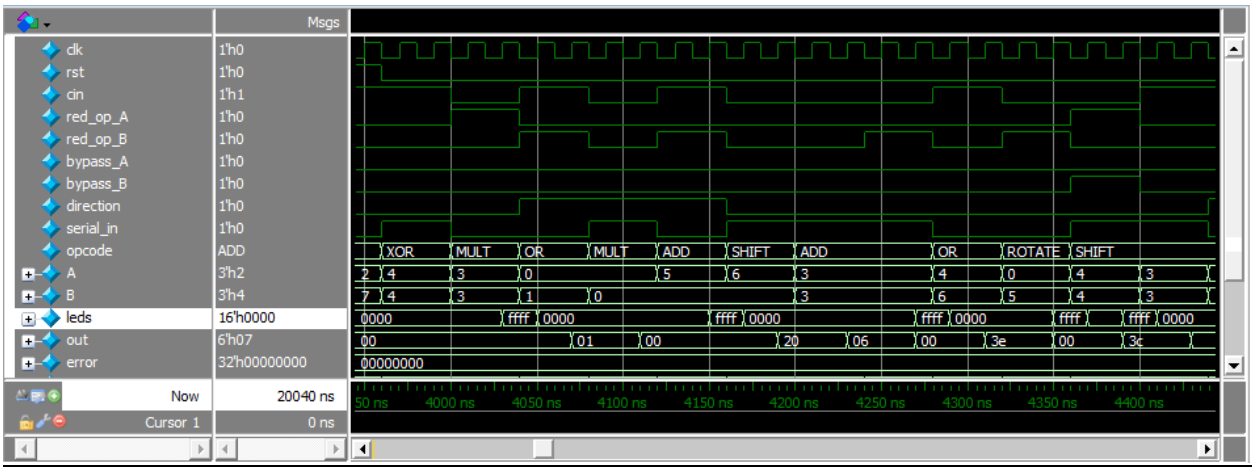
Covergroup	Metric	Goal	Status
TYPE /pack_alsu/transaction/cvr_gp	100.0%	100	Covered
covered/total bins:	17	17	
missing/total bins:	0	17	
% Hit:	100.0%	100	
Coverpoint cvr_gp::CB1	100.0%	100	Covered
covered/total bins:	5	5	
missing/total bins:	0	5	
% Hit:	100.0%	100	
Coverpoint cvr_gp::CB2	100.0%	100	Covered
covered/total bins:	5	5	
missing/total bins:	0	5	
% Hit:	100.0%	100	
Coverpoint cvr_gp::CB3	100.0%	100	Covered
covered/total bins:	7	7	
missing/total bins:	0	7	
% Hit:	100.0%	100	
CLASS transaction			
Covergroup instance \/pack_alsu::transaction::cvr_gp			
	100.0%	100	Covered
covered/total bins:	17	17	
missing/total bins:	0	17	
% Hit:	100.0%	100	
Coverpoint CB1	100.0%	100	Covered
covered/total bins:	5	5	
missing/total bins:	0	5	
% Hit:	100.0%	100	
bin A_data_0	79	1	Covered
bin A_data_max	101	1	Covered
bin A_data_min	126	1	Covered

