

## Assignment 1

1)

Truth tables:

GEL/AB	00	01	11	10
001	1	1	1	1
010	0	1	0	0
100	0	0	0	0

$$L = G'E'L + G'(XOR(E, L))A'B$$

GEL/AB	00	01	11	10
001	0	0	0	0
010	1	0	1	0
100	0	0	0	0

$$E = (A \text{ xnor } B)(G'EL')$$

GEL/AB	00	01	11	10
001	0	0	0	0
010	0	0	0	1
100	1	1	1	1

$$\begin{aligned} G &= GE'L' + AB'(G'EL') \\ &= GE'L' + G'EL'AB' \end{aligned}$$

Code:

```
// Filename: Vincent-Li-A1-Q1-iterMagComp.v
// Name: Vincent Li
// Course: CSE320 T/TH 4:30PM
// Professor: Vrudhula
/*
    Structural verilog module for a two-input, n-bit comparator.
    Design should be parameterized in terms of the
    word sizes of the operands. Test for n = 4.
*/

`include "gscl45nm.v"

// iterMagComp for 1 bit
module iterMagComp_1 (
    // input and output wires
    input wire a, b,
```

```

input wire pL, pE, pG,
output wire nL, nE, nG
);

wire g_bar, e_bar, l_bar, a_bar, b_bar;
wire w1, w2, w3, w4, w5, w6, w7;
// nots
not n1(g_bar, pG);
not n2(e_bar, pE);
not n3(l_bar, pL);
not n4(a_bar, a);
// for L = g_out[0]
and a1(w1, e_bar, pL);
xor xo1(w2, pE, pL);
and a2(w3, w2, a_bar, b);
or o1(w4, w1, w3);
and a3(nL, g_bar, w4);
// for E = g_out[1]
xnor xn1(w5, a, b);
and a4(nE, w5, g_bar, pE, l_bar);
// for G = g_out[2]
and a5(w6, pG, e_bar, l_bar);
and a6(w7, g_bar, pE, l_bar, a, b_bar);
or o2(nG, w6, w7);

endmodule

module iterMagComp #(parameter integer WIDTH = 4) (
    // input and output wires
    input wire [WIDTH - 1 : 0] a, b,
    input wire [2 : 0] in,
    output wire [2 : 0] out
);
    // main input and output per module
    wire [WIDTH - 1 : 0] pL, pE, pG, nL, nE, nG;

    // instantiate 1-bit modules
    iterMagComp_1 M[WIDTH - 1 : 0] (.a(a), .b(b), .pL(pL), .pE(pE), .pG(pG),
    .nL(nL), .nE(nE), .nG(nG));
    // Input to iterMagComp[n-1]
    assign pL[WIDTH - 1] = in[0];
    assign pE[WIDTH - 1] = in[1];
    assign pG[WIDTH - 1] = in[2];
    // Connecting the output signal to cell 0
    assign out[0] = nL[0];

```

```

    assign out[1] = nE[0];
    assign out[2] = nG[0];
    // Connecting the intermediate signals
    assign pL[WIDTH - 2 : 0] = nL[WIDTH - 1 : 1];
    assign pE[WIDTH - 2 : 0] = nE[WIDTH - 1 : 1];
    assign pG[WIDTH - 2 : 0] = nG[WIDTH - 1 : 1];

endmodule

```

```

// Filename: Vincent-Li-A1-Q1-iterMagComp_tb.v
// Name: Vincent Li
// Course: CSE320 T/TH 4:30PM
// Professor: Vrudhula
/*
    Test bench for Vincent-Li-A1-Q1-iterMagComp.v
    Prints "All input combinations successfully verified"
        after checking the actual results with the expected ones.
*/

`include "gscl45nm.v"

module iterMagComp_tb;
    parameter integer WIDTH = 4, REPS = 2;
    reg [WIDTH - 1 : 0] a, b;
    reg [2 : 0] in;
    wire [2 : 0] out;
    reg [2 : 0] expected;
    integer allCorrect = 1;

    // instantiate
    iterMagComp #(.WIDTH(WIDTH)) M(.a(a), .b(b), .in(in), .out(out));

    initial
        begin
            in = 3'b010;
            // reset expected
            expected = 0;
            // reset a
            a = 0;
            #1;

            // loops to evaluate IMC and compare output to expected
            repeat(REPS)
                begin

