Assignment 1

By: Abdelrahman Maher Hassan

Question 1

Codes:

```
module priority_enc (
input  clk,
input  rst,
input  [3:0] D,
output reg [1:0] Y,
output reg valid
);

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

Test Bench:

```
module priority_enc_TB ();

    reg   clk, rst;
    reg   [3:0] D;
    wire  [1:0] Y;
    wire  valid;

    integer correct_count, error_count;

    priority_enc Encoder_DUT (
                .clk(clk),
                .rst(rst),
                .D(D),
                .Y(Y),
                .valid(valid));

    initial begin
        clk =0;

        forever
          #1 clk = ~clk;
    end

    initial begin
        correct_count = 0;
        error_count = 0;
        D = 0;

        check_reset;

        D = 4'b1111;
        check_result (3,1);

        D = 4'b1000;
        check_result (0,1);

        D = 4'b1100;
        check_result (1,1);

        D = 4'b0000;
        check_result (2'bXX,0);

        D = 4'b1110;
        check_result (2,1);

        $display( "%t: At the END of the test the error count = %d and the correct count = %d", $time, error_count, correct_count);
        $stop;

    end

    task check_result (input [1:0]expected_result_Y, input expected_result_valid);
```

```
49      end
50
51      task check_result (input [1:0]expected_result_Y, input expected_result_valid);
52          @(negedge clk);
53          if ( expected_result_Y !== Y )
54          begin
55              error_count++;
56              $display("%t: Error in Y value = %d and the expected Y = %d",$time,Y,expected_result_Y);
57          end
58
59          else if (expected_result_valid !== valid)
60          begin
61              error_count++;
62              $display("%t: Error in valid value = %d and the expected valid = %d",$time,expected_result_valid,valid);
63          end
64
65          else
66              correct_count++;
67      endtask
68
69
70      task check_reset ();
71            rst = 1;
72            @(negedge clk);
73
74            if( Y !== 0)
75            begin
76              error_count++;
77              $display("%t: Error in reset value",$time);
78            end
79
80            else
81              correct_count++;
82
83            rst = 0;
84      endtask
85
86  endmodule
```

Simulation:

Question 2

Codes:

```
module ALU_4_bit (
    input  clk,
    input  reset,
    input  [1:0] Opcode,   // The opcode
    input  signed [3:0] A, // Input data A in 2's complement
    input  signed [3:0] B, // Input data B in 2's complement

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

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

   localparam Add            = 2'b00; // A + B
   localparam Sub            = 2'b01; // A - B
   localparam Not_A          = 2'b10; // ~A
   localparam ReductionOR_B  = 2'b11; // |B

   // Do the operation
   always @(*) begin
      case (Opcode)
         Add:            Alu_out = A + B;
         Sub:            Alu_out = A - B;
         Not_A:          Alu_out = ~A;
         ReductionOR_B:  Alu_out = |B;
        default: Alu_out = 5'b0;
      endcase
   end // always @ *

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

endmodule
```

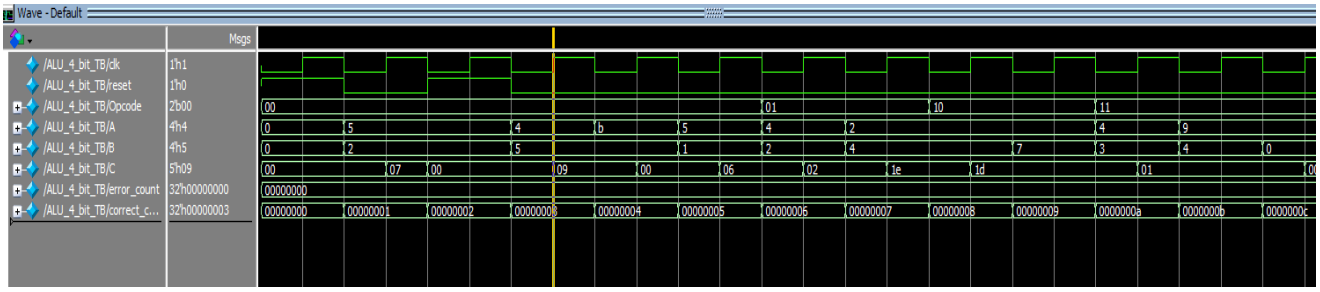
Test Bench:

```
module ALU_4_bit_TB ();
    reg  clk, reset;
    reg  [1:0] Opcode;        // The opcode
    reg  signed [3:0] A, B;   // Input data B in 2's complement

    wire signed [4:0] C;       // ALU output in 2's complement


   ALU_4_bit DUT_ALU (
                      .clk(clk),
                      .reset(reset),
                      .Opcode(Opcode),
                      .A(A),
                      .B(B),
                      .C(C));

   localparam Add            = 2'b00; // A + B
   localparam Sub            = 2'b01; // A - B
   localparam Not_A          = 2'b10; // ~A
   localparam ReductionOR_B  = 2'b11; // |B

   integer error_count, correct_count;

