EGC220
Class Notes
3/10/2022

Baback Izadi
Division of Engineering Programs
bai@engr.newpaltz.edu

Test 1

Spring 2022		
Average	#####	83
Median	#####	83
MAX	#####	100
Minimum	#####	42

Spring 2023			
Average			67
Median			71
MAX			96
Minimum			0

Gray code

0	00
1	01
2	11
3	10

2 bit Gray code

0	000
1	001
2	011
3	010
4	110
5	111
6	101
7	100

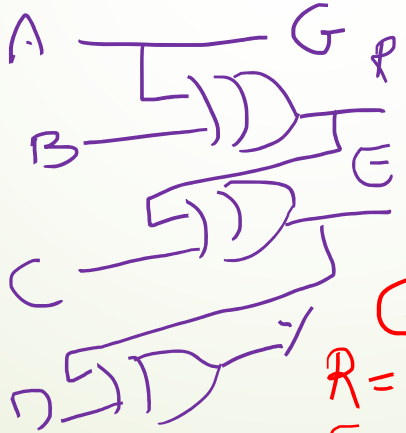
3 bit Gray code

ABCD	GRE Y
0000	0000
0001	0001
0010	0011
0011	0010
0100	0110
0101	0111
0110	0101
0111	0100
1000	1100
1001	1101
1010	1111
1011	1110
1100	1010
1101	1011
1110	1001
1111	1000