```

```

        // reset b
        b = 0;
        #1;
        repeat(WIDTH * WIDTH)
            begin
                // find expected
                if(a < b) expected = 3'b001;
                else if(a == b) expected = 3'b010;
                else expected = 3'b100;

                // check
                /*
                if(expected != out) begin
                    #1 $display("a = %b, b = %b, out = %b, expected =
%b, equal = %b", a, b, out, expected, out == expected);
                    allCorrect = 0;
                end
                */
                // increment b
                b = b + 1;
                #1;
            end

        // increment a
        a = a + 1;
        in = 3'b010;
        #1;
    end

end
initial
begin
    if(allCorrect == 1) begin
        $display("All input combinations successfully verified");
    end
end
endmodule

```

Results:

```
Select Command Prompt
a = 0000, b = 1111, out = 001, expected = 001, equal = 1
a = 0001, b = 0000, out = x00, expected = 100, equal = x
a = 0001, b = 0001, out = x10, expected = 010, equal = 1
a = 0001, b = 0010, out = 001, expected = 001, equal = 1
a = 0001, b = 0011, out = 001, expected = 001, equal = 1
a = 0001, b = 0100, out = 001, expected = 001, equal = 1
a = 0001, b = 0101, out = 001, expected = 001, equal = 1
a = 0001, b = 0110, out = 001, expected = 001, equal = 1
a = 0001, b = 0111, out = 001, expected = 001, equal = 1
a = 0001, b = 1000, out = 001, expected = 001, equal = 1
a = 0001, b = 1001, out = 001, expected = 001, equal = 1
a = 0001, b = 1010, out = 001, expected = 001, equal = 1
a = 0001, b = 1011, out = 001, expected = 001, equal = 1
a = 0001, b = 1100, out = 001, expected = 001, equal = 1
a = 0001, b = 1101, out = 001, expected = 001, equal = 1
a = 0001, b = 1110, out = 001, expected = 001, equal = 1
a = 0001, b = 1111, out = 001, expected = 001, equal = 1

C:\Users\Livin\Documents\CSE320 Files\Vincent-Li-A1>iverilog -g specify -T typ -o iterMagComp Vincent-Li-A1-Q1-iterMagCo
mp_tb.v Vincent-Li-A1-Q1-iterMagComp.v

C:\Users\Livin\Documents\CSE320 Files\Vincent-Li-A1>vvp iterMagComp

C:\Users\Livin\Documents\CSE320 Files\Vincent-Li-A1>iverilog -g specify -T typ -o iterMagComp Vincent-Li-A1-Q1-iterMagCo
mp_tb.v Vincent-Li-A1-Q1-iterMagComp.v

C:\Users\Livin\Documents\CSE320 Files\Vincent-Li-A1>vvp iterMagComp
All input combinations successfully verified

C:\Users\Livin\Documents\CSE320 Files\Vincent-Li-A1>
```

2)

Equations:

- $sum = c\_in \&\& XOR(a XOR b)$
- $c\_out = a \&\& b \ || \ c\_in \&\& (a XOR b)$

Code:

```
// Filename: Vincent-Li-A1-Q2-rca.v
// Name: Vincent Li
// Course: CSE320 T/TH 4:30PM
// Professor: Vrudhula
/*
    Ripple carry adder using n full adders. 3 inputs and 2 inputs.
    Uses the parameter feature to make it general for any input size n.
*/

`include "gsc145nm.v"

module rca_1 (
    input wire a, b, c_in,
    output wire sum, c_out
);
    wire AxorB, w1, w2;
    // for sum
    XOR2X1 xo1(a, AxorB, b);
```

```

    XOR2X1 x02(c_in, sum, AxorB);
    // for c_out
    AND2X1 a1(c_in, AxorB, w1);
    AND2X1 a2(a, b, w2);
    OR2X1 o1(w1, w2, c_out);

endmodule

module rca #(parameter integer WIDTH = 4) (
    output wire [WIDTH - 1 : 0] sum,
    output wire out,
    input wire [WIDTH - 1 : 0] a, b,
    input wire in
);

    wire [WIDTH - 1 : 0] c_in, c_out;
    wire [WIDTH - 1 : 0] s;

    rca_1 M[WIDTH - 1 : 0] (.a(a), .b(b), .c_in(c_in), .sum(s), .c_out(c_out));
    // connecting wires
    // for c_in and c_out
    assign c_in[0] = in;
    assign out = c_out[WIDTH - 1];
    assign c_in[WIDTH - 1 : 1] = c_out[WIDTH - 2 : 0];
    // for sum
    assign sum[WIDTH - 1 : 0] = s[WIDTH - 1 : 0];

endmodule