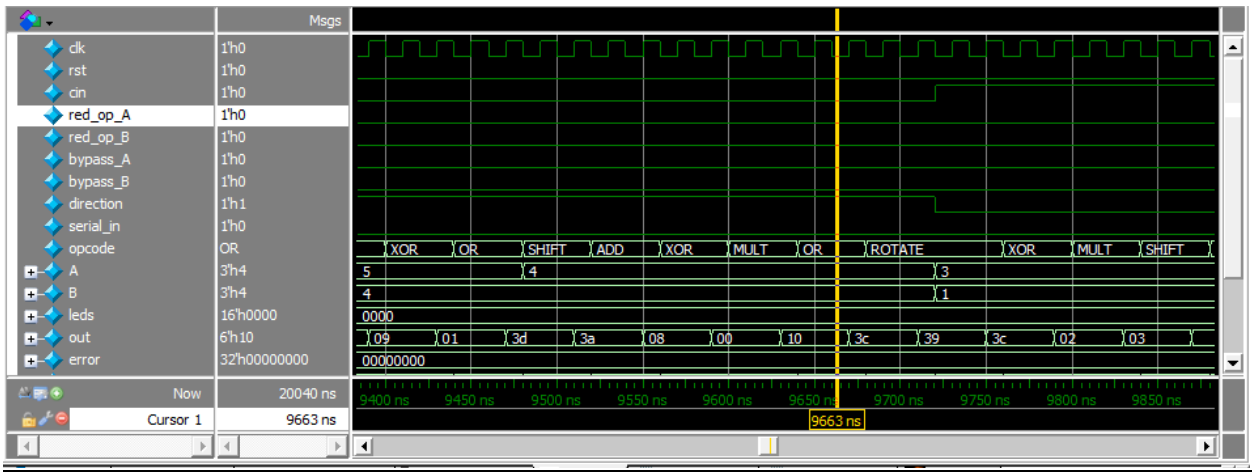
Coverpoint CB1	100.0%	100	Covered
covered/total bins:	5	5	
missing/total bins:	0	5	
% Hit:	100.0%	100	
bin A_data_0	79	1	Covered
bin A_data_max	101	1	Covered
bin A_data_min	126	1	Covered
bin A_data_walkingones[1]	4	1	Covered
bin A_data_walkingones[2]	2	1	Covered
default bin A_data_default	290		Occurred
Coverpoint CB2	100.0%	100	Covered
covered/total bins:	5	5	
missing/total bins:	0	5	
% Hit:	100.0%	100	
bin B_data_0	132	1	Covered
bin B_data_max	117	1	Covered
bin B_data_min	90	1	Covered
bin B_data_walkingones[1]	4	1	Covered
bin B_data_walkingones[2]	6	1	Covered
default bin B_data_default	266		Occurred
Coverpoint CB3	100.0%	100	Covered
covered/total bins:	7	7	
missing/total bins:	0	7	
% Hit:	100.0%	100	
illegal_bin Bins_invalid	0		ZERO
bin Bins_shift[SHIFT]	133	1	Covered
bin Bins_shift[ROTATE]	129	1	Covered
bin Bins_arith[ADD]	119	1	Covered
bin Bins_arith[MULT]	128	1	Covered
bin Bins_bitwise[OR]	127	1	Covered
bin Bins_bitwise[XOR]	123	1	Covered
bin Bins_trans	2	1	Covered

TOTAL COVERGROUP COVERAGE: 100.0% COVERGROUP TYPES: 1

Simulation




Msgs
clk
rst
cin
red_op_A
red_op_B
bypass_A
bypass_B
direction
serial_in
opcode
A
B
leds
out
error
1'h0
1'h0
1'h1
1'h0
1'h0
1'h0
1'h0
1'h0
1'h0
ADD
3'h2
3'h4
16'h0000
6'h07
32'h00000000
XOR MULT OR MULT ADD SHIFT ADD OR ROTATE SHIFT
2 4 3 0 5 6 3 4 0 4 3
7 4 3 1 0 3 6 5 4 3
0000 ffff 0000 ffff 0000 ffff 0000 ffff ffff 0000
00 01 00 20 06 00 3e 00 3c
00000000
Now
20040 ns
Cursor 1
0 ns
50 ns 4000 ns 4050 ns 4100 ns 4150 ns 4200 ns 4250 ns 4300 ns 4350 ns 4400 ns


Msgs
clk
rst
cin
red_op_A
red_op_B
bypass_A
bypass_B
direction
serial_in
opcode
A
B
leds
out
error
1'h0
1'h0
1'h0
1'h0
1'h0
1'h0
1'h0
1'h1
1'h0
OR
3'h4
3'h4
16'h0000
6'h10
32'h00000000
XOR OR SHIFT ADD XOR MULT OR ROTATE XOR MULT SHIFT
5 4 3
4 1
0000
09 01 3d 3a 08 00 10 3c 39 3c 02 03
00000000
Now
20040 ns
Cursor 1
9663 ns
9400 ns 9450 ns 9500 ns 9550 ns 9600 ns 9650 ns 9700 ns 9750 ns 9800 ns 9850 ns
9663 ns

