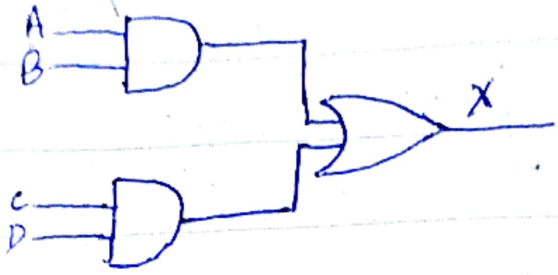
Chapter : 05

Ex: 5.1

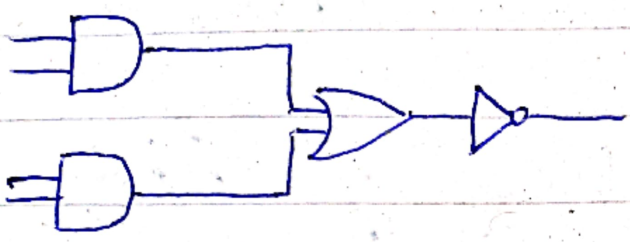
① AND-OR Logic:



$(AB)+(CD)$

~~$(A+B)(C+D)$~~

② AND-OR INVERT Logic:

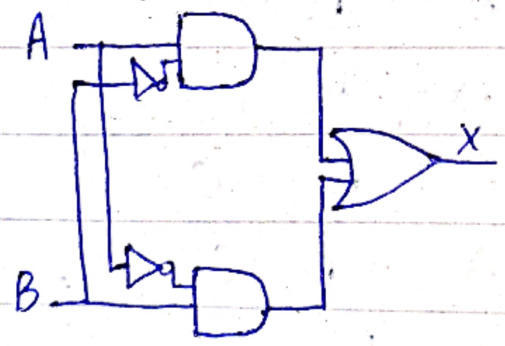


$\overline{AB+CD}$

$\overline{AB}\,\overline{CD}$

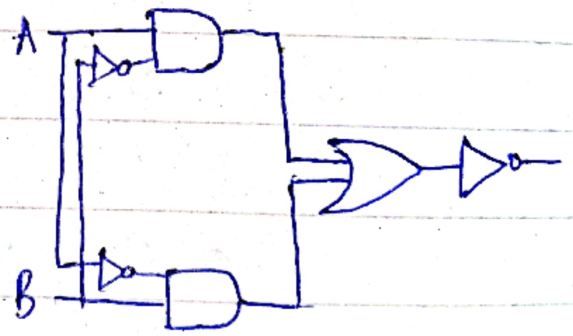
$(\overline{A}+\overline{B})(\overline{C}+\overline{D})$

③ XOR logic



$A\overline{B}+\overline{A}B$

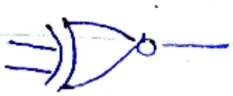
④ XNOR logic



$\overline{A\overline{B}+\overline{A}B} = \overline{(A\overline{B})}\,\overline{(\overline{A}B)}$

$(\overline{A}+\overline{\overline{B}})(\overline{\overline{A}}+\overline{B}) = (\overline{A}+B)(A+\overline{B})$

$= \overline{A}A+\overline{A}\overline{B}+AB+\overline{B}\overline{B}$



5.1 Checkup:

1) A=1, B=0, C=1, D=0
AB = 0, CD = 0, X = 0 → (1)

2) A=1, B=1, C=0, D=1
AB = 1, CD = 0, X = 1 → (0)

3) A=0, B=1, C=1, D=1
AB = 0, CD = 1, X = 1 → (0)

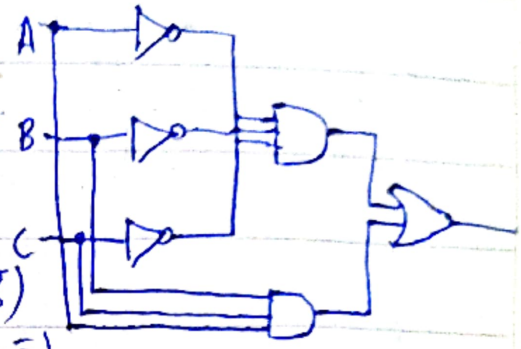
2) a) A=1, B=0, 1
b) A=1, B=1, D
c) A=0, B=1, 1
d) A=0, B=0, 0

