23/08/23
Wednesday.

Unit-2

Many to One.

→ Multiplexer → Universal Logic Circuit.

→ Mux → data Selector or many to One Circuit.

→ Parallel to Serial Converter.

→ n Select lines, input lines 2^n

→ Multiplexer involes AND & OR gate.

Multiplexer for implementing Boolean function:-

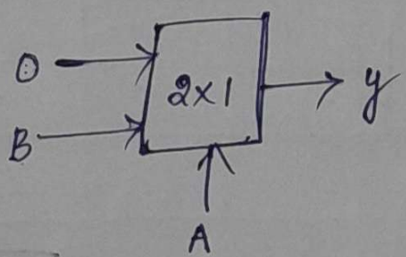
- n variables → n-1 Selection inputs.
- 1st n-1 variables are connected to Selection functions.

AND gate:

→ $y = A \cdot B$

* 2 input variables
* 1 Select input → 2×1 MUX

A	B	y
0	0	0
0	1	0
1	0	0
1	1	1

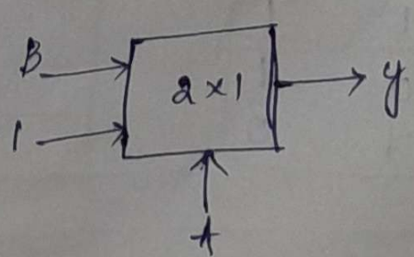


OR gate:

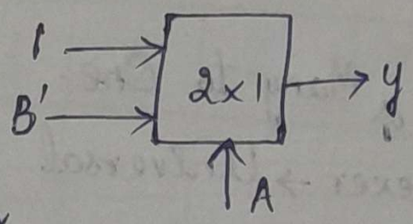
→ $y = A + B$

* 2 input variables
* 1 Select input → 2×1 MUX

A	B	y
0	0	0
0	1	1
1	0	1
1	1	1



NAND:



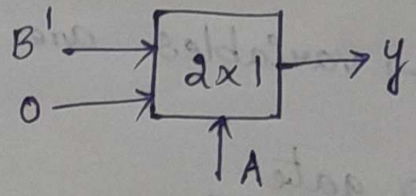
→ $y = \overline{A \cdot B}$

→ 2 input variables

→ 1 Select input ⇒ 2x1 Mux

A	B	y
0	0	1
0	1	1
1	0	1
1	1	0

NOR:



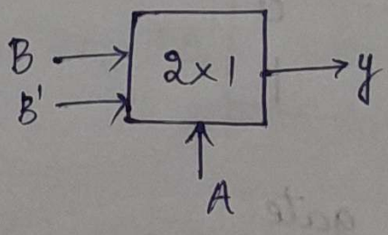
→ $y = \overline{A+B}$

→ 2 input varibles

→ 1 Select input ⇒ 2x1 MUX.

A	B	y
0	0	1
0	1	0
1	0	0
1	1	0

EX-OR:



→ $A \oplus B$

→ 2 input variables

→ 1 Select input ⇒ 2x1 MUX.

A	B	y
0	0	0
0	1	1
1	0	1
1	1	0

Multiplexer for implementing Boolean function.

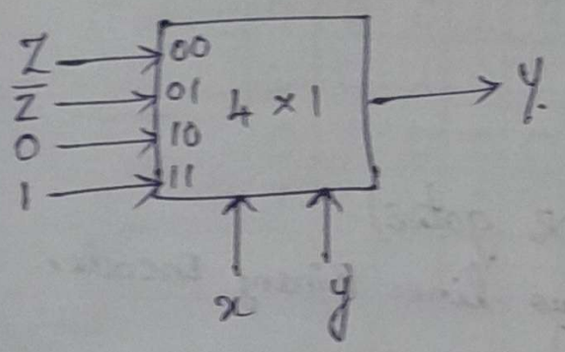
- Identify the no. of variables
- $F(x, y, z) = \Sigma(1, 2, 6, 7)$