MEMORY

Design

```
module my_mem(
input clk,
input write,
input read,
input [7:0] data_in,
input [15:0] address,
output reg [8:0] data_out
);
 // Declare a 9-bit associative array using the logic data type & the key of int datatype
logic [8:0]mem_array[int];

always @(posedge clk) begin
    if (write)
    mem_array[address] = {^data_in, data_in};

    else if (read)
    data_out = mem_array[address];
    else
    mem_array[address]= mem_array[address];
    end
endmodule
```

Tb

```
import pack::*;
module my_mem_tb();

bit clk=0;
logic write;
logic read;
logic [DATA_IN_BITS-1:0] data_in;
logic [ADDRESS_BITS-1:0] address;
logic [DATA_OUT_BITS-1:0] data_out;

int error=0,correct=0;

localparam No_of_inputs=100 ;

my_mem tb(.*);
always #5 clk=!clk;

transaction tr=new();

```

```
initial begin
    tr.write=1;
    tr.read=0;
    write=tr.write;
    read=tr.read;

    stimulus_gen(tr,No_of_inputs);

    for(int i=0; i<No_of_inputs;i++)begin
        data_in=tr.data_to_write_array[i];
        address=tr.address_array[i];
        @(negedge clk);
    end
    tr.write=0;
    tr.read=1;
    write=tr.write;
    read=tr.read;

    for(int i=No_of_inputs-1; i>=0;i--)begin
        address=tr.address_array[i];
        tr.data_out=data_out;
        self_check(tr);
        tr.data_read_queue.push_back(data_out);

    end
    while(tr.data_read_queue.size())
        $display("data =%0d",tr.data_read_queue.pop_front);

```

```
    tr.write=1;
    tr.read=0;
    write=tr.write;
    read=tr.read;

    stimulus_gen(tr,15);

    for(int i=0; i<15;i++)begin
        data_in=tr.data_to_write_array[i];
        address=tr.address_array[i];
        @(negedge clk);
    end
    tr.write=0;
    tr.read=1;
    write=tr.write;
    read=tr.read;
    for(int i=14; i>=0;i--)begin
        address=tr.address_array[i];
        tr.data_out=data_out;
        self_check(tr);
        tr.data_read_queue.push_back(data_out);
    end
    tr.write=0;
    tr.read=0;
    write=tr.write;
    read=tr.read;
    self_check(tr);
    $display("error=%0d,correct=%0d",error,correct);
$stop;
end
```

```
    self_check(tr);

    $display("error=%0d,correct=%0d",error,correct);

$stop;

end

task stimulus_gen(transaction trans,input int no_of_inputs);
    for(int i=0;i<no_of_inputs;i++)begin
        assert(trans.randomize());
        trans.address_array[i]=trans.address;
        trans.data_to_write_array[i]=trans.data_in;
    end
    golden_model(trans);
endtask

task golden_model(transaction fill);
    if(tr.write)begin
        foreach(fill.address_array[i])begin//{~^data_in, data_in}
            fill.data_read_expect_assoc[fill.address_array[i]]={^fill.data_to_write_array[i],fill.data_to_write_array[i]};
        end
    end
endtask

task self_check(transaction tr);
    @(negedge clk);
    if(tr.data_read_expect_assoc[address] != data_out)begin
        $display("@%0t there is error, address=%0d ,data_out=%0d ,right data_out=%0d",$time ,address,tr.data_out,tr.data_read_expect_assoc[a
        error++;
    end
```

