Verilog HDL

Assignment 2

By: Magdy Ahmed Abbas Abdelhamid

Question 1

Design has 5 inputs and 2 outputs.

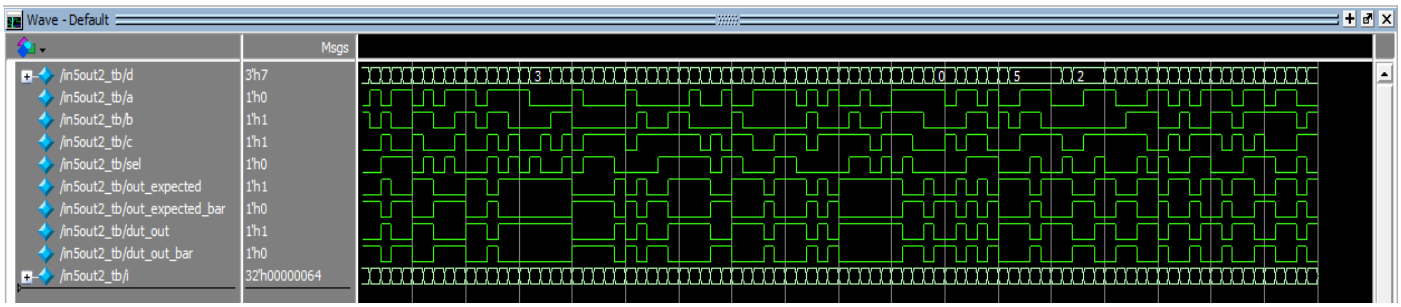
Verilog Code:

```
module in5out2 (D, A, B, C, Sel, Out, Out_Bar);

input [2: 0] D;
input A, B, C, Sel;

output reg Out, Out_Bar;

reg AND2, OR1, XNOR1, MUX1;

always @(*) begin
  AND2  =  D[1] & D[0];
  OR1  =  AND2 | D[2];
  XNOR1  =  ~(A ^ B ^C);

  if (Sel)
    Out  =  XNOR1;
  else
    Out  =  OR1;
  Out_Bar  =  ~Out;
end

endmodule
```

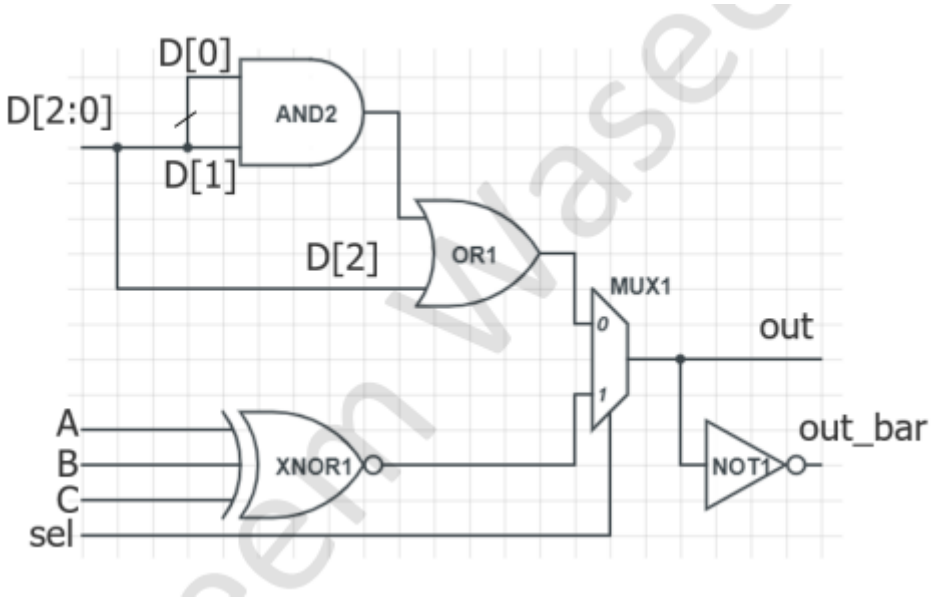
Testbench Code:

```
module in5out2_tb ();

reg [2: 0] d;
reg a, b, c, sel, out_expected, out_expected_bar;
wire dut_out, dut_out_bar;

in5out2 DUT (. D(d), . A(a), . B(b), . C(c), . Sel(sel), . Out(dut_out), . Out_Bar(dut_out_bar));

integer i;
initial begin
  for (i = 0 ; i < 100; i = i + 1) begin
    d = $random;
    a = $random;
    b = $random;
    c = $random;
    sel = $random;
    out_expected = (sel == 1)? ~(a ^ b ^ c) : (d[0] & d[1]) | d[2];
    out_expected_bar = ~out_expected;
    #10;
    if (out_expected ! = dut_out) begin
      $display("The Design is Wrong! ");
    end
    if (out_expected_bar ! = dut_out_bar) begin
      $display("The Design is Wrong! ");
    end
  end
  $stop;
end

initial begin
  $monitor("D0 = %b, D1 = %b, D2 = %b, A = %b, B = %b, C = %b, OUT = %b, OUT_BAR
            = %b ", d[0], d[1], d[2], a, b, c, dut_out, dut_out_bar);
end

endmodule
```