$n = 3$

$n - 1 = 2 \rightarrow$ Select I/P

Implement 4×1 Mux

2×1 → 1 S I/P
4×1 → 2 S I/P
8×1 → 3 S I/P
16×1 → 4 S I/P



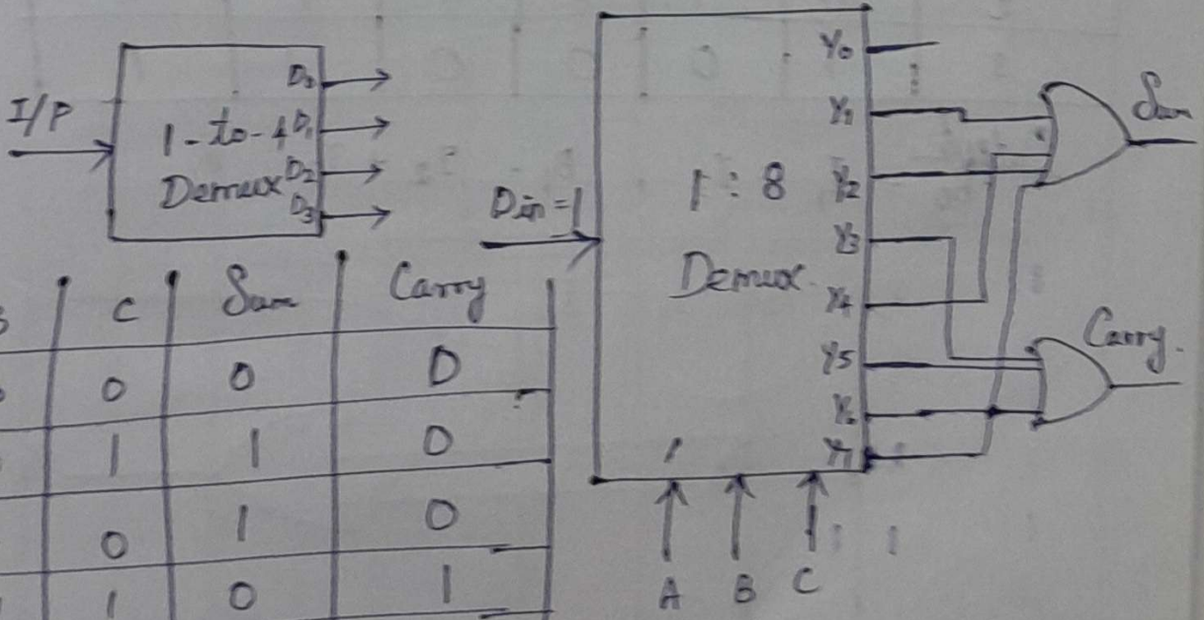
x	y	z	y	
0	0	0	0	z
0	0	1	1	
0	1	0	1	z̄
0	1	1	0	
1	0	0	0	0
1	0	1	0	
1	1	0	1	1
1	1	1	1	

30/08/2023
Wednesday

Demultiplexer → Data distributer

[One to many]

→ One Output line, 2^n output lines & n Select lines.



	A	B	C	Sum	Carry
Y_0	0	0	0	0	0
Y_1	0	0	1	1	0
Y_2	0	1	0	1	0
Y_3	0	1	1	0	1
Y_4	1	0	0	1	0
Y_5	1	0	1	0	1
Y_6	1	1	0	0	1
Y_7	1	1	1	1	1

31/08/2023
Thursday

Encoder:

Encoders:

★ Combinational Logic function has 2^n input lines and n output lines.

★ n output lines → binary code → 2^n input lines

Input $= 2^n$

Output $= n$

Encoder $- 2^n : n$ (OR gates)

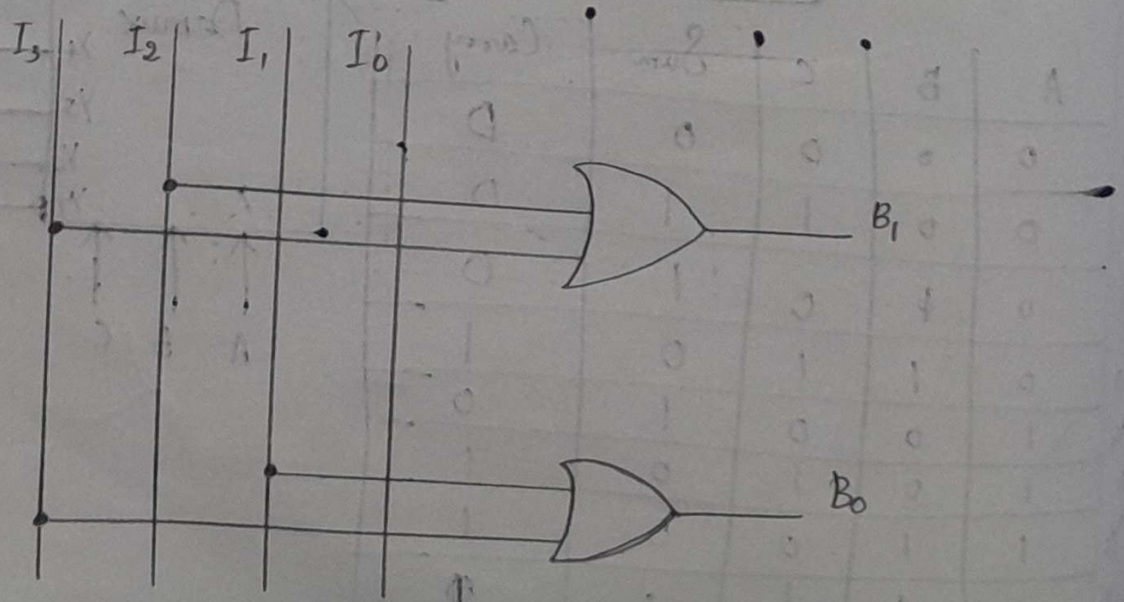
① Four (4) to two (2) lines Binary Encoder.