Package

```
package pack;
    localparam TEST =100,DATA_IN_BITS=8,DATA_OUT_BITS=9,ADDRESS_BITS=16 ;


    class transaction;
        bit write;
        bit read;
        rand bit [DATA_IN_BITS-1:0] data_in;
        rand bit [ADDRESS_BITS-1:0] address;
        bit [DATA_OUT_BITS-1:0] data_out;

        logic [ADDRESS_BITS-1:0] address_array[];
        logic [DATA_IN_BITS-1:0] data_to_write_array[];
        logic [DATA_OUT_BITS-1:0] data_read_expect_assoc[int];
        logic [DATA_OUT_BITS-1:0] data_read_queue[$];

        function new();
            address_array=new[TEST];
            data_to_write_array=new[TEST];
        endfunction

    endclass

endpackage
```

Do file

```
vlib work
vlog my_mem.sv my_mem_tb.sv pack.sv +cover -covercells
vsim -voptargs=+acc work.my_mem_tb -cover
add wave *
coverage save memory_tb.ucdb -onexit -du work.my_mem
run -all
quit -sim

vcover report memory_tb.ucdb -details -all -output coverage_report.txt
```

Verification plan

1	Label	Description	Stimulus Generation	Functionality Check
2	my_mem_write	we write at first in 100 random location with 100 random data	write=1,read=0,and randomized data_in,address and save value in two dynamic array	
3	my_mem_read	we read from 100 random location that store in 2 dynamic arrays	write=0,read=1,and and send address from dynamic array	we check functionality by co,pare valuse store in golden model in Associative array with data_out