Waveform:




```
# D0 = 1, D1 = 1, D2 = 0, A = 0, B = 1, C = 1, OUT = 1, OUT_BAR = 0
# D0 = 0, D1 = 0, D2 = 1, A = 1, B = 1, C = 0, OUT = 1, OUT_BAR = 0
# D0 = 1, D1 = 0, D2 = 0, A = 1, B = 0, C = 0, OUT = 0, OUT_BAR = 1
# D0 = 1, D1 = 1, D2 = 0, A = 0, B = 0, C = 1, OUT = 1, OUT_BAR = 0
# D0 = 0, D1 = 0, D2 = 1, A = 1, B = 1, C = 0, OUT = 1, OUT_BAR = 0
# D0 = 1, D1 = 0, D2 = 0, A = 0, B = 0, C = 1, OUT = 0, OUT_BAR = 1
# D0 = 0, D1 = 1, D2 = 1, A = 1, B = 0, C = 1, OUT = 1, OUT_BAR = 0
# D0 = 0, D1 = 0, D2 = 1, A = 1, B = 1, C = 1, OUT = 1, OUT_BAR = 0
# D0 = 1, D1 = 0, D2 = 0, A = 1, B = 1, C = 0, OUT = 0, OUT_BAR = 1
# D0 = 1, D1 = 1, D2 = 0, A = 0, B = 0, C = 1, OUT = 1, OUT_BAR = 0
# D0 = 0, D1 = 0, D2 = 0, A = 1, B = 1, C = 0, OUT = 1, OUT_BAR = 0
# D0 = 0, D1 = 1, D2 = 1, A = 1, B = 1, C = 1, OUT = 0, OUT_BAR = 1
# D0 = 0, D1 = 0, D2 = 0, A = 0, B = 1, C = 0, OUT = 0, OUT_BAR = 1
# D0 = 0, D1 = 1, D2 = 1, A = 1, B = 1, C = 0, OUT = 1, OUT_BAR = 0
# D0 = 1, D1 = 1, D2 = 1, A = 1, B = 1, C = 0, OUT = 1, OUT_BAR = 0
# D0 = 1, D1 = 0, D2 = 1, A = 0, B = 0, C = 1, OUT = 0, OUT_BAR = 1
# D0 = 1, D1 = 1, D2 = 1, A = 0, B = 1, C = 1, OUT = 1, OUT_BAR = 0
```



Question 2

Design 4 - Bit Priority Encoder

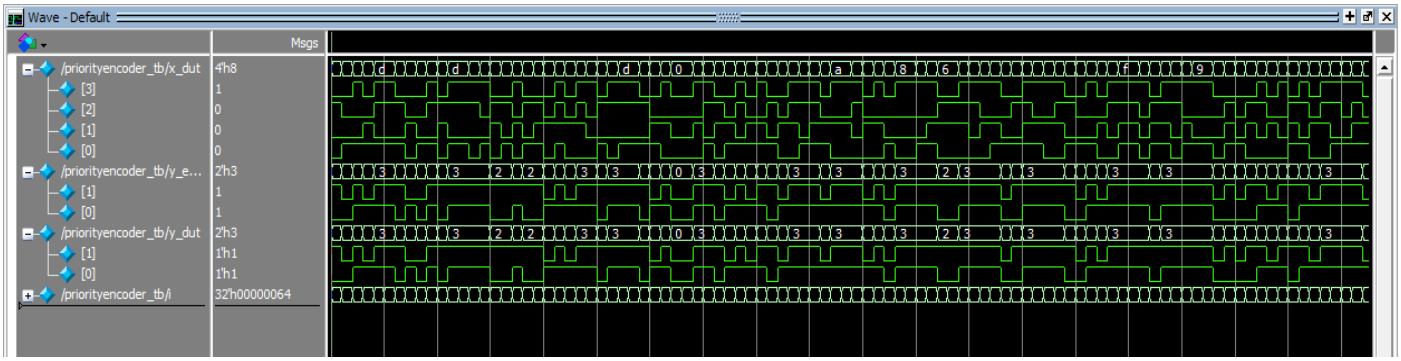
Verilog Code:

```
module priorityencoder (X, Y);

input [3: 0] X;
output reg [1: 0] Y;

always @(X) begin
   if (X[3])
      Y = 2'b11;
   else
      if (X[2])
         Y = 2'b10;
      else
         if (X[1])
            Y = 2'b01;
         else
            Y = 2'b00;
end

endmodule
/*
module priorityencoder (X, Y);

input [3: 0] X;
output reg [1: 0] Y;

always @(X) begin
   casex (X)
      4'b1xxx : Y = 2'b11;
      4'b01xx : Y = 2'b10;
      4'b001x : Y = 2'b01;
      4'b000x : Y = 2'b00;
   endcase
end
endmodule
*/
```