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

   initial begin
      error_count = 0;
      correct_count = 0;
      A = 0;
      B = 0;
      Opcode = 0;
      check_reset;

      A = 5;
      B = 2;
      Opcode = 0;
      check_result(7);

      check_reset;

      A = 4;
      B = 5;
      Opcode = 0;
      check_result(9);

      A = 11;
      B = 5;
```

```
49
50          A = 11;
51          B = 5;
52          Opcode = 0;
53          check_result(0);
54
55          A = 5;
56          B = 1;
57          Opcode = 0;
58          check_result(6);
59
60          A = 4;
61          B = 2;
62          Opcode = 1;
63          check_result(2);
64
65          A = 2;
66          B = 4;
67          Opcode = 1;
68          check_result(-2);
69
70          A = 2;
71          B = 4;
72          Opcode = 2;
73          check_result(-3);
74
75          A = 2;
76          B = 7;
77          Opcode = 2;
78          check_result(-3);
79
80          A = 4;
81          B = 3;
82          Opcode = 3;
83          check_result(1);
84
85          A = 9;
86          B = 4;
87          Opcode = 3;
88          check_result(1);
89
90          A = 9;
91          B = 0;
92          Opcode = 3;
93          check_result(0);
94
95          $display("%t: At the END of the test the error count = %d and the correct count = %d", $time, error_count, correct_count);
96          $stop;
97      end
98
99      task check_reset();
100         reset = 1;
```

```
88          check_result(1);
89
90          A = 9;
91          B = 0;
92          Opcode = 3;
93          check_result(0);
94
95          $display("%t: At the END of the test the error count = %d and the correct count = %d", $time, error_count, correct_count);
96          $stop;
97      end
98
99      task check_reset();
100         reset = 1;
101         @(negedge clk);
102
103         if( C !== 0)
104         begin
105             error_count++;
106             $display("%t: Error in reset.", $time);
107         end
108
109         else
110             correct_count++;
111
112         reset = 0;
113     endtask
114
115     task check_result (input [4:0]expected_output);
116         @(negedge clk);
117
118         if( C !== expected_output)
119         begin
120             error_count++;
121             $display("%t: Error in the output %d, and the expected is = %d.", $time, C, expected_output);
122         end
123
124         else
125             correct_count++;
126     endtask
127 endmodule
```

Simulation:


Wave - Default
Msgs
/ALU_4_bit_TB/clk
/ALU_4_bit_TB/reset
/ALU_4_bit_TB/Opcode
/ALU_4_bit_TB/A
/ALU_4_bit_TB/B
/ALU_4_bit_TB/C
/ALU_4_bit_TB/error_count
/ALU_4_bit_TB/correct_c...
1'h1
1'h0
2'b00
4'h4
4'h5
5'h09
32'h00000000
32'h00000003

Question 3

A Code:

```
//specs: if A is high in +ve edge oa aclock then signal B should be high after 2 clock cycles

module Q3_A ();

    sequence sr1;
        ##2 b;
    endsequence

    property pr1;
        @(posedge clk) a |-> sr1;
    endproperty

    assert property (pr1) $display("%t: Pass",$time); else $display("%t: Fail",$time);



endmodule
```

B Code:

```
//specs: if a is high & b is high then signal c should be high after 1 to 3 clock cycles later

module Q3_B ();

    sequence sr1;
        ##[1:3] c;
    endsequence

    property pr1;
        @(posedge clk) (a & b) |-> sr1;
    endproperty

    assert property (pr1) $display("%t: Pass",$time); else $display("%t: Fail",$time);



endmodule
```

C Code:

```
//specs: write a seuence s11b, after 2 +ve clock edges, signal b shold be low

module Q3_C ();

    sequence s11b;
        ##[2] (b==0);
    endsequence

    /*property pr1;
        @(posedge clk) |-> s11b;
    endproperty

    assert property (pr1) $display("%t: Pass",$time); else $display("%t: Fail",$time);
    */

endmodule
```

D-1 Code:

```
//specs: write a property: 3 to 8 decoder output Y
//       at each +ve edge of clock, Y must have only on bit high

module Q3_D_1 ();

    sequence Seq1;
        $onehot(Y);
    endsequence

    property pr1;
        @(posedge clk) |-> Seq1;
    endproperty

    assert property (pr1) $display("%t: Pass",$time); else $display("%t: Fail",$time);


endmodule

```

D-2 Code:

```
//specs: write a property: 4 to 2 priority encoder output valid
//       at each +ve edge of clock, if D bits are low then after one cycle output valid must be low

module Q3_D_2 ();

    sequence Seq1;
        ##1 (valid == 0);
    endsequence

    property pr1;
        @(posedge clk) (D = 4'b0000) |-> Seq1;
    endproperty

    assert property (pr1) $display("%t: Pass",$time); else $display("%t: Fail",$time);

endmodule

```