```

```

// Filename: Vincent-Li-A1-Q2-rca_tb.v
// Name: Vincent Li
// Course: CSE320 T/TH 4:30PM
// Professor: Vrudhula
/*
    Test bench for rca.v
*/

module rca_tb;
    parameter integer WIDTH = 4;
    integer reps = 1;

    reg [WIDTH - 1 : 0] a, b;
    reg in;
    wire [WIDTH - 1 : 0] sum;

```

```

wire out;
reg [WIDTH : 0] expectedSum;
integer allCorrect = 1;

// instantiate rca
rca #(.WIDTH(WIDTH)) Mn(.a(a), .b(b), .in(in), .sum(sum), .out(out));

initial
    begin
        // initial input values
        a = 0;
        b = 0;
        in = 0;

        // find value for REPS
        repeat(WIDTH) begin
            reps = reps * 2;
        end

        // main loops
        repeat(reps) begin // modify a
            repeat(reps) begin // modify b
                repeat(2) begin // modify in
                    expectedSum = a + b + in;

                    // #1 $display("a = %b\tb = %b\tin = %b\tsum = %b\tout =
                    %b\texpected sum = %b\texpected out = %b", a, b, in, sum, out, expectedSum[WIDTH
                    - 1 : 0], expectedSum[WIDTH]);

                    if(expectedSum[WIDTH - 1 : 0] != sum ||
expectedSum[WIDTH] != out) begin
                        allCorrect = 0;

                        end
                        in = in + 1;

                    end
                    in = 0;
                    b = b + 1;

                end
                b = 0;
                a = a + 1;

            end
        end
    end
end

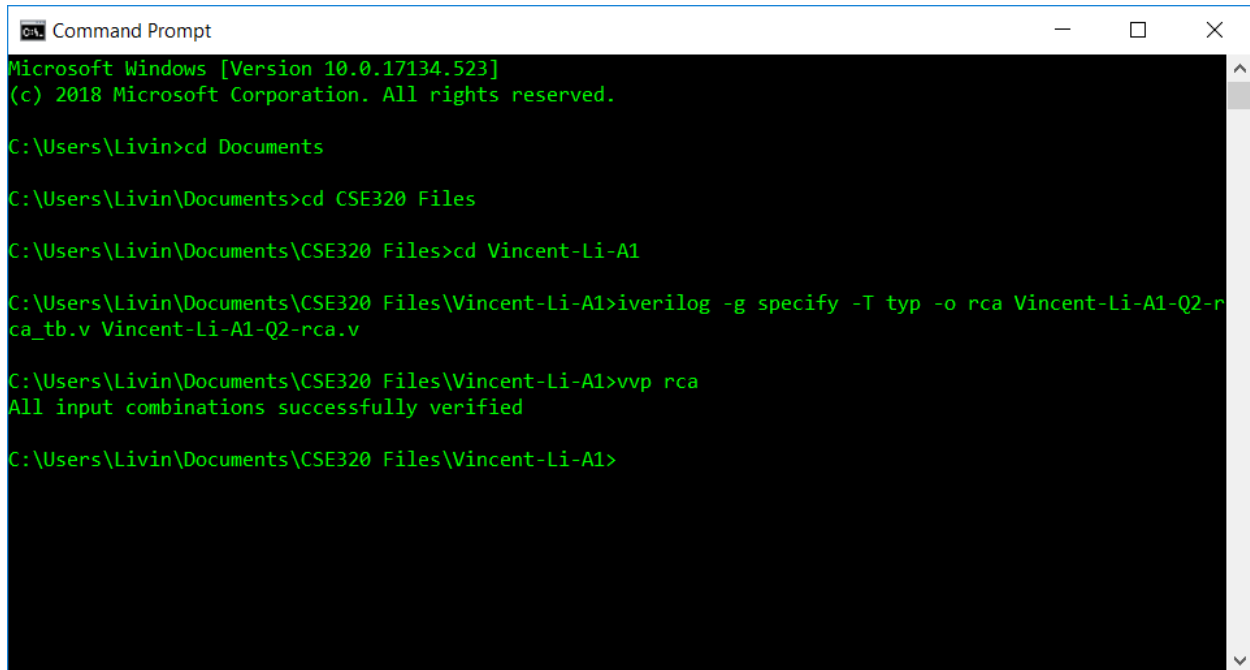
```

```

        // check allCorrect
        if(allCorrect == 1) begin
            $display("All input combinations successfully verified");
        end
    end
endmodule