$R = G \oplus B$

$E = R \oplus C$

$Y = E \oplus D$



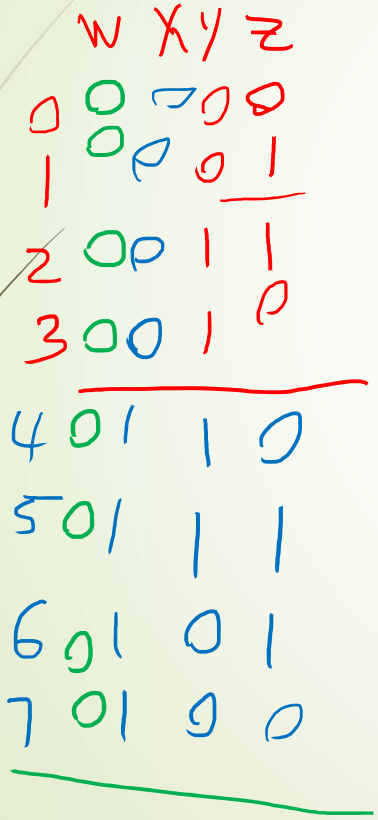
$G = A$

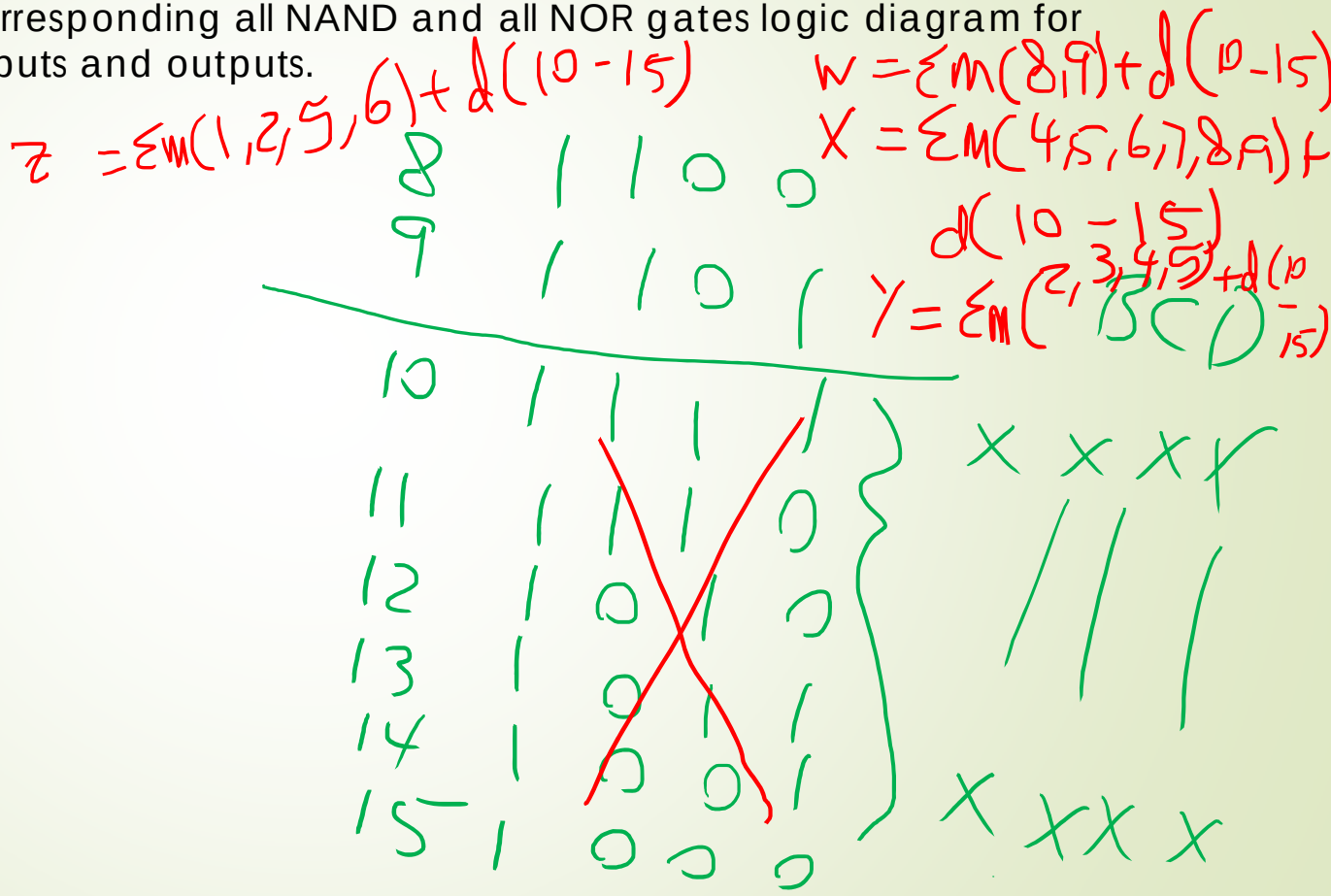
$R = \Sigma m(4,5,6,7,8,9,10,11)$

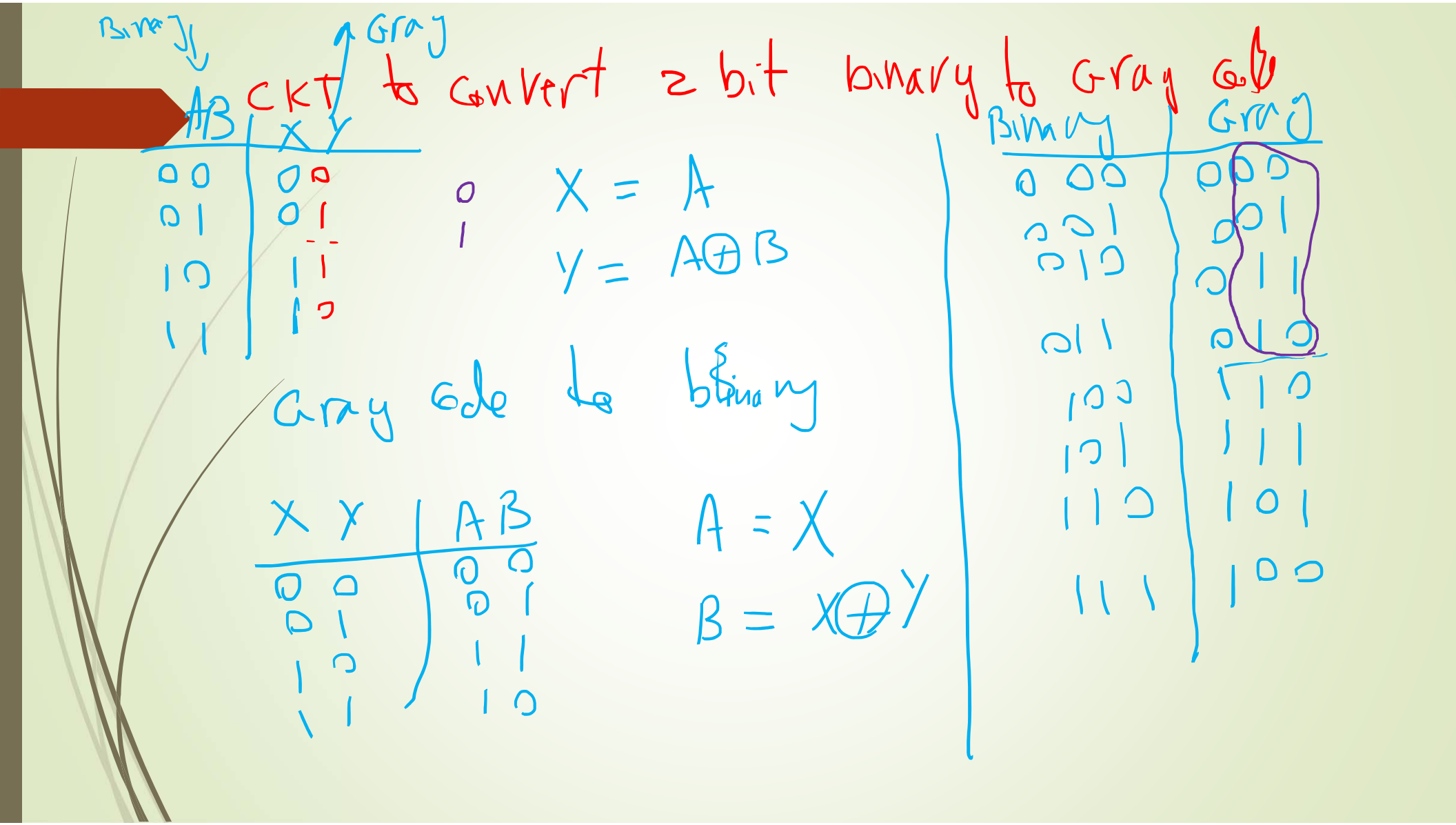
$E = \Sigma m(2,3,4,5,10,11,12,13)$

$Y = \Sigma m(1,2,5,6,9,10,13,14)$

Design a circuit that can convert a BCD code into a Gray. a. Write the truth table for this circuit. b. Find the minimized logic equations in SOP and POS for each output c. Draw the corresponding all NAND and all NOR gates logic diagram for this circuit. Label all inputs and outputs.






Binary
Gray
AB CKT to convert 2 bit binary to Gray code
A B | X Y
0 0 | 0 0
0 1 | 0 1
1 0 | 1 1
1 1 | 1 0
X = A
Y = A⊕B
Gray code to binary
X X | A B
0 0 | 0 0
0 1 | 0 1
1 0 | 1 1
1 1 | 1 0
A = X
B = X⊕Y
Binary | Gray