Truth table

Decimal value	Input				Output	
	I_3	I_2	I_1	I_0	B_1	B_0
0	0	0	0	1	0	0
1	0	0	1	0	0	1
2	0	1	0	0	1	0
3	1	0	0	0	1	1

Output:

$B_0 = I_1 + I_3$; $B_1 = I_2 + I_3$



② Eight to three lines Binary Encoder.
(8) (3)

Truth table:

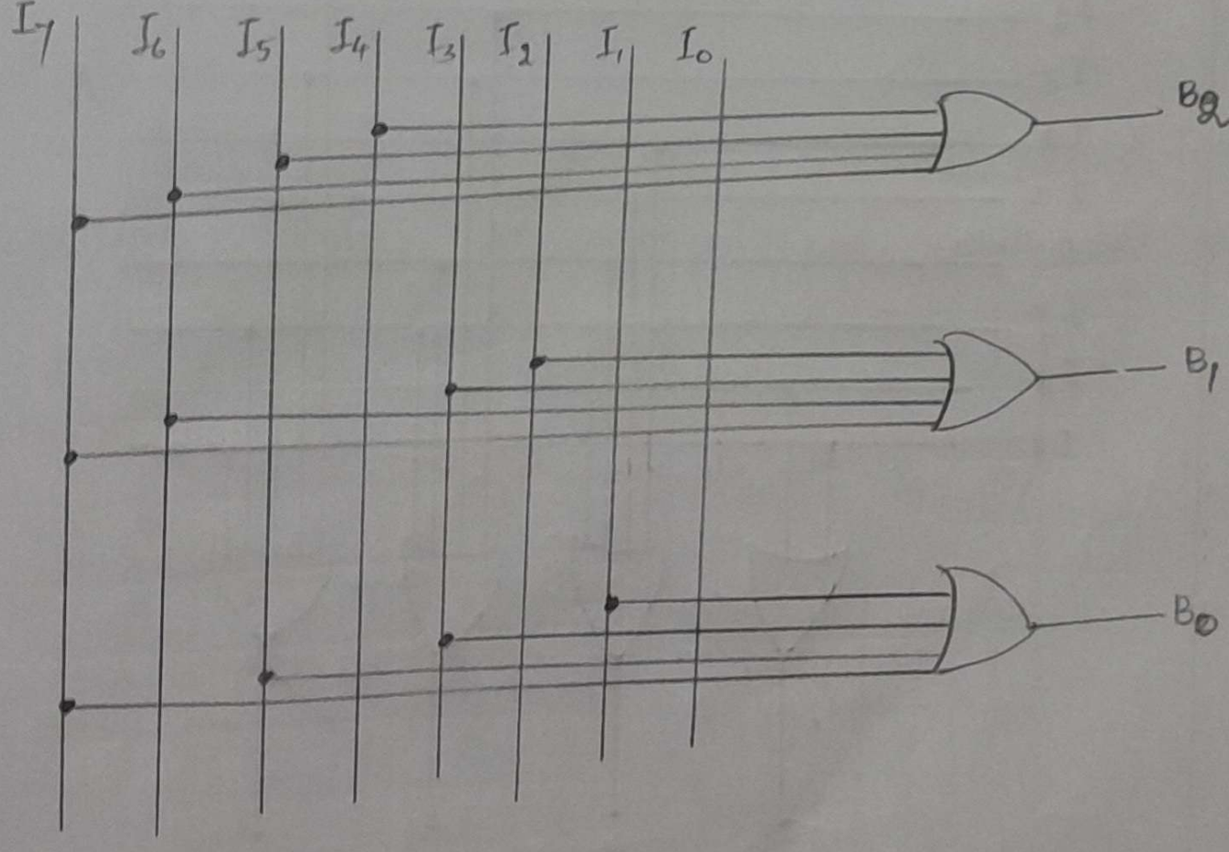
Decimal value	Input								Output		
	I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	B_2	B_1	B_0
0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	1	0	0	0	1
2	0	0	0	0	0	1	0	0	0	1	0
3	0	0	0	0	1	0	0	0	0	1	1
4	0	0	0	1	0	0	0	0	1	0	0
5	0	0	1	0	0	0	0	0	1	0	1
6	0	1	0	0	0	0	0	0	1	1	0
7	1	0	0	0	0	0	0	0	1	1	1
~~8~~											