```

Results:



```

C:\> Command Prompt
Microsoft Windows [Version 10.0.17134.523]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Livin>cd Documents

C:\Users\Livin\Documents>cd CSE320 Files

C:\Users\Livin\Documents\CSE320 Files>cd Vincent-Li-A1

C:\Users\Livin\Documents\CSE320 Files\Vincent-Li-A1>iverilog -g specify -T typ -o rca Vincent-Li-A1-Q2-rca_tb.v Vincent-Li-A1-Q2-rca.v

C:\Users\Livin\Documents\CSE320 Files\Vincent-Li-A1>vvp rca
All input combinations successfully verified

C:\Users\Livin\Documents\CSE320 Files\Vincent-Li-A1>

```

3b)

Equation:

$Y[n] = nE1 \mid \mid (\text{enabled}) \ \&\& \ !(\text{decimal value of A})$

Note: When nE1 is 1, outputs are 1. Otherwise, output is 0 at the location defined by A

Code:

```

// Filename: Vincent-Li-A1-Q3b-dec3x8.v
// Name: Vincent Li
// Course: CSE320 T/TH 4:30PM
// Professor: Vrudhula
/*
    Behavioral verilog description of a 3x8 decoder with 3 controls,
    E1, E2, and E3, and 3 inputs, A2, A1, A0.
    Every output will be high unless E1 and E2 are low and E3 is high.
*/

```



```

    When the decoder is disabled, all outputs are 1.
    When enabled, only the i'th output is active (0)
    and all other outputs are 1.
*/
module dec3x8 (
    input wire nE1, nE2, E3,
    input wire [2 : 0] A,
    output wire [7 : 0] nY
);
    wire enable;

    assign enable = !nE1 && !nE2 && E3;

    assign nY[0] = nE1 || enable && !(A[2] && !A[1] && !A[0]);
    assign nY[1] = nE1 || enable && !(A[2] && !A[1] && A[0]);
    assign nY[2] = nE1 || enable && !(A[2] && A[1] && !A[0]);
    assign nY[3] = nE1 || enable && !(A[2] && A[1] && A[0]);
    assign nY[4] = nE1 || enable && !A[2] && !A[1] && !A[0];
    assign nY[5] = nE1 || enable && !A[2] && !A[1] && A[0];
    assign nY[6] = nE1 || enable && !A[2] && A[1] && !A[0];
    assign nY[7] = nE1 || enable && !A[2] && A[1] && A[0];

endmodule

```

```

// Filename: Vincent-Li-A1-Q3b-dec3x8.v
// Name: Vincent Li
// Course: CSE320 T/TH 4:30PM
// Professor: Vrudhula
/*
    Test bench for Vincent-Li-A1-Q1-dec3x8.v
*/
module dec3x8_tb;
    reg nE1, nE2, E3;
    reg [2 : 0] A;
    wire [7 : 0] nY;
    integer allCorrect = 1;

    dec3x8 M(.nE1(nE1), .nE2(nE2), .E3(E3), .A(A), .nY(nY));

    initial
        begin
            // nE1 = 1, nE2 = X, E3 = X
            nE1 = 1;
            #1;