5.2 checkup

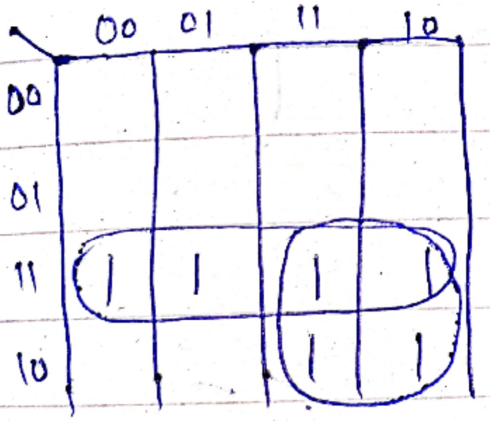
1) Ez

2) 0 0 0 1 $\bar{A}\bar{B}\bar{C}$
1 1 1 1 ABC

3) $ABC(D+\bar{D}) + AB(C+\bar{C})(D+\bar{D}) + AC(B+\bar{B})(D+\bar{D})$



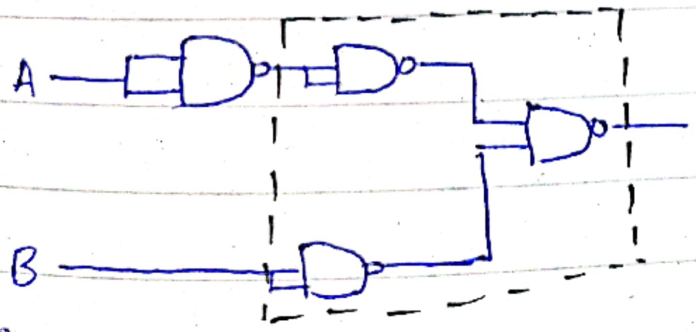
$ABCD + ABC\bar{D} + ABCD + ABC\bar{D} + AB\bar{C}D + AB\bar{C}\bar{D} + ABCD + \bar{A}BCD + AB\bar{C}\bar{D} + A\bar{B}C\bar{D}$



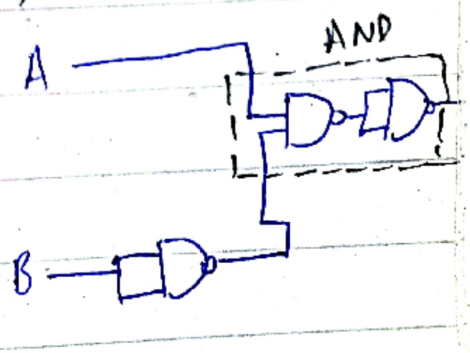
$AB + AC$ (By K-Map)

5.3 Checkup:

a) $X = \bar{A} + B$ OR



b) $X = A\bar{B}$



0	0	1	0	1	1
0	1	1	0	0	1
1	0	0	1	1	0
1	1				

5.1 Checkup:

1) A=1, B=0, C=1, D=0
   AB = 0, CD = 0, X = 0 → (1)
2) A=1, B=1, C=0, D=1
   AB = 1, CD = 0, X = 1 → (0)
3) A=0, B=1, C=1, D=1
   AB = 0, CD = 1, X = 1 → (0)

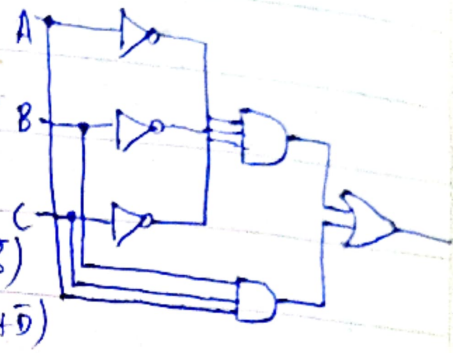
2) a) A=1, B=0, 1
   b) A=1, B=1, D
   c) A=0, B=1, 1
   d) A=0, B=0, 0