Testbench Code:

```
module priorityencoder_tb ();

reg [3:0] x_dut;
reg [1:0] y_expected;
wire [1:0] y_dut;

priorityencoder DUT (.X(x_dut), .Y(y_dut));

integer i;
initial begin
  for (i = 0; i < 100; i = i + 1) begin
    x_dut = $random;

   casex (x_dut)
    4'b1xxx : y_expected = 2'b11;
    4'b01xx : y_expected = 2'b10;
    4'b001x : y_expected = 2'b01;
    4'b000x : y_expected = 2'b00;
   endcase
    #10;
   if (y_dut != y_expected) begin
    $display("The Design is Wrong! ");
    $stop;
   end
  end
  $stop;
end

initial begin
  $monitor("X3 X2 X1 X0 = %b, Y1 Y0 = %b ", x_dut, y_dut);
end

endmodule
```

Waveform:




```
# X3 X2 X1 X0 = 0000, Y1 Y0 = 00
# X3 X2 X1 X0 = 0111, Y1 Y0 = 10
# X3 X2 X1 X0 = 0001, Y1 Y0 = 00
# X3 X2 X1 X0 = 0110, Y1 Y0 = 10
# X3 X2 X1 X0 = 1100, Y1 Y0 = 11
# X3 X2 X1 X0 = 0010, Y1 Y0 = 01
# X3 X2 X1 X0 = 1000, Y1 Y0 = 11
# X3 X2 X1 X0 = 0111, Y1 Y0 = 10
# X3 X2 X1 X0 = 1101, Y1 Y0 = 11
# X3 X2 X1 X0 = 0010, Y1 Y0 = 01
# X3 X2 X1 X0 = 1110, Y1 Y0 = 11
# X3 X2 X1 X0 = 1101, Y1 Y0 = 11
# X3 X2 X1 X0 = 1001, Y1 Y0 = 11
# X3 X2 X1 X0 = 1111, Y1 Y0 = 11
# X3 X2 X1 X0 = 0011, Y1 Y0 = 01
# X3 X2 X1 X0 = 0101, Y1 Y0 = 10
# X3 X2 X1 X0 = 1000, Y1 Y0 = 11
```

x3	x2	x1	x0	y1	y0
1	X	X	X	1	1
0	1	X	X	1	0
0	0	1	X	0	1
0	0	0	X	0	0

Question 3

Design a Decimal to BCD Encoder

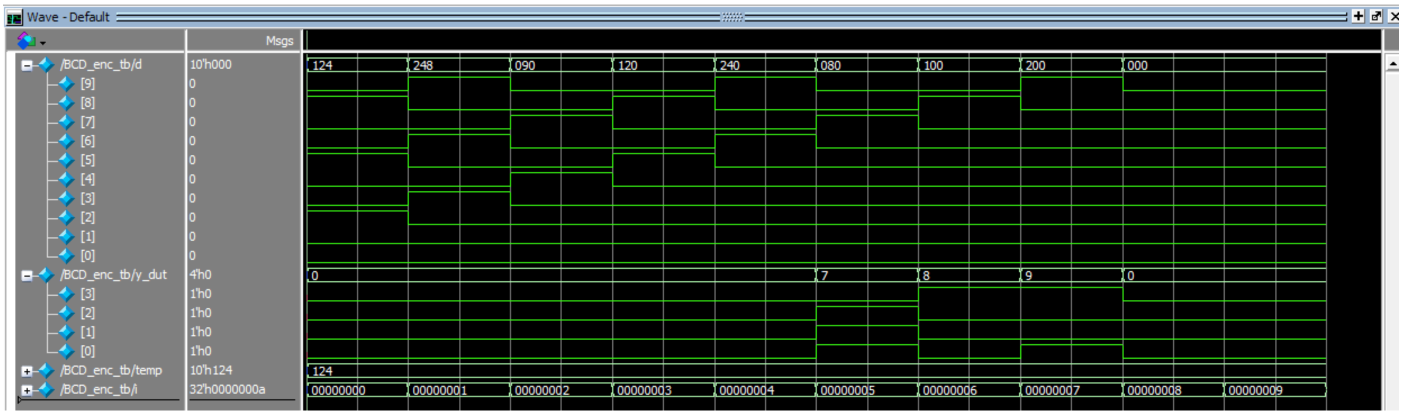
Verilog Code:

```
module BCD_enc (D, Y);

input [9: 0] D;
output reg [3: 0] Y;

always @(*) begin
  case (D)
    10'b0000000001 : Y = 4'b0000;
    10'b0000000010 : Y = 4'b0001;
    10'b0000000100 : Y = 4'b0010;
    10'b0000001000 : Y = 4'b0011;
    10'b0000010000 : Y = 4'b0100;
    10'b0000100000 : Y = 4'b0101;
    10'b0001000000 : Y = 4'b0110;
    10'b0010000000 : Y = 4'b0111;
    10'b0100000000 : Y = 4'b1000;
    10'b1000000000 : Y = 4'b1001;
    default : Y = 4'b0000;
  endcase
end

endmodule
```