0 0 0 | 0 0 0
0 0 1 | 0 0 1
0 1 0 | 0 1 1
0 1 1 | 0 1 0
1 0 0 | 1 1 0
1 0 1 | 1 1 1
1 1 0 | 1 0 1
1 1 1 | 1 0 0

GRAY Code

BCD → GRAY

BCD

A B C D	GREY	
0 0 0 0	0 0 0 0	← 0
0 0 0 1	0 0 0 1	← 1
0 0 1 0	0 0 1 1	← 2
0 0 1 1	0 0 1 0	
0 1 0 0	0 1 1 0	
0 1 0 1	0 1 1 1	
0 1 1 0	0 1 0 1	
0 1 1 1	0 1 0 0	
1 0 0 0	1 1 0 0	
1 0 0 1	1 1 0 1	
1 0 1 0	1 1 1 1	dont
1 0 1 1	1 1 1 0	←
1 1 0 0	1 0 1 0	cares
1 1 0 1	1 0 1 1	
1 1 1 0	1 0 0 1	
1 1 1 1	1 0 0 0	← 15

$Y = \Sigma m(1,2,5,6,9) + d(\ )$

$G(A,B,C,D) = \Sigma m(8,9) + d(10,11,12,13,14,15)$

$R(A,B,C,D) = \Sigma m(4,5,6,7,8,9) + d(\ ,,\ )$

$E(A,B,C,D) = \Sigma m(2,3,4,5) + d(\ ,,\ )$

BCD → GRAY

ABCD	GREY
0000	0000
0001	0001
0010	0011
0011	0010
0100	0110
0101	0111
0110	0101
0111	0100
1000	1100
1001	1101
1010	1111
1011	1110
1100	1010
1101	1011
1110	1001
1111	1000