```

```

        // $display("nE1 = %b\t nE2 = %b\t E3 = %b\t A2 = %b\t A1 = %b\t A0 =
        %b\t\t nY7 = %b\t nY6 = %b\t nY5 = %b\t nY4 = %b\t nY3 = %b\t nY2 = %b\t nY1 = %b\t nY0 =
        %b", nE1, nE2, E3, A[2], A[1], A[0], nY[7], nY[6], nY[5], nY[4], nY[3], nY[2],
        nY[1], nY[0]);
        if(nY[7] != 1 || nY[6] != 1 || nY[5] != 1 || nY[4] != 1 || nY[3] != 1
        || nY[2] != 1 || nY[1] != 1 || nY[0] != 1) begin
            allCorrect = 0;
        end

        // nE1 = X, nE2 = 1, E3 = X
        nE2 = 1;
        #1;
        // $display("nE1 = %b\t nE2 = %b\t E3 = %b\t A2 = %b\t A1 = %b\t A0 =
        %b\t\t nY7 = %b\t nY6 = %b\t nY5 = %b\t nY4 = %b\t nY3 = %b\t nY2 = %b\t nY1 = %b\t nY0 =
        %b", nE1, nE2, E3, A[2], A[1], A[0], nY[7], nY[6], nY[5], nY[4], nY[3], nY[2],
        nY[1], nY[0]);
        if(nY[7] != 1 || nY[6] != 1 || nY[5] != 1 || nY[4] != 1 || nY[3] != 1
        || nY[2] != 1 || nY[1] != 1 || nY[0] != 1) begin
            allCorrect = 0;
        end

        // nE1 = X, nE2 = X, E3 = 0
        E3 = 0;
        #1;
        // $display("nE1 = %b\t nE2 = %b\t E3 = %b\t A2 = %b\t A1 = %b\t A0 =
        %b\t\t nY7 = %b\t nY6 = %b\t nY5 = %b\t nY4 = %b\t nY3 = %b\t nY2 = %b\t nY1 = %b\t nY0 =
        %b", nE1, nE2, E3, A[2], A[1], A[0], nY[7], nY[6], nY[5], nY[4], nY[3], nY[2],
        nY[1], nY[0]);
        if(nY[7] != 1 || nY[6] != 1 || nY[5] != 1 || nY[4] != 1 || nY[3] != 1
        || nY[2] != 1 || nY[1] != 1 || nY[0] != 1) begin
            allCorrect = 0;
        end

        // increment A
        #1 nE1 = 0;
        #1 nE2 = 0;
        #1 E3 = 1;
        #1 A = 0;
        repeat(8) begin
            #1;
            // $display("nE1 = %b\t nE2 = %b\t E3 = %b\t A2 = %b\t A1 = %b\t A0 =
            %b\t\t nY7 = %b\t nY6 = %b\t nY5 = %b\t nY4 = %b\t nY3 = %b\t nY2 = %b\t nY1 = %b\t nY0 =
            %b", nE1, nE2, E3, A[2], A[1], A[0], nY[7], nY[6], nY[5], nY[4], nY[3], nY[2],
            nY[1], nY[0]);
            if(nY[A] != 0) begin

```

```

        allCorrect = 0;
    end
    A = A + 1;

end
end

initial
begin
    if(allCorrect == 1) begin
        $display("Actual outputs equal expected outputs for all input
combinations.");
    end
end
endmodule

