Assignment 2

Q1

CODE :

array.sv

```
1   module array ();
2       int dyn_arr1 [] ;
3       int dyn_arr2 [] = '{9,1,8,3,4,4} ;
4       int size_arr;
5
6       initial begin
7           dyn_arr1 = new [6];
8           foreach (dyn_arr1 [i])
9                   dyn_arr1[i] = i;
10
11          size_arr = $size(dyn_arr1);
12          $display("dyn_arr1 = %p %d , dyn_arr2 = %p", dyn_arr1,size_arr, dyn_arr2);
13
14          dyn_arr1.delete();
15
16          dyn_arr2.reverse();
17          $display("dyn_arr2 = %p", dyn_arr2);
18
19          dyn_arr2.sort();
20          $display("dyn_arr2 = %p", dyn_arr2);
21
22          dyn_arr2.rsort();
23          $display("dyn_arr2 = %p", dyn_arr2);
24
25          dyn_arr2.shuffle();
26          $display("dyn_arr2 = %p", dyn_arr2);
27
28      end
29
30
31  endmodule
```

DISPLAY :

```
VSIM 23> run -all
# dyn_arr1 = '{0, 1, 2, 3, 4, 5}           6 , dyn_arr2 = '{9, 1, 8, 3, 4, 4}
# dyn_arr2 = '{4, 4, 3, 8, 1, 9}
# dyn_arr2 = '{1, 3, 4, 4, 8, 9}
# dyn_arr2 = '{9, 8, 4, 4, 3, 1}
# dyn_arr2 = '{8, 1, 4, 3, 9, 4}
```

Q2

CODE TB :

counter_tb.sv

```
import counter_pkg ::*;
module counter_tb ();

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

    logic [WIDTH-1:0] count_out_ex;
    int error_counter = 0;
    int correct_counter = 0;

    counter #(.WIDTH(WIDTH)) g (.*);

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

   task cheack_result (input [WIDTH-1:0] x);

      @(negedge clk);
      if (x != count_out) begin
          $display("%t error",$time);
          error_counter++;
      end
      else
          correct_counter++;

   endtask

   task cheack_reset ;
          rst_n = 0;
          @(negedge clk);
          cheack_result (0);
          rst_n = 1;

    endtask

```

```
48      task cheack_max ;
49          @(negedge clk);
50          if (max_count)
51              correct_counter++;
52          else begin
53              $display("%t error",$time);
54              correct_counter++;
55          end
56      endtask
57
58      task cheack_zero ;
59          @(negedge clk);
60          if (zero)
61              correct_counter++;
62          else begin
63              $display("%t error",$time);
64              correct_counter++;
65          end
66      endtask
67
68      initial begin
69      ///label1/////
70      cheack_reset;
71      //////label2////
72      load_n = 0;
73      data_load = 4'b0001;
74      cheack_result (data_load);
75      load_n = 1;
76      ///label3/////
77      cheack_reset;
78      ////////label4//////
79      ce = 1;
80      up_down = 0;
81      cheack_result (count_out-1);
82      up_down = 1;
83      cheack_result(count_out +1);
84      //////label5////
85      ce = 0;
86      load_n = 0;
87      data_load = 4'b0000;
88      cheack_result (data_load);
89      cheack_zero ;
90      load_n = 1;
```

```
    ce = 1;
    load_n = 0;
    data_load = 4'b0000;
    cheack_result (data_load);
    cheack_zero ;
    load_n = 1;
    ////////label7//////
    up_down = 0;
    cheack_result (count_out-1);
    up_down = 1;
    cheack_result(count_out +1);
    //////label8////
    load_n = 0;
    data_load = 4'b0100;
    cheack_result (data_load);
    load_n = 1;
    ////////label9//////
    up_down = 0;
    cheack_result (count_out-1);
    up_down = 1;
    cheack_result(count_out +1);
    //////label10////
    load_n = 0;
    data_load = 4'b1111;
    cheack_result (data_load);
    cheack_max ;
    load_n = 1;
    //////label11////
    load_n = 0;
    data_load = 4'b0010;
    cheack_result (data_load);
    load_n = 1;
    //////label12////
    load_n = 0;
    data_load = 4'b1100;
    cheack_result (data_load);
    load_n = 1;
    ///label13/////
    cheack_reset;


    $display("error_counter = %d  correct_counter = %d",error_counter,correct_counter);
    $stop;
  end
```

```
    counter_class counter1;
    initial begin
     counter1 = new;
     for (int i = 0 ; i<100 ; i++) begin
         assert (counter1.randomize());
         rst_n = counter1.reset;
         load_n = counter1.load_n;
         ce = counter1.ce;
         data_load = counter1.data;
         up_down = counter1.up_down;
         golden_model ;
         cheack_out ;

     end
         $display ("error_counter = %d  correct_counter = %d",error_counter,correct_counter);
         $stop;
    end
     initial begin
         $monitor ("clk=%b rst=%b load=%b up_down=%b data=%d count=%d count_ex=%d",clk,rst_n,load_n,up_down,data_load,count_out,count_out_ex);

     end
    task golden_model ;
     if (!rst_n)
         count_out_ex <= 0;
     else if (!load_n)
         count_out_ex <= data_load;
     else if (ce)
         if (up_down)
             count_out_ex <= count_out_ex + 1;
         else
             count_out_ex <= count_out_ex - 1;
    endtask

    task cheack_out ;

       @(negedge clk);
       if (count_out_ex != count_out) begin
          $display("%t error",$time);
          error_counter++;
       end
       else
          correct_counter++;

    endtask

endmodule
```