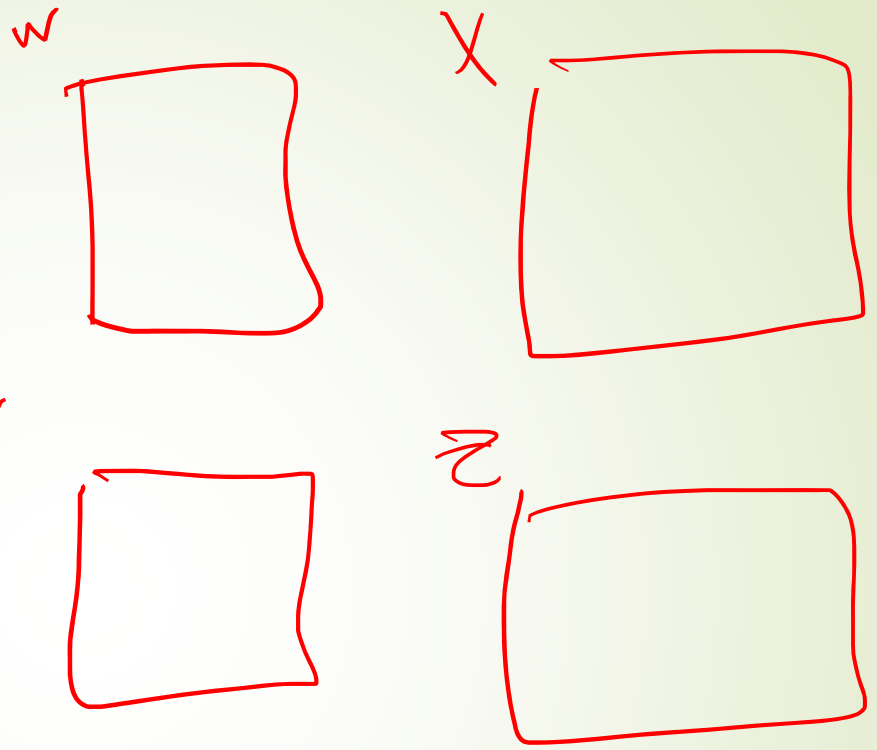
GRAY → BCD

GREY	ABCD
0000	0000
0001	0001
0010	0011
0011	0010
0100	0111
0101	0110
0110	0101
0111	0100
1000	XXXX
1001	XXXX
1010	X
1011	X
1100	1000
1101	1001
1110	XX
1111	XX

$A = \Sigma m(13, \ldots)$

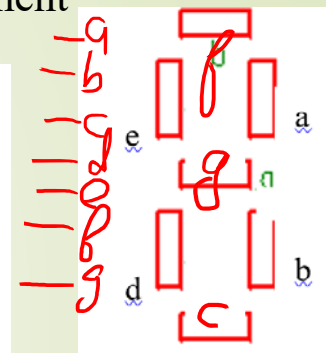
$B = \Sigma m(\_\_\_)$

ABCD	WXYZ
0000	0000
0001	0001
0010	0011
0011	0010
0100	0110
0101	0111
0110	0101
0111	0100
1000	1100
1001	1101
1010	1111
1011	1110
1100	1010
1101	1011
1110	1001
1111	1000



Problem 1

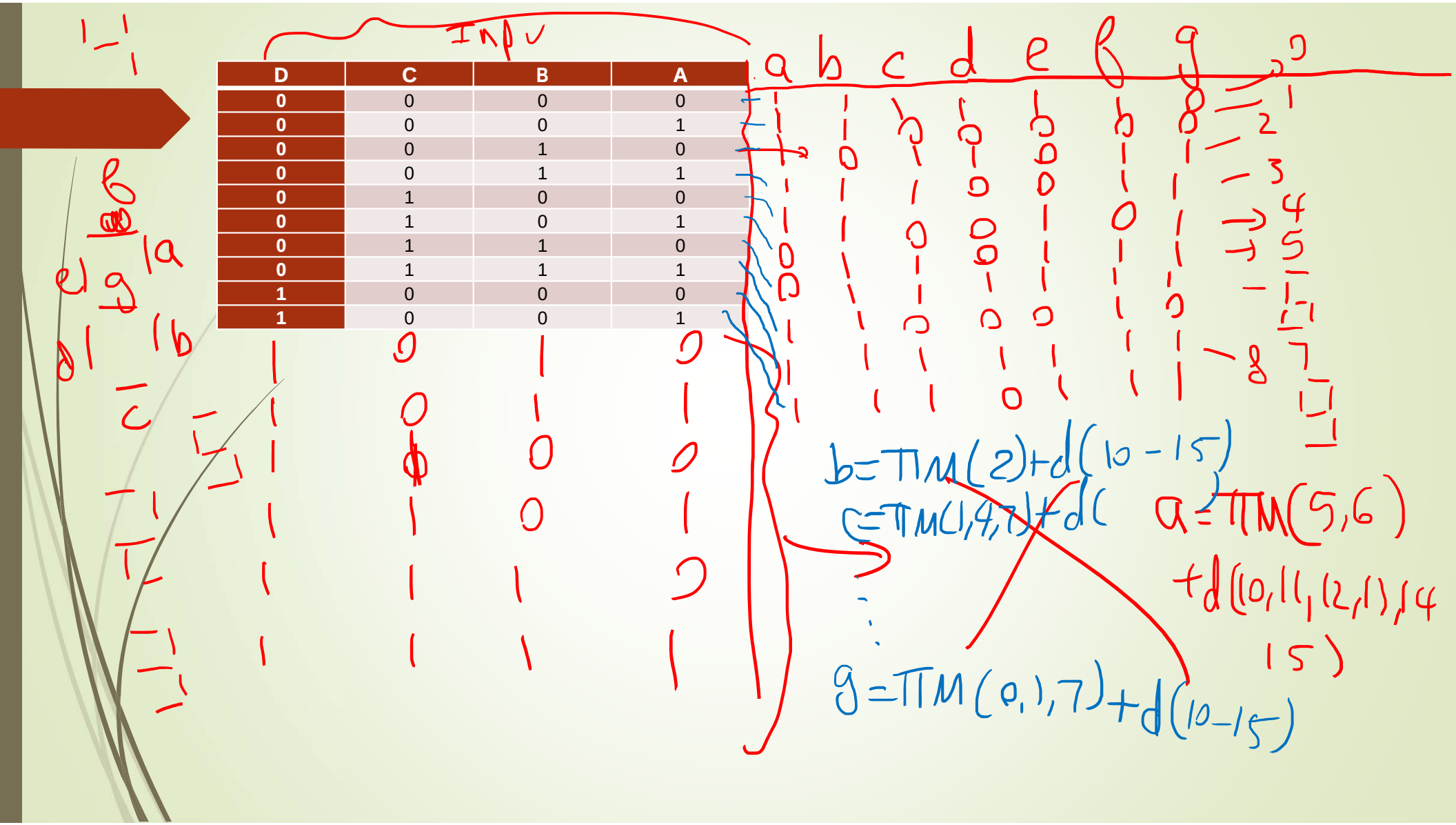
There is an integrated circuit called a BCD-seven segment decode that takes 4 inputs and has seven output. The inputs represent a number between 0 and 9, and each of the seven outputs corresponds to one of seven LED's in a seven-segment display. A typical seven segment display is shown below.