Testbench Code:

```
module BCD_enc_tb ();

reg [9: 0] d;
wire [3: 0] y_dut;
BCD_enc DUT (. D(d), . Y(y_dut));
reg [9: 0]temp;



integer i;
initial begin
   d  =  $random;
   temp  =  d;
   for (i  =  0 ; i  <  10 ; i  =  i  +  1) begin
      d  =  temp  <<  i;
      #10;
   end
   $stop;
end

initial begin
   $monitor("D9 D8 D7 D6 D5 D4 D3 D2 D1 D0  =  %b,
                              Y2 Y1 Y0  =  %b ", d, y_dut);
end

endmodule
```

Waveform:




```
# D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 = 0100100100, Y2 Y1 Y0 = 0000
# D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 = 1001001000, Y2 Y1 Y0 = 0000
# D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 = 0010010000, Y2 Y1 Y0 = 0000
# D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 = 0100100000, Y2 Y1 Y0 = 0000
# D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 = 1001000000, Y2 Y1 Y0 = 0000
# D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 = 0010000000, Y2 Y1 Y0 = 0111
# D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 = 0100000000, Y2 Y1 Y0 = 1000
# D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 = 1000000000, Y2 Y1 Y0 = 1001
# D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 = 0000000000, Y2 Y1 Y0 = 0000
```

Input										Output			
D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	1

Question 4

N - Bit Adder Using Dataflow

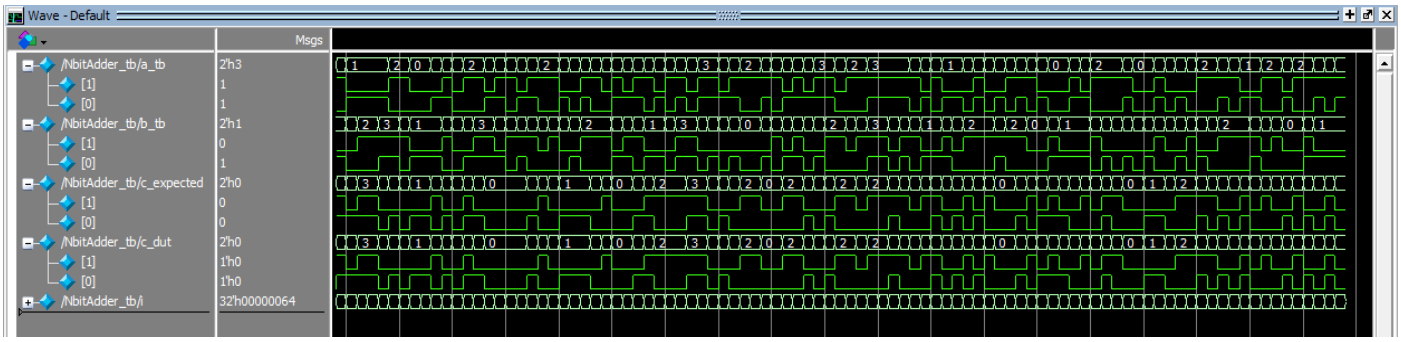
Verilog Code:

```
module NbitAdder (A, B, C);

parameter N = 1;
input [N - 1 : 0] A, B;
output [N - 1 : 0] C;

assign C = A + B;

endmodule
```

Testbench Code:

```
module NbitAdder_tb ();

parameter n = 2;
reg [n - 1 : 0] a_tb, b_tb, c_expected;
wire [n - 1 : 0] c_dut;

NbitAdder #(.N(n)) DUT (.A(a_tb), .B(b_tb), .C(c_dut));

integer i;
initial begin

  // Directed Testing
  #0 a_tb = 0; b_tb = 1; c_expected = 1;
  #10 a_tb = 1; b_tb = 2; c_expected = 3;
  #10 a_tb = 1; b_tb = 7; c_expected = 8;
  #10 a_tb = 5; b_tb = 4; c_expected = 9;
  #10;
  // Randomized Testing
  for (i = 0 ; i < 100 ; i = i + 1) begin
    a_tb = $random;
    b_tb = $random;
    c_expected = a_tb + b_tb;
    #10;
    if (c_dut != c_expected) begin
      $display("The Nbit Adder Design is Wrong! ");
      $stop;
    end
  end
  $stop;
end

initial begin
  $monitor("A = %b, B = %b, C_DUT = %b,
                        C_Expected = %b ", a_tb, b_tb, c_dut, c_expected);
end

endmodule
```