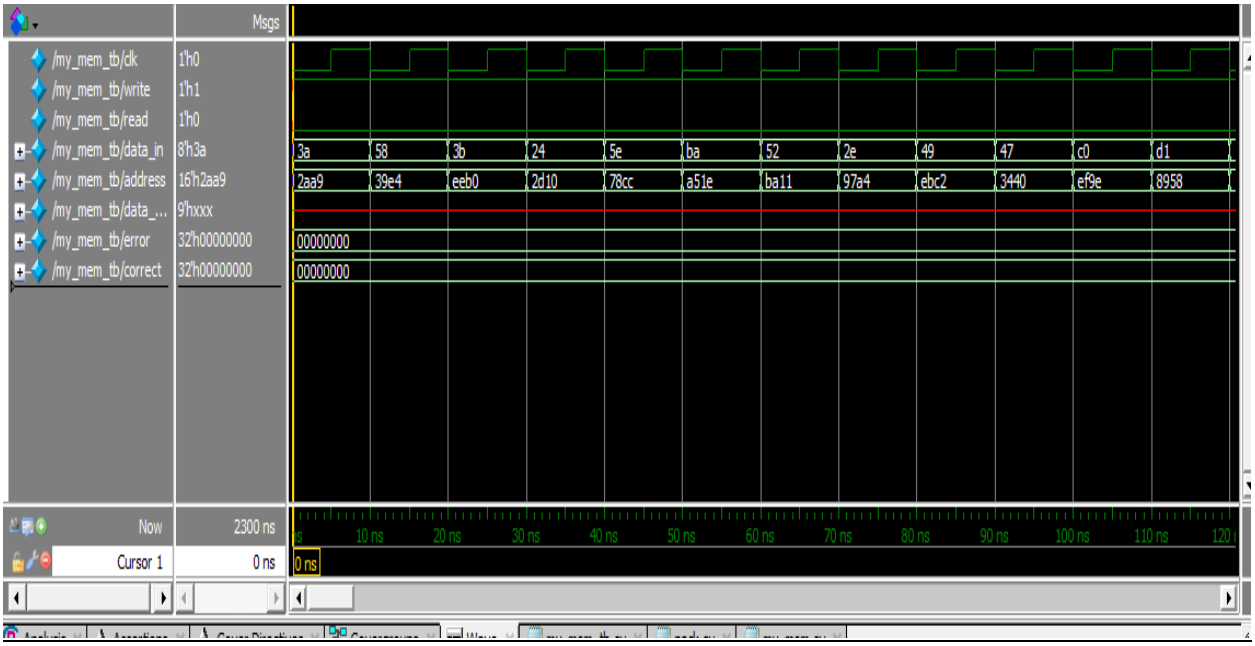
Code Coverage report

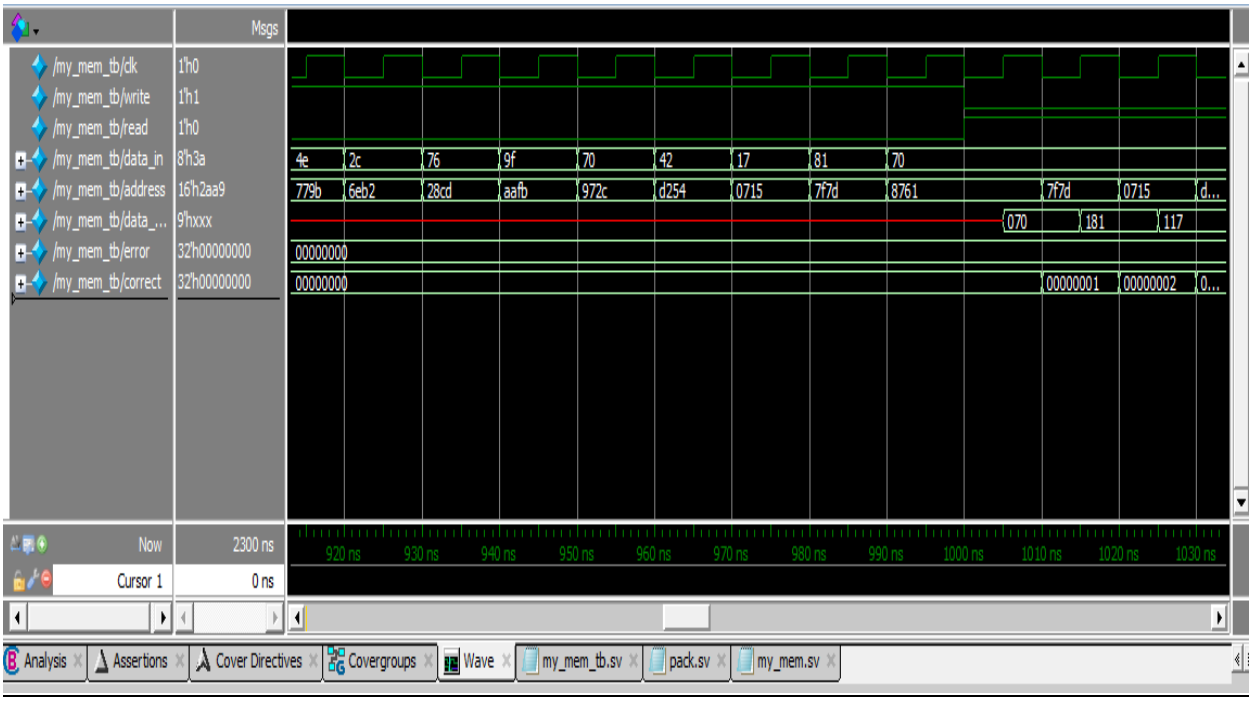
```
================================================================================
Statement Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Stmts                            4         4         0     100.0

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

Statement Coverage for file my_mem.sv --

    1                                           module my_mem(
    2                                           input clk,
    3                                           input write,
    4                                           input read,
    5                                           input [7:0] data_in,
    6                                           input [15:0] address,
    7                                           output reg [8:0] data_out
    8                                           );
    9                                            // Declare a 9-bit associative array using the logic data type & the key of int datatype
    10                                          logic [8:0]mem_array[int];
    11
    12              1                     231   always @(posedge clk) begin
    13                                              if (write)
    14              1                     115       mem_array[address] = {~^data_in, data_in};
    15
    16                                              else if (read)
    17              1                     115       data_out = mem_array[address];
    18                                              else
    19              1                       1       mem_array[address]= mem_array[address];
    20                                              end
    21                                          endmodule
    22
```