a. Write the truth table for each segment "a, b, c, d, e, f, g" with inputs A, B, C, and D. Make sure to adhere to the indicated segment notations.
b. Simplify each output in Minimum S.O.P.
c. Implement each output using all NAND gates.

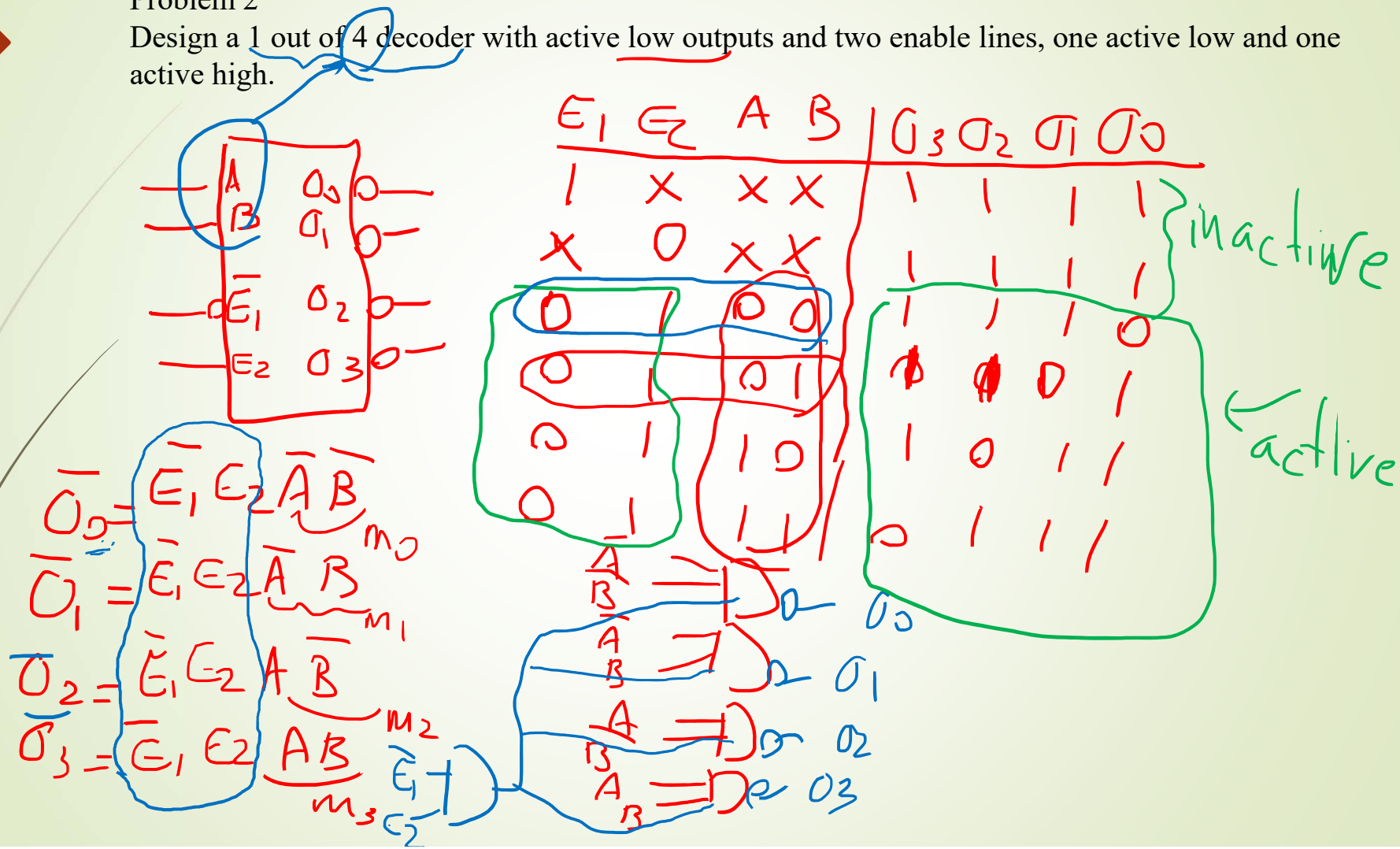
Digit	Inputs				Output (7 Segments)
	D	C	B	A	
0	0	0	0	0	
1	0	0	0	1	
2	0	0	1	0	
3	0	0	1	1	
4	0	1	0	0	
5	0	1	0	1	
6	0	1	1	0	
7	0	1	1	1	
8	1	0	0	0	
9	1	0	0	1	