Waveform:




```
# A = 00, B = 10, C_DUT = 10, C_Expected = 10
# A = 10, B = 00, C_DUT = 10, C_Expected = 10
# A = 10, B = 10, C_DUT = 00, C_Expected = 00
# A = 11, B = 11, C_DUT = 10, C_Expected = 10
# A = 11, B = 00, C_DUT = 11, C_Expected = 11
# A = 11, B = 11, C_DUT = 10, C_Expected = 10
# A = 10, B = 10, C_DUT = 00, C_Expected = 00
# A = 01, B = 01, C_DUT = 10, C_Expected = 10
# A = 10, B = 01, C_DUT = 11, C_Expected = 11
# A = 01, B = 11, C_DUT = 00, C_Expected = 00
# A = 01, B = 00, C_DUT = 01, C_Expected = 01
# A = 00, B = 10, C_DUT = 10, C_Expected = 10
# A = 11, B = 10, C_DUT = 01, C_Expected = 01
# A = 00, B = 10, C_DUT = 10, C_Expected = 10
# A = 01, B = 11, C_DUT = 00, C_Expected = 00
# A = 10, B = 10, C_DUT = 00, C_Expected = 00
# A = 11, B = 10, C_DUT = 01, C_Expected = 01
# A = 10, B = 00, C_DUT = 10, C_Expected = 10
# A = 01, B = 00, C_DUT = 01, C_Expected = 01
# A = 00, B = 11, C_DUT = 11, C_Expected = 11
# A = 00, B = 01, C_DUT = 01, C_Expected = 01
# A = 11, B = 01, C_DUT = 00, C_Expected = 00
```

Question 5

Design N - Bit ALU

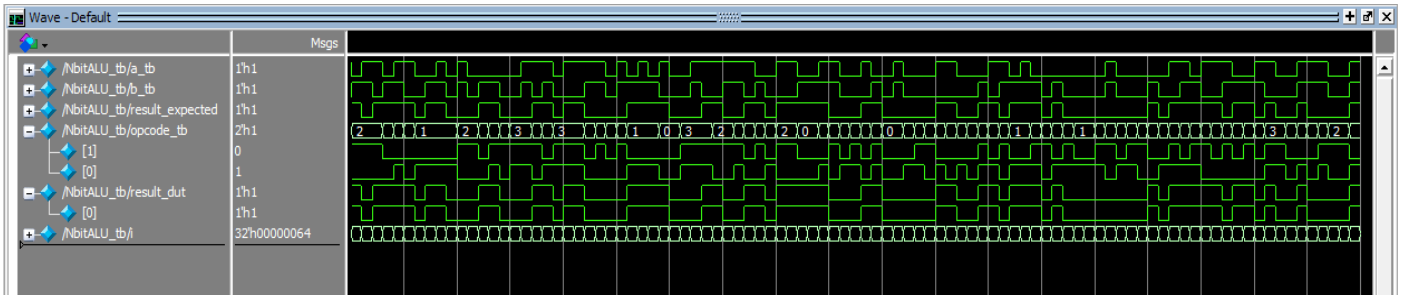
Verilog Code:

```
module NbitALU (A, B, OPCODE, RESULT);

parameter M = 1;
input [M - 1 : 0] A, B;
input [1 : 0] OPCODE;
output reg [M - 1 : 0] RESULT;

wire [M - 1 : 0] adder_out;

NbitAdder #(.N(M)) Addition(.A(A), .B(B), .C(adder_out));

always @(*) begin
  case (OPCODE)
    2'b00 : RESULT = adder_out;
    2'b10 : RESULT = A - B;
    2'b01 : RESULT = A | B;
    2'b11 : RESULT = A ^ B;
  endcase
end

endmodule
```

Testbench Code:

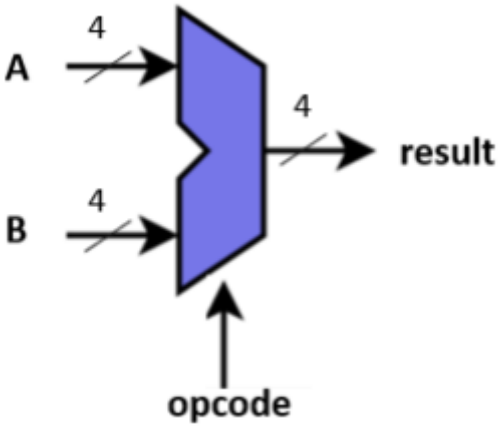
```
module NbitALU_tb ();

parameter n = 1;
reg [n - 1 : 0] a_tb, b_tb;
reg [n - 1 : 0] result_expected;
reg [1 : 0] opcode_tb;
wire [n - 1 : 0] result_dut;

NbitALU #(.M(n)) DUT
(.A(a_tb), .B(b_tb), .OPCODE(opcode_tb), .RESULT(result_dut));

integer i;
initial begin
  for (i = 0 ; i < 100 ; i = i + 1) begin
    a_tb = $random;
    b_tb = $random;
    opcode_tb = $random;
    case (opcode_tb)
      2'b00 : result_expected = a_tb + b_tb;
      2'b10 : result_expected = a_tb - b_tb;
      2'b01 : result_expected = a_tb | b_tb;
      2'b11 : result_expected = a_tb ^ b_tb;
    endcase
    #10;
    if (result_dut != result_expected) begin
      $display ("The ALU Design is Wrong! ");
      $stop;
    end
  end
  $stop;
End
initial begin
  $monitor("A = %b, B = %b, Opcode = %b,
                        Result = %b ", a_tb, b_tb, opcode_tb, result_dut);
end

endmodule
```