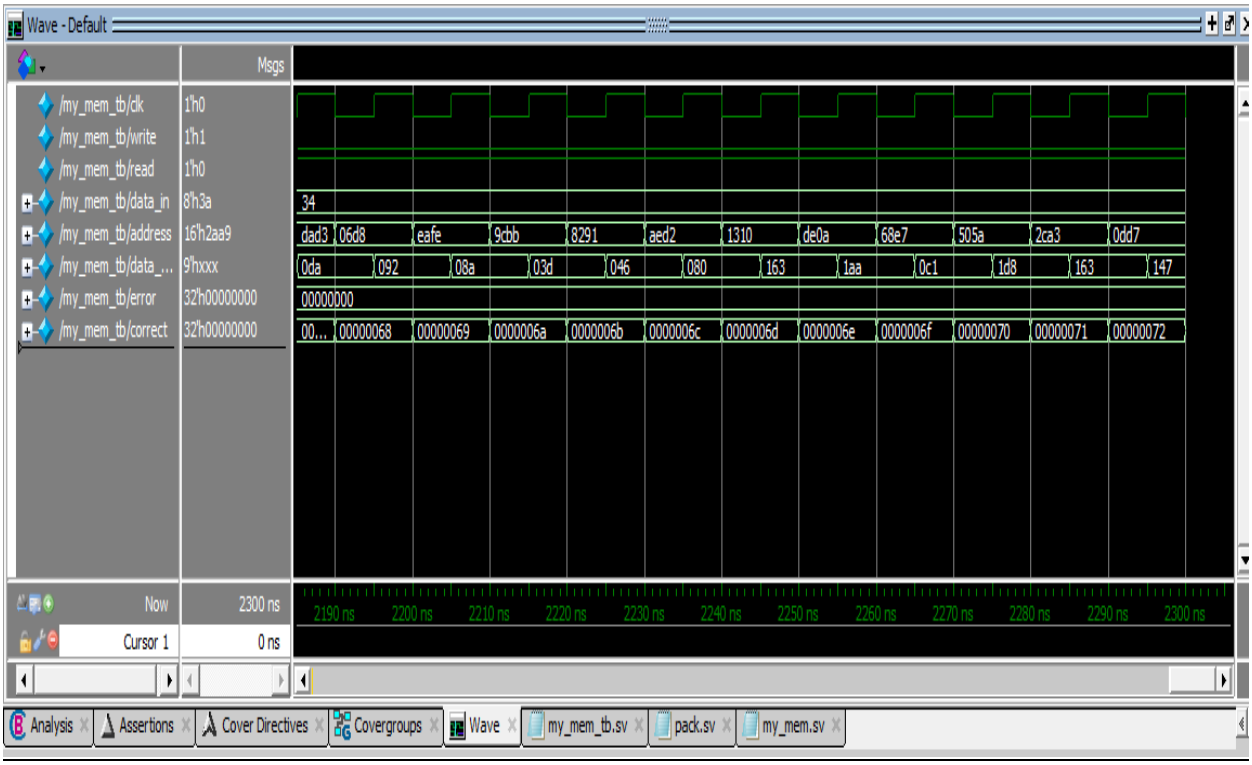
```
Branch Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Branches                         3         3         0     100.0

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

Branch Coverage for file my_mem.sv --

------------------------------------IF Branch------------------------------------
    13                                      231     Count coming in to IF
    13              1                       115         if (write)
    16              1                       115         else if (read)
    18              1                         1         else
Branch totals: 3 hits of 3 branches = 100.0%


Condition Coverage:
    Enabled Coverage            Active   Covered    Misses % Covered
    ----------------            ------      ----    ------ ---------
    FEC Condition Terms              0         0         0     100.0
Expression Coverage:
    Enabled Coverage            Active   Covered    Misses % Covered
    ----------------            ------      ----    ------ ---------
    FEC Expression Terms             0         0         0     100.0
FSM Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    FSMs                                                       100.0
        States                       0         0         0     100.0
        Transitions                  0         0         0     100.0
```

```
Toggle Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Toggle Bins                     72        72         0     100.0

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

Toggle Coverage for File my_mem.sv --

       Line                                 Node      1H->0L      0L->1H  "Coverage"
--------------------------------------------------------------------------------------
          2                                  clk           1           1     100.00
          3                                write           1           1     100.00
          4                                 read           1           1     100.00
          5                           data_in[7]           1           1     100.00
          5                           data_in[6]           1           1     100.00
          5                           data_in[5]           1           1     100.00
          5                           data_in[4]           1           1     100.00
          5                           data_in[3]           1           1     100.00
          5                           data_in[2]           1           1     100.00
          5                           data_in[1]           1           1     100.00
          5                           data_in[0]           1           1     100.00
          6                           address[9]           1           1     100.00
          6                           address[8]           1           1     100.00
          6                           address[7]           1           1     100.00
          6                           address[6]           1           1     100.00
          6                           address[5]           1           1     100.00
          6                           address[4]           1           1     100.00
          6                           address[3]           1           1     100.00
          6                           address[2]           1           1     100.00
          6                           address[1]           1           1     100.00
          6                          address[15]           1           1     100.00
          6                          address[14]           1           1     100.00
          6                          address[13]           1           1     100.00
```