Output

$B_0 = I_1 + I_3 + I_5 + I_7$

$B_1 = I_2 + I_3 + I_6 + I_7$

$B_2 = I_4 + I_5 + I_6 + I_7$



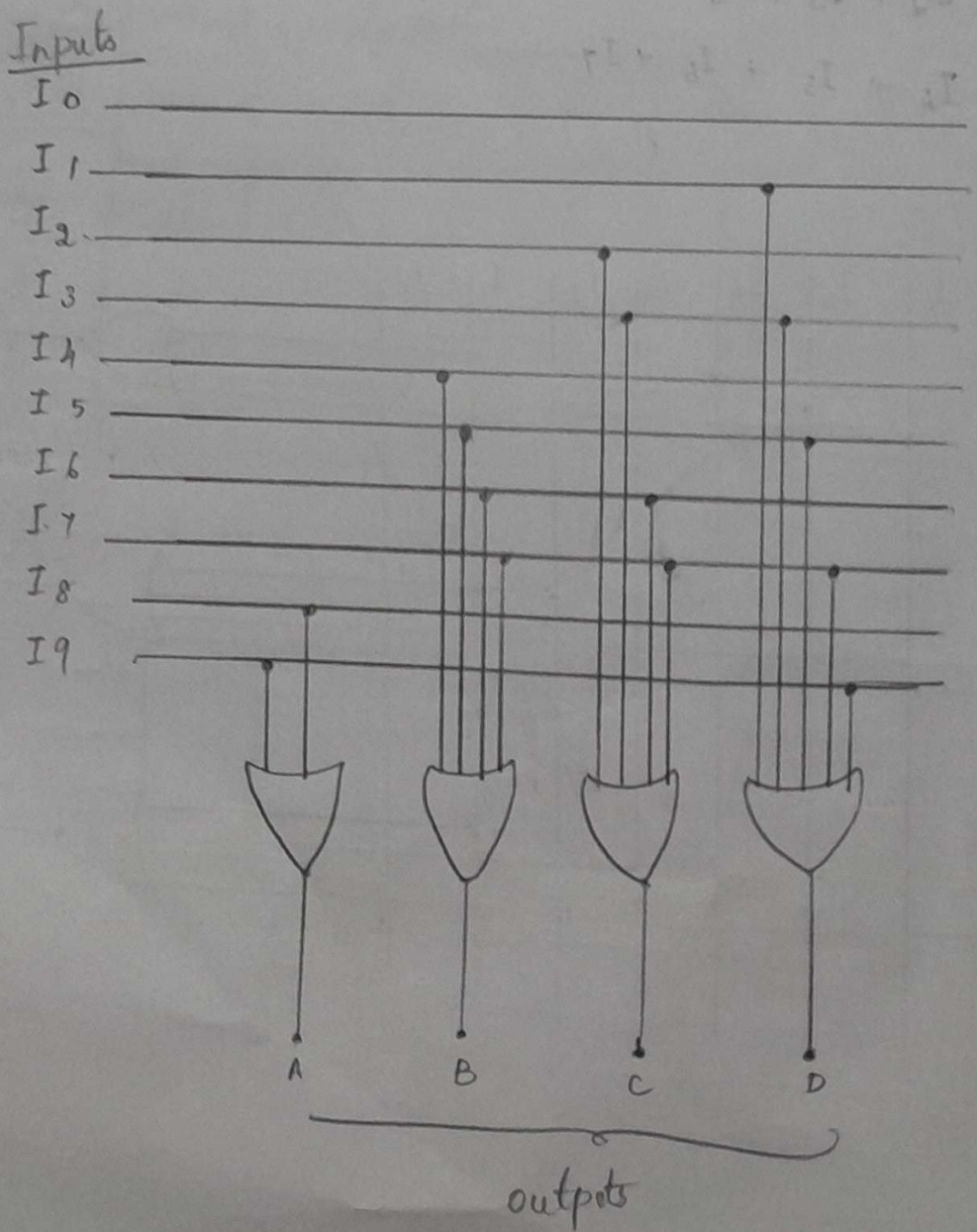
⑨ Decimal to BCD
(10) (4)