D	C	B	A
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1



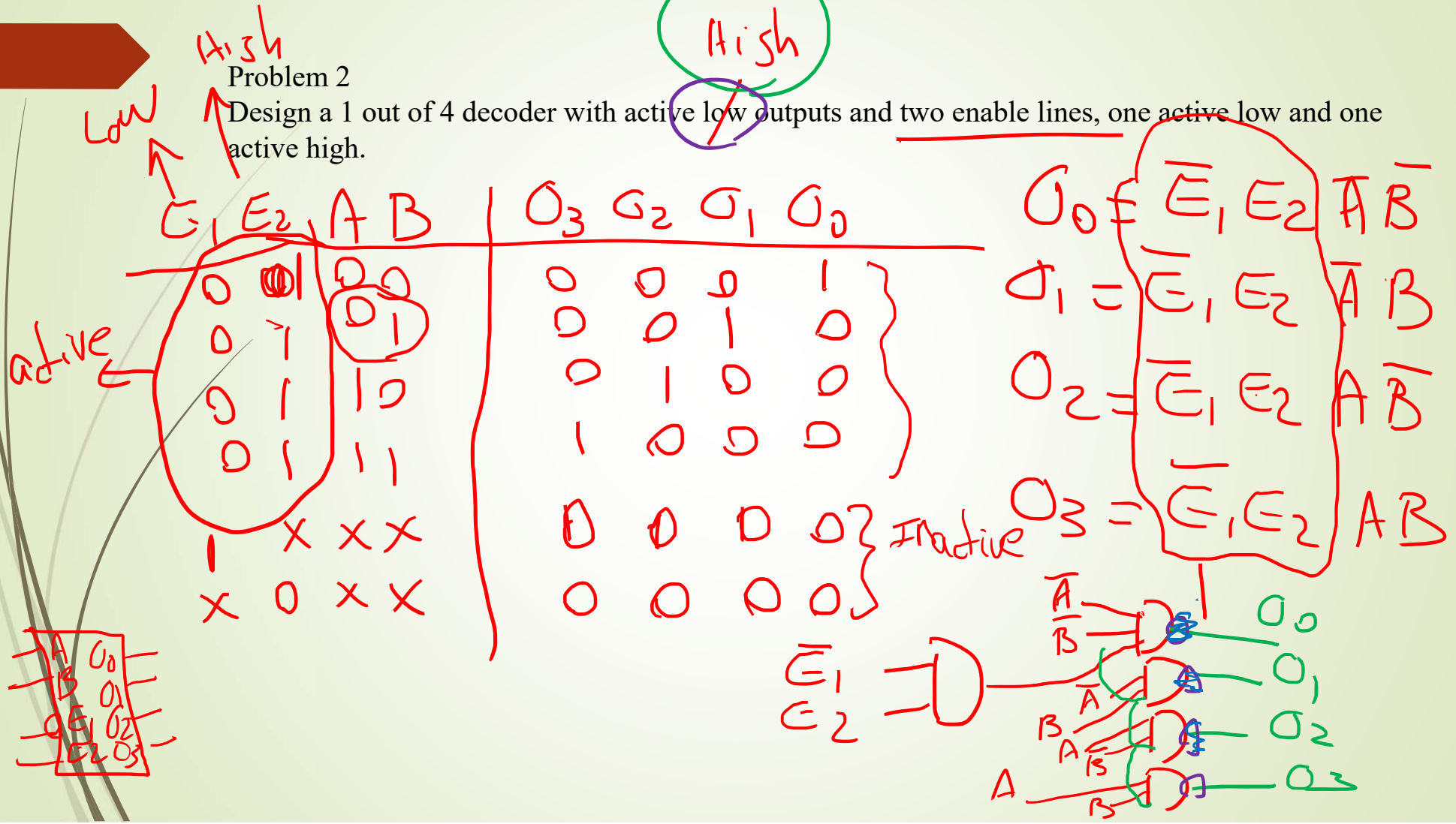
Problem 2

Design a 1 out of 4 decoder with active low outputs and two enable lines, one active low and one active high.