```

Results:

```

C:\Users\Livin\Documents\CSE320 Files\Vincent-Li-A1>iverilog -o dec3x8 Vincent-Li-A1-Q3b-dec3x8_tb.v Vincent-Li-A1-Q3b-dec3x8.v

C:\Users\Livin\Documents\CSE320 Files\Vincent-Li-A1>vvp dec3x8
nE1 = 0 nE2 = 0 E3 = 1 A2 = 0 A1 = 0 A0 = 0      nY7 = 1 nY6 = 1 nY5 = 1 nY4 = 1 nY3 = 1 nY2 = 1 nY1 = 1 nY0 = 0
nE1 = 0 nE2 = 0 E3 = 1 A2 = 0 A1 = 0 A0 = 1      nY7 = 1 nY6 = 1 nY5 = 1 nY4 = 1 nY3 = 1 nY2 = 1 nY1 = 0 nY0 = 1
nE1 = 0 nE2 = 0 E3 = 1 A2 = 0 A1 = 1 A0 = 0      nY7 = 1 nY6 = 1 nY5 = 1 nY4 = 1 nY3 = 1 nY2 = 0 nY1 = 1 nY0 = 1
nE1 = 0 nE2 = 0 E3 = 1 A2 = 0 A1 = 1 A0 = 1      nY7 = 1 nY6 = 1 nY5 = 1 nY4 = 1 nY3 = 0 nY2 = 1 nY1 = 1 nY0 = 1
nE1 = 0 nE2 = 0 E3 = 1 A2 = 1 A1 = 0 A0 = 0      nY7 = 1 nY6 = 1 nY5 = 1 nY4 = 0 nY3 = 1 nY2 = 1 nY1 = 1 nY0 = 1
nE1 = 0 nE2 = 0 E3 = 1 A2 = 1 A1 = 0 A0 = 1      nY7 = 1 nY6 = 1 nY5 = 0 nY4 = 1 nY3 = 1 nY2 = 1 nY1 = 1 nY0 = 1
nE1 = 0 nE2 = 0 E3 = 1 A2 = 1 A1 = 1 A0 = 0      nY7 = 1 nY6 = 0 nY5 = 1 nY4 = 1 nY3 = 1 nY2 = 1 nY1 = 1 nY0 = 1
nE1 = 0 nE2 = 0 E3 = 1 A2 = 1 A1 = 1 A0 = 1      nY7 = 0 nY6 = 1 nY5 = 1 nY4 = 1 nY3 = 1 nY2 = 1 nY1 = 1 nY0 = 1

C:\Users\Livin\Documents\CSE320 Files\Vincent-Li-A1>vvp dec3x8
nE1 = 0 nE2 = 0 E3 = 1 A2 = 0 A1 = 0 A0 = 0      nY7 = 1 nY6 = 1 nY5 = 1 nY4 = 1 nY3 = 1 nY2 = 1 nY1 = 1 nY0 = 0
nE1 = 0 nE2 = 0 E3 = 1 A2 = 0 A1 = 0 A0 = 1      nY7 = 1 nY6 = 1 nY5 = 1 nY4 = 1 nY3 = 1 nY2 = 1 nY1 = 0 nY0 = 1
nE1 = 0 nE2 = 0 E3 = 1 A2 = 0 A1 = 1 A0 = 0      nY7 = 1 nY6 = 1 nY5 = 1 nY4 = 1 nY3 = 1 nY2 = 0 nY1 = 1 nY0 = 1
nE1 = 0 nE2 = 0 E3 = 1 A2 = 0 A1 = 1 A0 = 1      nY7 = 1 nY6 = 1 nY5 = 1 nY4 = 1 nY3 = 0 nY2 = 1 nY1 = 1 nY0 = 1
nE1 = 0 nE2 = 0 E3 = 1 A2 = 1 A1 = 0 A0 = 0      nY7 = 1 nY6 = 1 nY5 = 1 nY4 = 0 nY3 = 1 nY2 = 1 nY1 = 1 nY0 = 1
nE1 = 0 nE2 = 0 E3 = 1 A2 = 1 A1 = 0 A0 = 1      nY7 = 1 nY6 = 1 nY5 = 0 nY4 = 1 nY3 = 1 nY2 = 1 nY1 = 1 nY0 = 1
nE1 = 0 nE2 = 0 E3 = 1 A2 = 1 A1 = 1 A0 = 0      nY7 = 1 nY6 = 0 nY5 = 1 nY4 = 1 nY3 = 1 nY2 = 1 nY1 = 1 nY0 = 1
nE1 = 0 nE2 = 0 E3 = 1 A2 = 1 A1 = 1 A0 = 1      nY7 = 0 nY6 = 1 nY5 = 1 nY4 = 1 nY3 = 1 nY2 = 1 nY1 = 1 nY0 = 1

C:\Users\Livin\Documents\CSE320 Files\Vincent-Li-A1>iverilog -o dec3x8 Vincent-Li-A1-Q3b-dec3x8_tb.v Vincent-Li-A1-Q3b-dec3x8.v

C:\Users\Livin\Documents\CSE320 Files\Vincent-Li-A1>vvp dec3x8
Actual outputs equal expected outputs for all input combinations.

C:\Users\Livin\Documents\CSE320 Files\Vincent-Li-A1>