CODE PKG :

counter_pkg.sv

```
1   package counter_pkg;
2
3     class counter_class ;
4       rand logic reset;
5       rand logic load_n;
6       rand logic ce;
7       rand logic [3:0] data;
8       rand logic up_down;
9
10      constraint counter_cst {
11          reset dist {0 := 1, 1:= 99};
12          load_n  dist {0 := 30, 1:= 70};
13          ce dist {0 := 30, 1:= 70};
14
15
16      };
17
18    endclass
19
20  endpackage
```

DO FILE :

counter.do

```
1   vlib work
2   vlog counter.v counter_tb.sv counter_pkg.sv  +cover -covercells
3   vsim -voptargs=+acc work.counter_tb -cover
4   add wave *
5   coverage save counter_tb.ucdb -onexit
6   run -all
```

DISPLAY :

```
# error_counter =           0  correct_counter =         119
# ** Note: $stop    : counter_tb.sv(149)
#    Time: 244 ns  Iteration: 1  Instance: /counter_tb
# Break in Module counter_tb at counter_tb.sv line 149
```

COVERAGE :

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

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

Toggle Coverage for instance /counter_tb/g --

                                          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)

--------------------------------------------------------------------------------
```

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

                                Condition Details
--------------------------------------------------------------------------------
```

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

---------------------------------Branch Details---------------------------------
```

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

VERIFICATION PLAN :

label	description	stimulus generation	function functionality cheack
counter_1	when the rst is asserted the output dout value must be low	directed at the start of the simulation	cheacker in the testbench to make sure the output is correct
counter_2	verifying load_n desserted	directed at the start of the simulation	cheacker in the testbench to make sure the output is correct
counter_3	when the rst is asserted the output dout value must be low	directed at the start of the simulation	cheacker in the testbench to make sure the output is correct
counter_4	verifying ce desserted and up_down eqaual 0	directed at the start of the simulation	cheacker in the testbench to make sure the output is correct
counter_5	verifying ce desserted and load_en active	directed at the start of the simulation	cheacker in the testbench to make sure the output is correct
counter_6	verifying ce asserted and load_en active	directed at the start of the simulation	cheacker in the testbench to make sure the output is correct
counter_14	verifying randomize and constraint input		

Q3

CODE TB :

ALU.sv

```
import ALU_pkg ::*;
module ALU ();
    parameter INPUT_PRIORITY = "A";
    parameter FULL_ADDER = "ON";
    logic clk, cin, rst, 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;
    bit [15:0] leds_ex;
    logic [5:0] out;
    bit [5:0] out_ex;

    int error_counter = 0;
    int correct_counter = 0;

    ALSU o (A,B,cin,serial_in,red_op_A,red_op_B,opcode,bypass_A,bypass_B,clk,rst,direction,leds,out);
    ALSU O2 (A,B,cin,serial_in,red_op_A,red_op_B,opcode,bypass_A,bypass_B,clk,rst,direction,leds_ex,out_ex);

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

    ALU_class my_alu;
    initial begin

        //1
        cheack_reset;