Waveform:




```
# A = 0, B = 0, Opcode = 10, Result = 0
# A = 0, B = 1, Opcode = 00, Result = 1
# A = 0, B = 0, Opcode = 01, Result = 0
# A = 1, B = 0, Opcode = 11, Result = 1
# A = 1, B = 1, Opcode = 01, Result = 1
# A = 0, B = 1, Opcode = 11, Result = 1
# A = 1, B = 0, Opcode = 10, Result = 1
# A = 1, B = 0, Opcode = 00, Result = 1
# A = 1, B = 1, Opcode = 10, Result = 0
# A = 0, B = 1, Opcode = 00, Result = 1
# A = 0, B = 1, Opcode = 10, Result = 1
# A = 1, B = 1, Opcode = 00, Result = 0
# A = 1, B = 0, Opcode = 11, Result = 1
# A = 0, B = 0, Opcode = 11, Result = 0
# A = 0, B = 0, Opcode = 00, Result = 0
# A = 1, B = 1, Opcode = 01, Result = 1
# A = 1, B = 1, Opcode = 00, Result = 0
# A = 1, B = 1, Opcode = 11, Result = 0
# A = 0, B = 0, Opcode = 10, Result = 0
# A = 1, B = 1, Opcode = 01, Result = 1
```

Inputs		Outputs
opcode		Operation
0	0	Addition
1	0	Subtraction
0	1	OR
1	1	XOR



Question 6

4 - Bit ALU Display On 7 Segment LED Display

Verilog Code:

```
module ALU4bit7SegmentLED (A, B, Opcode, enable, a, b, c, d, e, f, g);
parameter N = 1;
input [N - 1 : 0] A, B;
input [1: 0] Opcode;
input enable;
output reg a, b, c, d, e, f, g;
wire [N - 1 : 0] Result;
reg [6: 0] LEDs;
NbitALU #(.M(N)) ALU(.A(A), .B(B), .OPCODE(Opcode), .RESULT(Result));
always @(*) begin
  if (enable) begin
    case (Result)
      4'h0 : LEDs = 7'b111_1110; // LEDS = {a, b, c, d, e, f, g}
      4'h1 : LEDs = 7'b011_0000;
      4'h2 : LEDs = 7'b110_1101;
      4'h3 : LEDs = 7'b111_1001;
      4'h4 : LEDs = 7'b011_0011;
      4'h5 : LEDs = 7'b101_1011;
      4'h6 : LEDs = 7'b101_1111;
      4'h7 : LEDs = 7'b111_0000;
      4'h8 : LEDs = 7'b111_1111;
      4'h9 : LEDs = 7'b111_1011;
      4'hA : LEDs = 7'b111_0111;
      4'hB : LEDs = 7'b001_1111;
      4'hC : LEDs = 7'b100_1110;
      4'hD : LEDs = 7'b011_1101;
      4'hE : LEDs = 7'b100_1111;
      4'hF : LEDs = 7'b100_0111;
    endcase
  end  else    LEDs = 7'b000_0000;
end
always @(*) begin
  a = LEDs[6];
  b = LEDs[5];
  c = LEDs[4];
  d = LEDs[3];
  e = LEDs[2];
  f = LEDs[1];
  g = LEDs[0];
end endmodule
```