```

4)

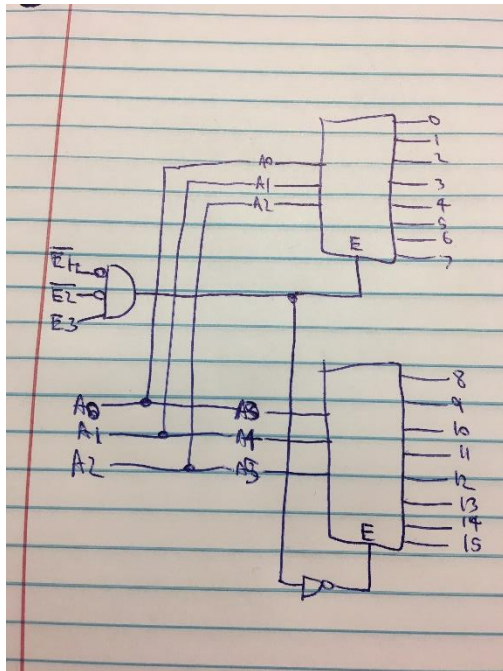
Truth table:

	A2	A1	A0	AnE1	AnE2	AE3	BnE1	BnE2	BE3	Output
	0	0	0	0	0	1	1	1	0	Y[0]
	0	0	1	0	0	1	1	1	0	Y[1]

	0	1	0	0	0	1	1	1	0	Y[2]
	0	1	1	0	0	1	1	1	0	Y[3]
	1	0	0	0	0	1	1	1	0	Y[4]
	1	0	1	0	0	1	1	1	0	Y[5]
	1	1	0	0	0	1	1	1	0	Y[6]
	1	1	1	1	1	0	0	0	1	Y[7]
	0	0	0	1	1	0	0	0	1	Y[8]
	0	0	1	1	1	0	0	0	1	Y[9]
	0	1	0	1	1	0	0	0	1	Y[10]
	0	1	1	1	1	0	0	0	1	Y[11]
	1	0	0	1	1	0	0	0	1	Y[12]
	1	0	1	1	1	0	0	0	1	Y[13]
	1	1	0	1	1	0	0	0	1	Y[14]
	1	1	1	1	1	0	0	0	1	Y[15]

Note: One enable should be the opposite of the other

Logic diagram:



Code:

```
// Filename: Vincent-Li-A1-Q3c-dec4x16.v
// Name: Vincent Li
// Course: CSE320 T/TH 4:30PM
// Professor: Vrudhula
/*
  A 4x16 decoder composed of two 3x8 decoders (74HC138).
*/
```

```

module dec3x8 (
    input wire nE1, nE2, E3,
    input wire [2 : 0] A,
    output wire [7 : 0] nY
);
    wire enable;

    assign enable = !nE1 && !nE2 && E3;

    assign nY[0] = nE1 || enable && !(A[2] && !A[1] && !A[0]);
    assign nY[1] = nE1 || enable && !(A[2] && !A[1] && A[0]);
    assign nY[2] = nE1 || enable && !(A[2] && A[1] && !A[0]);
    assign nY[3] = nE1 || enable && !(A[2] && A[1] && A[0]);
    assign nY[4] = nE1 || enable && !A[2] && !A[1] && !A[0];
    assign nY[5] = nE1 || enable && !A[2] && !A[1] && A[0];
    assign nY[6] = nE1 || enable && !A[2] && A[1] && !A[0];
    assign nY[7] = nE1 || enable && !A[2] && A[1] && A[0];

endmodule

module dec4x16 (
    input wire nE1, nE2, E3,
    input wire [2 : 0] A,
    output wire [15 : 0] nY
);
    wire E1, E2, nE3;
    assign E1 = !nE1;
    assign E2 = !nE2;
    assign nE3 = !E3;

    wire [7 : 0] nYFirstHalf, nYSecondHalf;
    assign nY[7 : 0] = nYFirstHalf;
    assign nY[15 : 8] = nYSecondHalf;

    // D1 for outputs 0 to 7.
    dec3x8 D1(.nE1(nE1), .nE2(nE2), .E3(E3), .A(A), .nY(nYFirstHalf));
    // D2 for outputs 8 to 15. Enable is negated
    dec3x8 D2(.nE1(E1), .nE2(E2), .E3(nE3), .A(A), .nY(nYSecondHalf));

endmodule