        //2
        my_alu = new;
        repeat (90000) begin
            assert (my_alu.randomize());
            rst = my_alu.rst;
            red_op_A = my_alu.red_op_A;
            red_op_B = my_alu.red_op_B;
            bypass_A = my_alu.bypass_A;
            bypass_B = my_alu.bypass_B;
            direction = my_alu.direction;
            cin = my_alu.cin;
            serial_in = my_alu.serial;
            A = my_alu.A;
            B = my_alu.B;
            opcode = my_alu.my_opcode;
            cheack_result;
        end
```

```
        //3
        cheack_reset;

        //4
        repeat (90000) begin
            assert (my_alu.randomize());
            cheack_result ;
        end

        //5
        cheack_reset;

            $display("error_counter = %d  correct_counter = %d",error_counter,correct_counter);
            $stop;
    end

    task cheack_result ;
      @(negedge clk);
      if ((out_ex != out) || (leds_ex != leds)) begin
         $display("%t error",$time);
         error_counter++;
      end
      else
        correct_counter++;

    endtask

    task cheack_reset ;
        rst = 0;
        @(negedge clk);
        cheack_result ;
        rst = 1;

    endtask
endmodule
```

CODE PKG :

ALU_pkg.sv

```
1   package ALU_pkg;
2
3       typedef enum { opcode0,opcode1,opcode2,opcode3,opcode4,opcode5,invalid6,invalid7 } opcode_e;
4
5       localparam MAXPOS = 3, ZERO = 0, MAXNEG = -4;
6
7       class ALU_class ;
8        rand logic rst;
9        rand logic signed [2:0] A,B;
10       rand logic red_op_A,red_op_B;
11       rand opcode_e my_opcode;
12       rand logic bypass_A, bypass_B;
13       rand logic direction,cin;
14       rand logic serial;        Chat (CTRL + I) / Edit (CTRL + L)
15
16       constraint ALU_cns {
17           //label1
18           rst dist { 0:=99 , 1:=1};
19           //label2
20           (my_opcode == opcode2 || my_opcode == opcode3) -> A dist { MAXNEG,MAXPOS,ZERO :/ 60};
21           (my_opcode == opcode2 || my_opcode == opcode3) -> B dist { MAXNEG,MAXPOS,ZERO :/ 60};
22           //label3
23           (my_opcode == opcode0 || my_opcode == opcode1 && red_op_A ==1) -> A dist { 1,2,4 :/ 80};
24           (my_opcode == opcode0 || my_opcode == opcode1 && red_op_A ==1) -> B dist { 0 := 100};
25           //label4
26           (my_opcode == opcode0 || my_opcode == opcode1 && red_op_B ==1) -> B dist { 1,2,4 :/ 80};
27           (my_opcode == opcode0 || my_opcode == opcode1 && red_op_B ==1) -> A dist { 0 := 100};
28           //label5
29           my_opcode dist { [invalid6:invalid7] := 20,[opcode0:opcode5] := 80};
30           //label6
31           bypass_A dist { 0:=80 , 1:=20};
32           bypass_B dist { 0:=80 , 1:=20};
33       }
34
35       endclass
36
37  endpackage
```

DO FILE :

ALU.do

```
1   vlib work
2   vlog ALSU.v ALU.sv ALU_pkg.sv  +cover -covercells
3   vsim -voptargs=+acc work.ALU -cover
4   add wave *
5   coverage save ALU.ucdb -onexit        Chat (CTRL + I) / Edit (CTRL + L)
6   run -all
```

DISPLAY :

```
# Loading work.ALSU(fast)
# error_counter =           0   correct_counter =      180003
# ** Note: $stop    : ALU.sv(63)
#    Time: 360012 ns  Iteration: 1  Instance: /ALU
```

COVERAGE :

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

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

Toggle Coverage for instance /ALU/o --

                                          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)
```

VERIFICATION PLAN :

	label	description	stimulus generation	function functionality cheack
2	ALU_1	when the rst is asserted the output dout value must be low	directed at the start of the simulation	cheacker in the testbench to make sure the output is correct
3	ALU_2	verifying randomize and constraint input	directed at the start of the simulation	cheacker in the testbench to make sure the output is correct
4	ALU_3	when the rst is asserted the output dout value must be low	directed at the start of the simulation	cheacker in the testbench to make sure the output is correct
5	ALU_4	verifying randomize and constraint input	directed at the start of the simulation	cheacker in the testbench to make sure the output is correct
6	ALU_5	when the rst is asserted the output dout value must be low	directed at the start of the simulation	cheacker in the testbench to make sure the output is correct