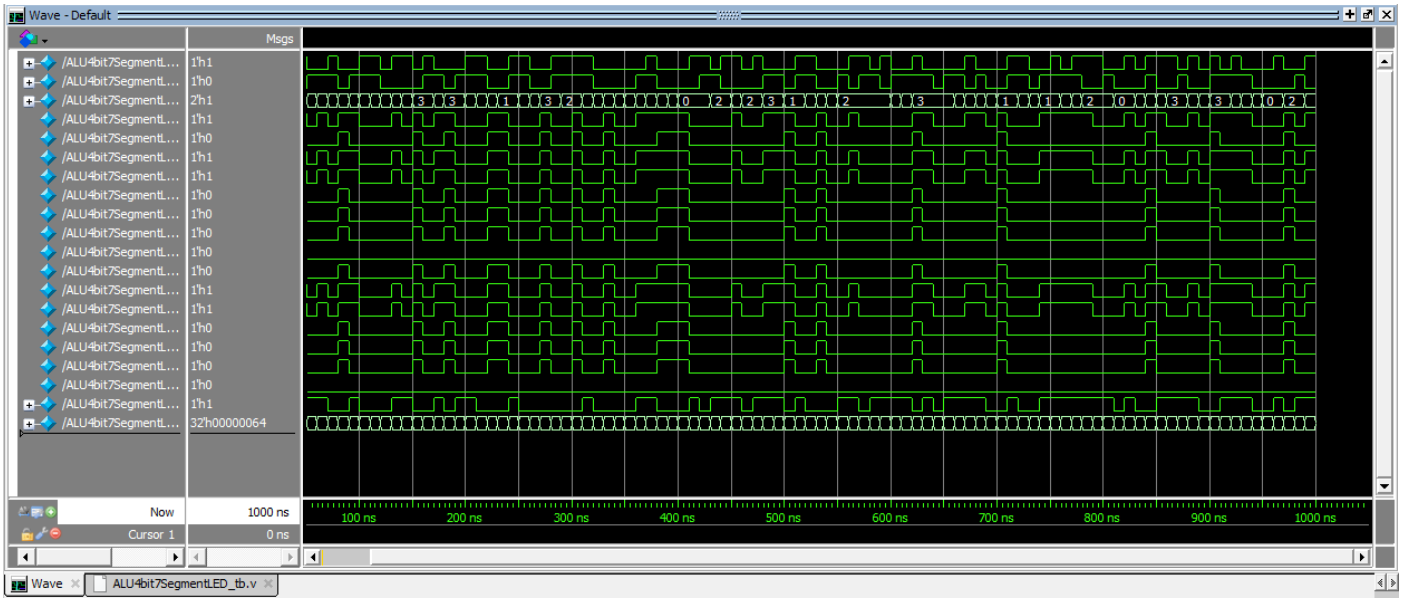
Testbench Code:

```
module ALU4bit7SegmentLED_tb ();

parameter n = 1;
reg [n - 1 : 0] A_tb, B_tb;
reg [1: 0] Op_code;
reg Enable;
wire a_tb, b_tb, c_tb, d_tb, e_tb, f_tb, g_tb;
reg a_tb_Exp, b_tb_Exp, c_tb_Exp, d_tb_Exp, e_tb_Exp, f_tb_Exp, g_tb_Exp;
reg [n - 1 : 0] Result_Expected;

ALU4bit7SegmentLED #(.N(n)) DUT (A_tb, B_tb, Op_code, Enable, a_tb, b_tb, c_tb, d_tb, e_tb, f_tb, g_tb);

integer i;
initial begin
  for (i = 0 ; i < 100 ; i = i + 1) begin
    A_tb = $random;
    B_tb = $random;
    Op_code = $random;
    Enable = $random;
    case (Op_code)
      2'b00 : Result_Expected = A_tb + B_tb;
      2'b10 : Result_Expected = A_tb - B_tb;
      2'b01 : Result_Expected = A_tb | B_tb;
      2'b11 : Result_Expected = A_tb ^ B_tb;
    endcase
    if (Enable) begin
      case (Result_Expected)
        // {a_tb_Exp, b_tb_Exp, c_tb_Exp, d_tb_Exp, e_tb_Exp, f_tb_Exp, g_tb_Exp} = {a, b, c, d, e, f, g}
        4'h0 : {a_tb_Exp, b_tb_Exp, c_tb_Exp, d_tb_Exp, e_tb_Exp, f_tb_Exp, g_tb_Exp} = 7'b111_1110;
        4'h1 : {a_tb_Exp, b_tb_Exp, c_tb_Exp, d_tb_Exp, e_tb_Exp, f_tb_Exp, g_tb_Exp} = 7'b011_0000;
        4'h2 : {a_tb_Exp, b_tb_Exp, c_tb_Exp, d_tb_Exp, e_tb_Exp, f_tb_Exp, g_tb_Exp} = 7'b110_1101;
        4'h3 : {a_tb_Exp, b_tb_Exp, c_tb_Exp, d_tb_Exp, e_tb_Exp, f_tb_Exp, g_tb_Exp} = 7'b111_1001;
        4'h4 : {a_tb_Exp, b_tb_Exp, c_tb_Exp, d_tb_Exp, e_tb_Exp, f_tb_Exp, g_tb_Exp} = 7'b011_0011;
        4'h5 : {a_tb_Exp, b_tb_Exp, c_tb_Exp, d_tb_Exp, e_tb_Exp, f_tb_Exp, g_tb_Exp} = 7'b101_1011;
        4'h6 : {a_tb_Exp, b_tb_Exp, c_tb_Exp, d_tb_Exp, e_tb_Exp, f_tb_Exp, g_tb_Exp} = 7'b101_1111;
        4'h7 : {a_tb_Exp, b_tb_Exp, c_tb_Exp, d_tb_Exp, e_tb_Exp, f_tb_Exp, g_tb_Exp} = 7'b111_0000;
        4'h8 : {a_tb_Exp, b_tb_Exp, c_tb_Exp, d_tb_Exp, e_tb_Exp, f_tb_Exp, g_tb_Exp} = 7'b111_1111;
        4'h9 : {a_tb_Exp, b_tb_Exp, c_tb_Exp, d_tb_Exp, e_tb_Exp, f_tb_Exp, g_tb_Exp} = 7'b111_1011;
        4'hA : {a_tb_Exp, b_tb_Exp, c_tb_Exp, d_tb_Exp, e_tb_Exp, f_tb_Exp, g_tb_Exp} = 7'b111_0111;
        4'hB : {a_tb_Exp, b_tb_Exp, c_tb_Exp, d_tb_Exp, e_tb_Exp, f_tb_Exp, g_tb_Exp} = 7'b001_1111;
        4'hC : {a_tb_Exp, b_tb_Exp, c_tb_Exp, d_tb_Exp, e_tb_Exp, f_tb_Exp, g_tb_Exp} = 7'b100_1110;
        4'hD : {a_tb_Exp, b_tb_Exp, c_tb_Exp, d_tb_Exp, e_tb_Exp, f_tb_Exp, g_tb_Exp} = 7'b011_1101;
        4'hE : {a_tb_Exp, b_tb_Exp, c_tb_Exp, d_tb_Exp, e_tb_Exp, f_tb_Exp, g_tb_Exp} = 7'b100_1111;
        4'hF : {a_tb_Exp, b_tb_Exp, c_tb_Exp, d_tb_Exp, e_tb_Exp, f_tb_Exp, g_tb_Exp} = 7'b100_0111;
      endcase
  end
```