```

```

// Filename: Vincent-Li-A1-Q3c-dec4x16_tb.v
// Name: Vincent Li
// Course: CSE320 T/TH 4:30PM

```

```

// Professor: Vrudhula
/*
    Test bench for Vincent-Li-A1-Q3c-dec4x16.v
*/
module dec4x16_tb;
    reg nE1, nE2, E3;
    reg [2 : 0] A;
    wire [15 : 0] nY;
    integer allCorrect = 1;

    dec4x16 D(.nE1(nE1), .nE2(nE2), .E3(E3), .A(A), .nY(nY));

    initial
        begin
            nE1 = 0;
            nE2 = 0;
            E3 = 1;
            A = 0;
            #1;
            repeat(2) begin // modify enable from 001 to 110
                repeat(8) begin // increment A
                    #1;
                    $display("nE1 = %b, nE2 = %b, E3 = %b, A = %b, nY = %b", nE1,
nE2, E3, A, nY);
                    A = A + 1;
                end
                // reset a
                A = 0;
                // negating enable
                nE1 = 1;
                nE2 = 1;
                E3 = 0;
                #1;
            end
        end
endmodule

```

Results:

```
Command Prompt
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Livin>cd Documents

C:\Users\Livin\Documents>cd CSE320 Files

C:\Users\Livin\Documents\CSE320 Files>cd Vincent-Li-A1

C:\Users\Livin\Documents\CSE320 Files\Vincent-Li-A1>iverilog -o dec4x16 Vincent-Li-A1-Q3c-dec4x16_tb.v Vincent-Li-A1-Q3c-dec4x16.v

C:\Users\Livin\Documents\CSE320 Files\Vincent-Li-A1>vvp dec4x16
nE1 = 0, nE2 = 0, E3 = 1, A = 000, nY = 1111111111111110
nE1 = 0, nE2 = 0, E3 = 1, A = 001, nY = 1111111111111101
nE1 = 0, nE2 = 0, E3 = 1, A = 010, nY = 1111111111111011
nE1 = 0, nE2 = 0, E3 = 1, A = 011, nY = 1111111111111011
nE1 = 0, nE2 = 0, E3 = 1, A = 100, nY = 1111111111110111
nE1 = 0, nE2 = 0, E3 = 1, A = 101, nY = 1111111111110111
nE1 = 0, nE2 = 0, E3 = 1, A = 110, nY = 1111111111110111
nE1 = 0, nE2 = 0, E3 = 1, A = 111, nY = 1111111111110111
nE1 = 1, nE2 = 1, E3 = 0, A = 000, nY = 1111111011111111
nE1 = 1, nE2 = 1, E3 = 0, A = 001, nY = 1111111011111111
nE1 = 1, nE2 = 1, E3 = 0, A = 010, nY = 1111111011111111
nE1 = 1, nE2 = 1, E3 = 0, A = 011, nY = 1111111011111111
nE1 = 1, nE2 = 1, E3 = 0, A = 100, nY = 1110111111111111
nE1 = 1, nE2 = 1, E3 = 0, A = 101, nY = 1101111111111111
nE1 = 1, nE2 = 1, E3 = 0, A = 110, nY = 1011111111111111
nE1 = 1, nE2 = 1, E3 = 0, A = 111, nY = 0111111111111111

C:\Users\Livin\Documents\CSE320 Files\Vincent-Li-A1>
```