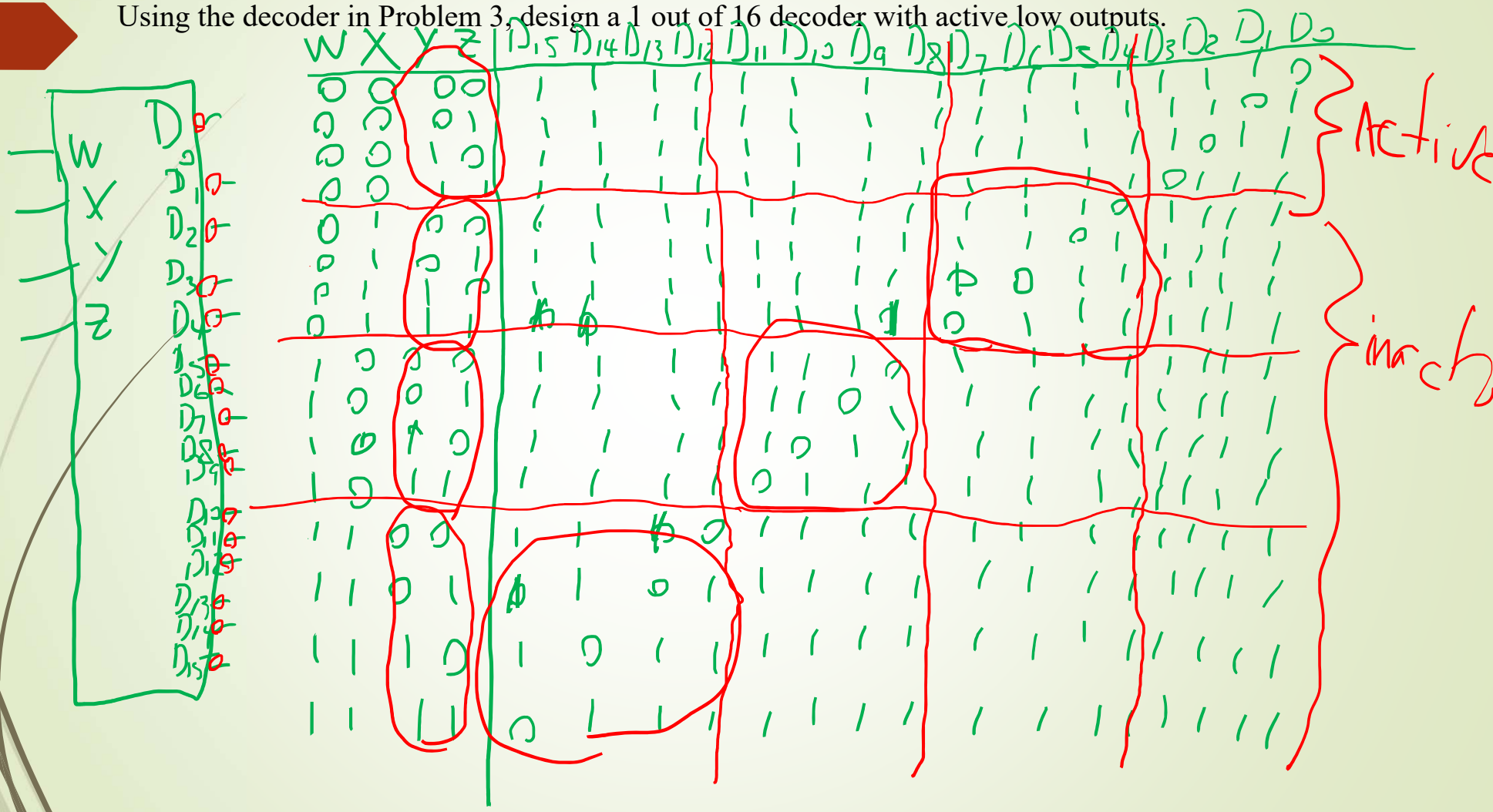
Problem 2

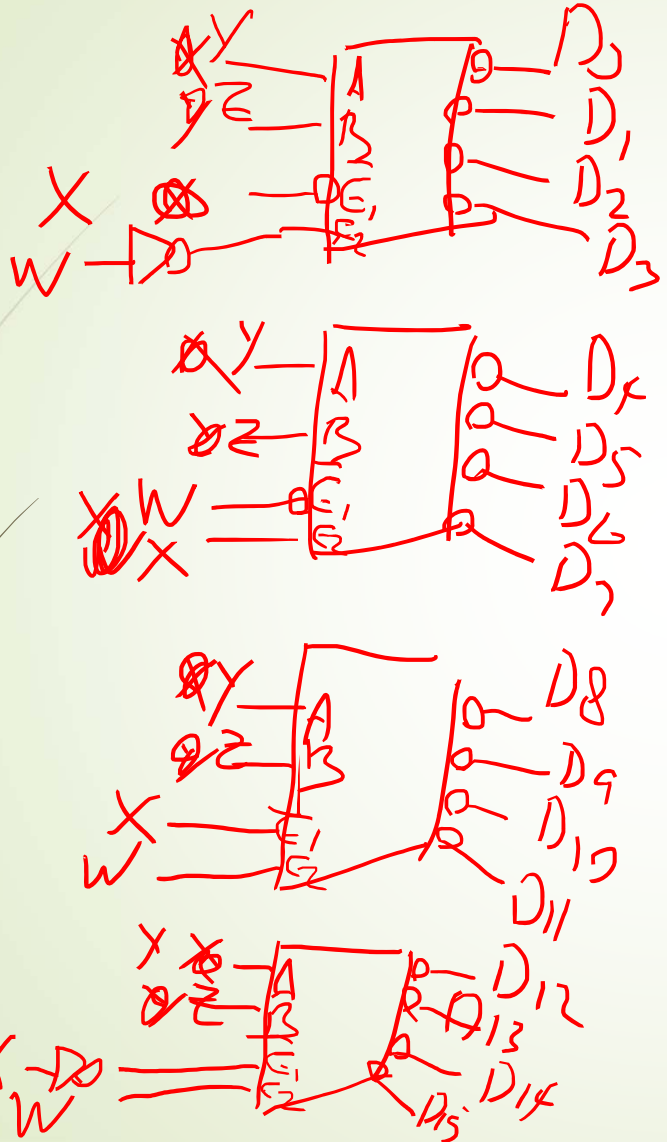
Design a 1 out of 4 decoder with active low outputs and two enable lines, one active low and one active high.



Problem 3

Using the decoder in Problem 3, design a 1 out of 16 decoder with active low outputs.




Y
Z
X
W
A
B
E1
E2
D0
D1
D2
D3
Y
Z
W
X
A
B
E1
E2
D4
D5
D6
D7
Y
Z
X
W
A
B
E1
E2
D8
D9
D10
D11
X
Z
X
W
A
B
E1
E2
D12
D13
D14
D15

Problem 4

Write a Verilog code for the decoder in Problem 2: 1-out of 4 decoder active low outputs and two enable lines, one active low and one active high.

```
module decoder-1-out-4 (EN, E, A, B, O0, O1, O2, O3);
input EN, E, A, B;
output O0, O1, O2, O3;
Assign ~O0 = ~EN & E & ~A & ~B;
Assign ~O1 = ~EN & E & ~A & B;
Assign ~O2 = ~EN & E & A & ~B;
Assign ~O3 = ~EN & E & A & B;
endmodule
```

1-out-of-4

```
// 2-to-4-Line Decoder with Enable: Dataflow Verilog Desc.          //  1
// (See Example 3-16 for logic diagram)                             //  2
module decoder_2_to_4_df_v(EN, A0, A1, D0, D1, D2, D3);             //  3
   input EN, A0, A1;                                                //  4
   output D0, D1, D2, D3;                                           //  5
                                                                    //  6
   assign D0 = EN & ~A1 & ~A0;                                      //  7
   assign D1 = EN & ~A1 & A0;                                       //  8
   assign D2 = EN & A1 & ~A0;                                       //  9
   assign D3 = EN & A1 & A0;                                        // 10
                                                                    // 11
endmodule                                                           // 12
```

Structural Verilog Description of 2–to–4-Line Decoder

```
// 2-to-4-Line Decoder with Enable: Structural Verilog Desc.          //  1
// (See Figure 3-16 for logic diagram)                                //  2
module decoder_2_to_4_st_v (EN, A0, A1, D0, D1, D2, D3);              //  3
   input EN, A0, A1;                                                  //  4
   output D0, D1, D2, D3;                                             //  5
                                                                      //  6
   wire A0_n, A1_n, N0, N1, N2, N3;                                   //  7
   not                                                                //  8
      g0(A0_n, A0),                                                   //  9
      g1(A1_n, A1);                                                   // 10
   and                                                                // 11
      g3(N0, A0_n, A1_n),                                             // 12
      g4(N1, A0, A1_n),                                               // 13
      g5(N2, A0_n, A1),                                               // 14
      g6(N3, A0, A1),                                                 // 15
      g7(D0, N0, EN),                                                 // 16
      g8(D1, N1, EN),                                                 // 17
      g9(D2, N2, EN),                                                 // 18
      g10(D3, N3, EN);                                                // 19
endmodule                                                             // 20
```