5.2 checkup

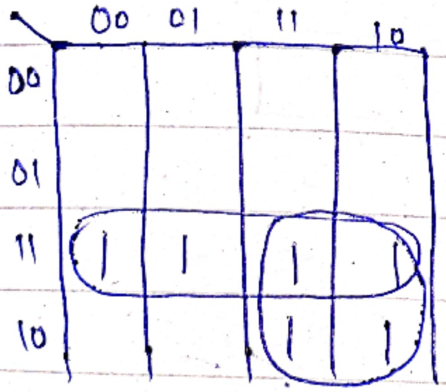
1) Ez

2) 0 0 0 1 $\bar{A}\bar{B}\bar{C}$
   1 1 1 1 ABC

3) $ABC(D+\bar{D}) + AB(C+\bar{C})(D+\bar{D}) + AC(B+\bar{B})(D+\bar{D})$



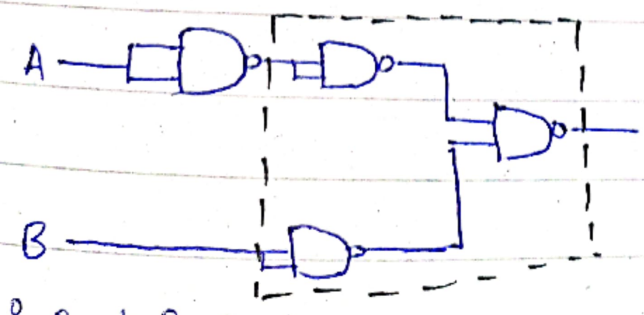
$ABCD + ABC\bar{D} + ABCD + ABC\bar{D} + AB\bar{C}D + AB\bar{C}\bar{D} + ABCD + \bar{A}BCD + \bar{A}BC\bar{D} + A\bar{B}C\bar{D}$



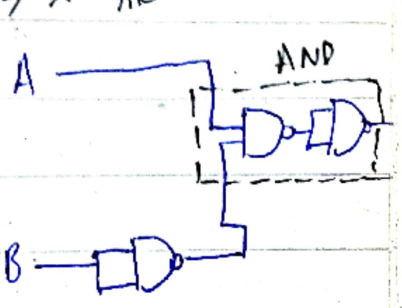
$AB + AC$ (By K-Map)