Simulation


Msgs
/my_mem_tb/clk
/my_mem_tb/write
/my_mem_tb/read
/my_mem_tb/data_in
/my_mem_tb/address
/my_mem_tb/data_...
/my_mem_tb/error
/my_mem_tb/correct
1'h0
1'h1
1'h0
8'h3a
16'h2aa9
9'hxxx
32'h00000000
32'h00000000
3a 58 3b 24 5e ba 52 2e 49 47 c0 d1
2aa9 39e4 eeb0 2d10 78cc a51e ba11 97a4 ebc2 3440 ef9e 8958
00000000
00000000
Now
2300 ns
Cursor 1
0 ns
10 ns 20 ns 30 ns 40 ns 50 ns 60 ns 70 ns 80 ns 90 ns 100 ns 110 ns 120


Msgs
/my_mem_tb/clk
/my_mem_tb/write
/my_mem_tb/read
/my_mem_tb/data_in
/my_mem_tb/address
/my_mem_tb/data_...
/my_mem_tb/error
/my_mem_tb/correct
1'h0
1'h1
1'h0
8'h3a
16'h2aa9
9'hxxx
32'h00000000
32'h00000000
4e 2c 76 9f 70 42 17 81 70
779b 6eb2 28cd aafb 972c d254 0715 7f7d 8761 7f7d 0715 d...
070 181 117
00000000
00000000 00000001 00000002 0...
Now
2300 ns
Cursor 1
0 ns
920 ns 930 ns 940 ns 950 ns 960 ns 970 ns 980 ns 990 ns 1000 ns 1010 ns 1020 ns 1030 ns
Analysis
Assertions
Cover Directives
Covergroups
Wave
my_mem_tb.sv
pack.sv
my_mem.sv




```
# data =112
# data =385
# data =279
# data =322
# data =112
# data =415
# data =118
# data =44
# data =334
# data =372
# data =345
# data =505
# data =261
# data =502
# data =196
# data =259
# data =157
# data =109
# data =475
# data =345
# data =502
# data =451
# data =302
# data =372
# data =93
# data =296
```

```
# data =94
# data =213
# data =179
# data =422
# data =339
# data =62
# data =361
# data =70
# data =412
# data =55
# data =227
# data =316
# data =171
# data =49
# data =481
# data =128
# data =19
# data =388
# data =307
# data =62
# data =128
# data =38
# data =42
# data =379
# data =247
# data =143
# data =100
# data =174
# data =87
# data =109
```

```
# data =494
# data =362
# data =494
# data =319
# data =358
# data =138
# data =466
# data =107
# data =346
# data =234
# data =152
# data =248
# data =465
# data =448
# data =327
# data =73
# data =302
# data =82
# data =186
# data =94
# data =292
# data =59
# data =88
# data =314
# error=0,correct=116
```