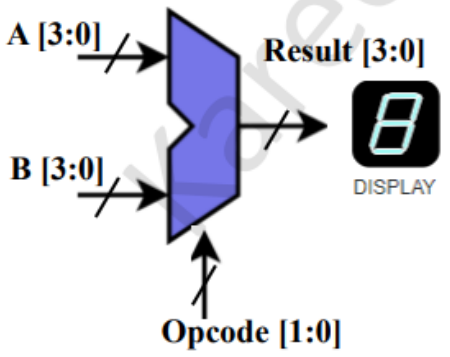
```
  else
    {a_tb_Exp, b_tb_Exp, c_tb_Exp, d_tb_Exp, e_tb_Exp, f_tb_Exp, g_tb_Exp} = 7'b000_0000;
  #10;
  if ({a_tb_Exp, b_tb_Exp, c_tb_Exp, d_tb_Exp, e_tb_Exp, f_tb_Exp, g_tb_Exp} !
                = {a_tb, b_tb, c_tb, d_tb, e_tb, f_tb, g_tb}) begin
    $display("The ALU 4 bit 7 Segment LEDs Design is Wrong! ");
    $stop;
  end
  end
  $stop;
end

initial begin
  $monitor("Digit = %d, abcdefg = %b ", Result_Expected, {a_tb, b_tb, c_tb, d_tb, e_tb, f_tb, g_tb});
end

endmodule
```

	Input	Output						
Digit	enable	a	b	c	d	e	f	g
0	1	1	1	1	1	1	1	0
1	1	0	1	1	0	0	0	0
2	1	1	1	0	1	1	0	1
3	1	1	1	1	1	0	0	1
4	1	0	1	1	0	0	1	1
5	1	1	0	1	1	0	1	1
6	1	1	0	1	1	1	1	1
7	1	1	1	1	0	0	0	0
8	1	1	1	1	1	1	1	1
9	1	1	1	1	1	0	1	1
A	1	1	1	1	0	1	1	1
b	1	0	0	1	1	1	1	1
C	1	1	0	0	1	1	1	0
d	1	0	1	1	1	1	0	1
E	1	1	0	0	1	1	1	1
F	1	1	0	0	0	1	1	1
x	0	0	0	0	0	0	0	0

Waveform:




```
# Digit = 1, abcdefg = 0000000
# Digit = 1, abcdefg = 0110000
# Digit = 0, abcdefg = 0000000
# Digit = 0, abcdefg = 1111110
# Digit = 1, abcdefg = 0000000
# Digit = 0, abcdefg = 0000000
# Digit = 1, abcdefg = 0110000
# Digit = 1, abcdefg = 0000000
# Digit = 0, abcdefg = 0000000
# Digit = 1, abcdefg = 0110000
# Digit = 0, abcdefg = 0000000
# Digit = 0, abcdefg = 1111110
# Digit = 1, abcdefg = 0110000
# Digit = 1, abcdefg = 0000000
# Digit = 1, abcdefg = 0110000
# Digit = 0, abcdefg = 0000000
# Digit = 0, abcdefg = 1111110
# Digit = 1, abcdefg = 0110000
# Digit = 0, abcdefg = 0000000
# Digit = 1, abcdefg = 0000000
# Digit = 0, abcdefg = 1111110
# Digit = 1, abcdefg = 0000000
```



7-segment decoder