Truth Table:

Decimal Inputs										BCD output			
0	1	2	3	4	5	6	7	8	9	A	B	C	D
1										0	0	0	0
	1									0	0	0	1
		1								0	0	1	0
			1							0	0	1	1
				1						0	1	0	0
					1					0	1	0	1
						1				0	1	1	0
							1			0	1	1	1
								1		1	0	0	0
									1	1	0	0	1

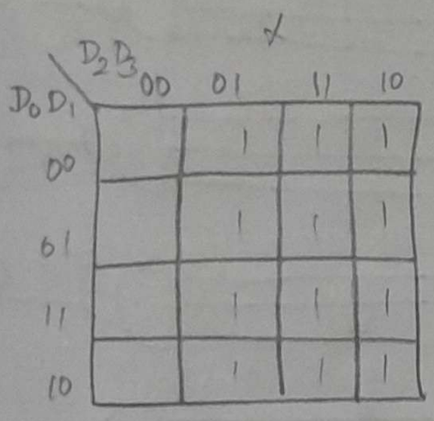
Output:

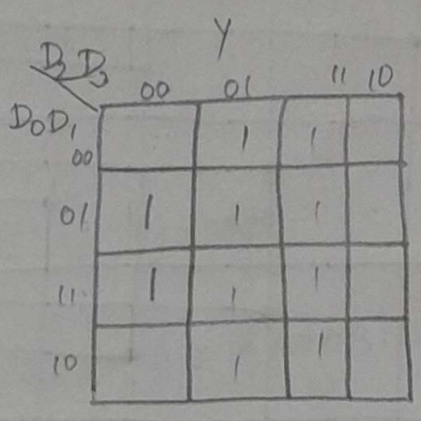
A = 8+9 , B = 4+5+6+7, C = 2+3+6+7, D = 1+3+5+7+9



Priority Encoder:

Inputs				Output		
D_0	D_1	D_2	D_3	X	Y	V
1	0	0	0	0	0	1
x	1	0	0	0	1	1
x	x	1	0	1	0	1
x	x	x	1	1	1	1





01/08/2023
Friday

Decoders:

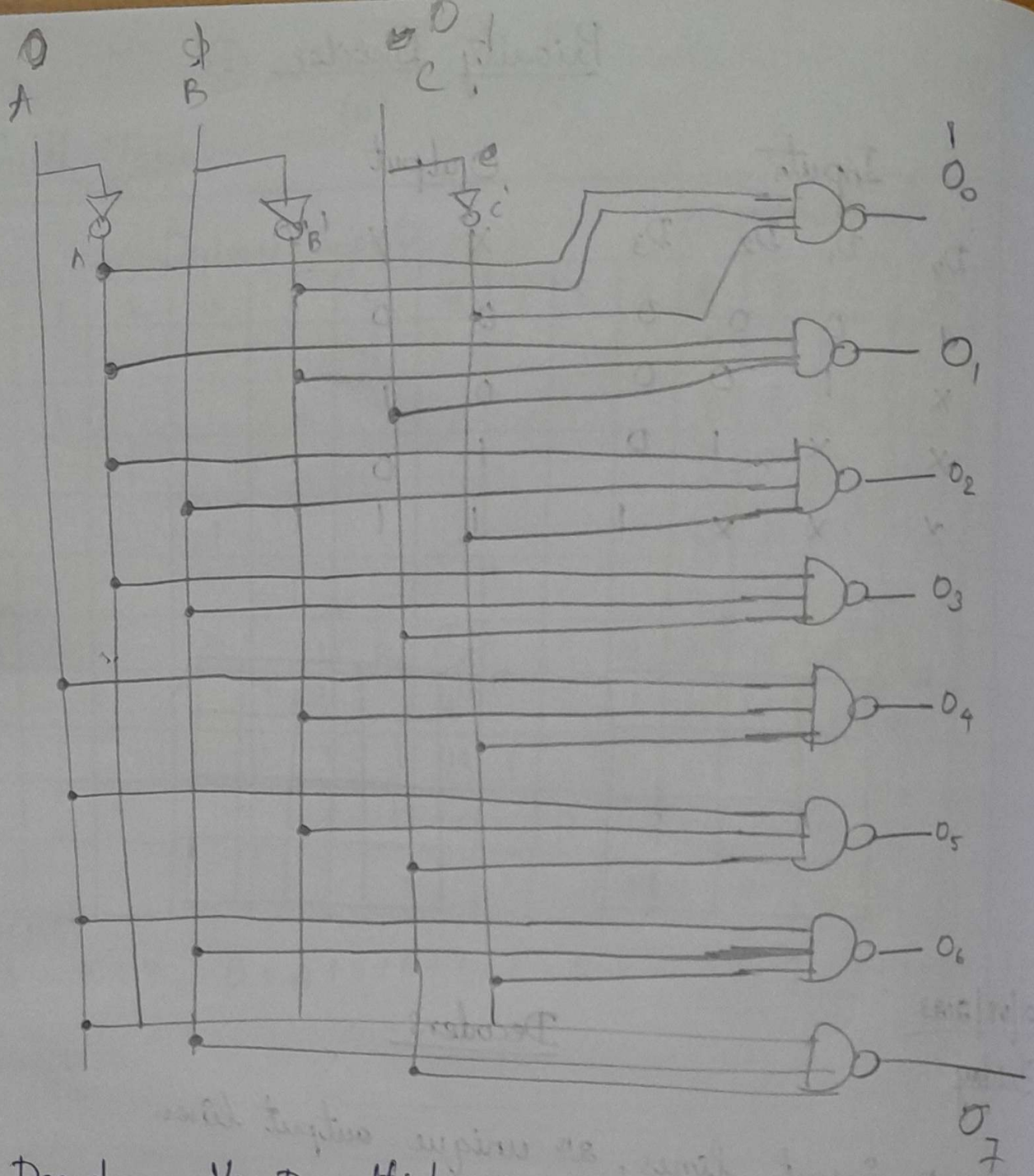
→ n input lines, 2^n unique output lines.

→ And gates & Inverters.

Three (3) to Eight (8) Decoder:

Active Low output (0)
Active high output (1)

Input				Output							
E	A	B	C	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
1	x	x	x	1	1	1	1	1	1	1	1
0	0	0	0	0	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1
0	1	1	0	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	0



Decoder Vs Demultiplexer:-

↓

→ Decoder is many inputs to many outputs.

→ There are no Selection lines.

Demux:

→ One input to many output

→ Selection of Specific output line is Controlled by the value of Selection lines.

Magnitude Comparator

→ Compare the values of two numbers.

Step 1:

If A = B

if not

Start Comparing from MSB

[MSB – Most Significant bit]

Step 2:

A > B (or) A < B.

Design equations:

$A > B : A_1B_1' + A_0B_1'B_0' + A_1A_0B_0'$

$A = B : A_1'A_0'B_1'B_0' + A_1'A_0B_0'B_0 + A_1A_0B_1B_0 + A_1A_0'B_1B_0'$

$: A_1'B_1'(A_0'B_0' + A_0B_0) + A_1B_1(A_0B_0 + A_0'B_0')$

$: (A_0B_0 + A_0'B_0')(A_1B_1 + A_1'B_1')$

: (A0 Ex-Nor B0) (A1 Ex-Nor B1)

$A < B : A_1'B_1 + A_0'B_1B_0 + A_1'A_0'B_0.$

07/09/2023
Thursday

Binary to Gray code Conversion

↳ Non-weighted code.

$G_1 = B_1$

$G_2 = B_1 \oplus B_2$

$G_3 = B_2 \oplus B_3$

$G_4 = B_3 \oplus B_4$

Gray to Binary

$B_1 = G_1$

$B_2 = G_1 \oplus G_2$

$B_3 = G_2 \oplus G_3$

$B_4 = G_3 \oplus G_4$

Four Bit Binary				Four Bit Gray code			
B_1	B_2	B_3	B_4	G_1	G_2	G_3	G_4
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
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
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0