5.3 Checkup:

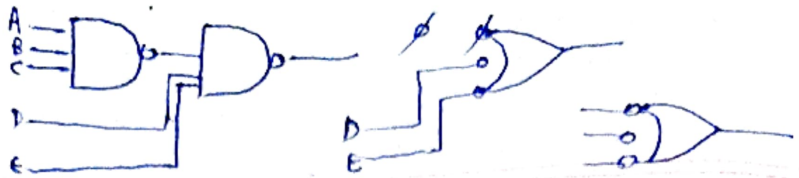
a) $X = \bar{A} + B$



b) $X = A\bar{B}$

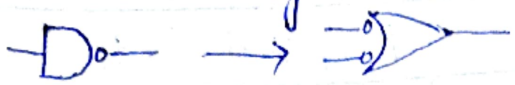


0 0 1 0 1 1
0 1 1 0 0 1
1 0 0 1 1 0
1 1 0 1 0 1

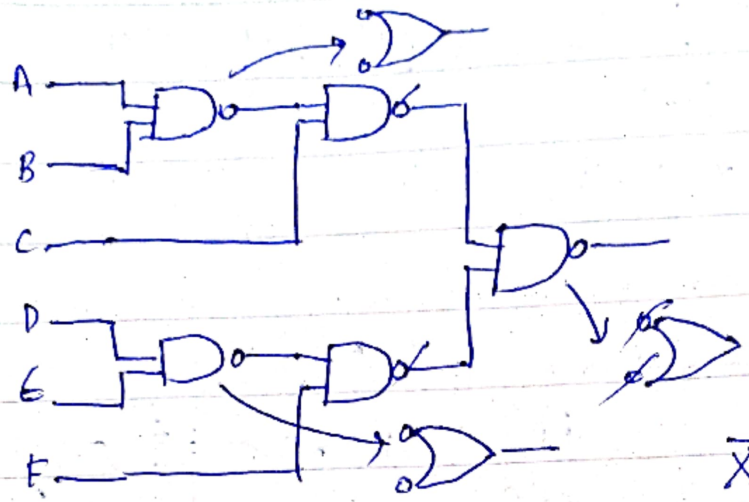


Section 5.4

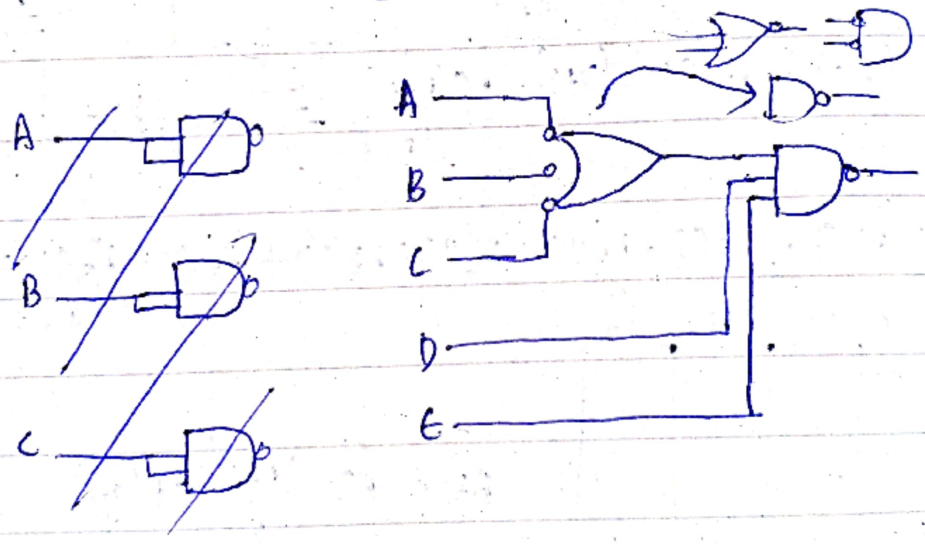
- NAND = Neg OR (De Morgan)

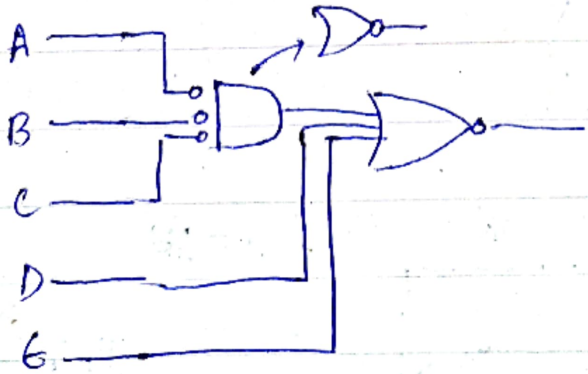


- Bubble cancel out when NOT on both side,



$\overline{X+Y} = \bar{X}\bar{Y}$





Chapter : 06

* Half Adder

$0+0 = 0$

$0+1 = 1$

$1+0 = 1$

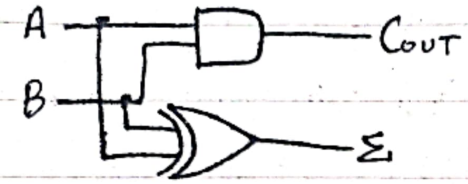
$1+1 = 0$ carry 1

- $C_{OUT} = AB$

generated only when 1 both (so AND gate)

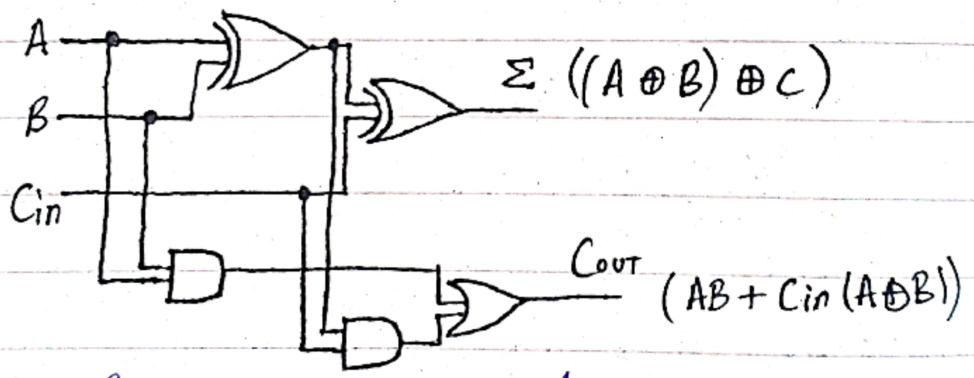
- $\Sigma = A \oplus B$

generated 1 only when different inputs (so XOR)



- Two inputs, 1 carry, 1 sum

* Full Adder:



- 3 inputs, 1 carry, 1 sum

Example:

F: A = ? when A=1, B=1, Cin=1 ?

$1+1+1 = 1$ carry 1

* Section 6.1:

a) 01

1 carry 0

b) 00

0 carry 0

c) 10

1 carry 0

d) 11

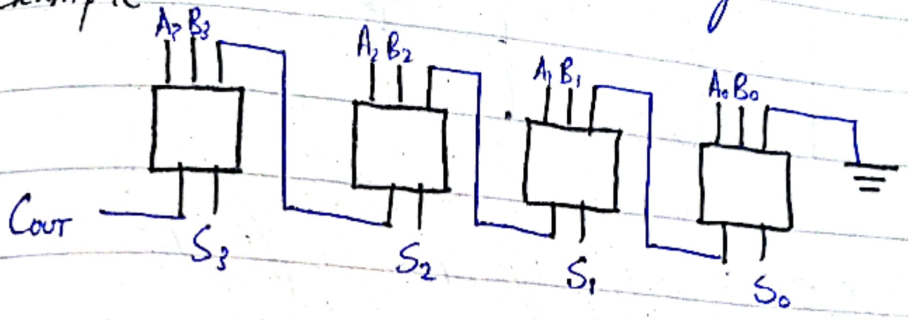
0 carry 1

• 4 bit num = 4 FA
• 3 " = 3 "
Num of bit = Num of FA

* Parallel Binary Adders:

- Cascade connection of full adders
- Need arise when are wanted to use adder for multibit 2 number as full adder can only comprehend 2 singlebit & 1 carry.

Example:

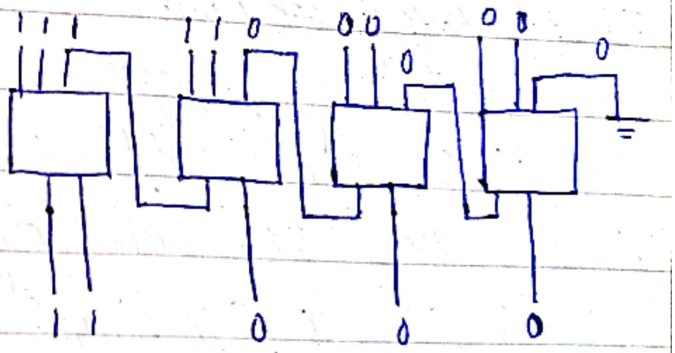


- 4 parallel full adders used for 4 bit 2 numbers

Example: 6·3

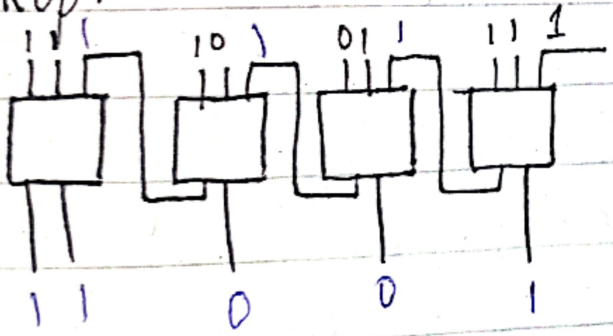
A = 1 1 0 0

B = 1 1 0 0



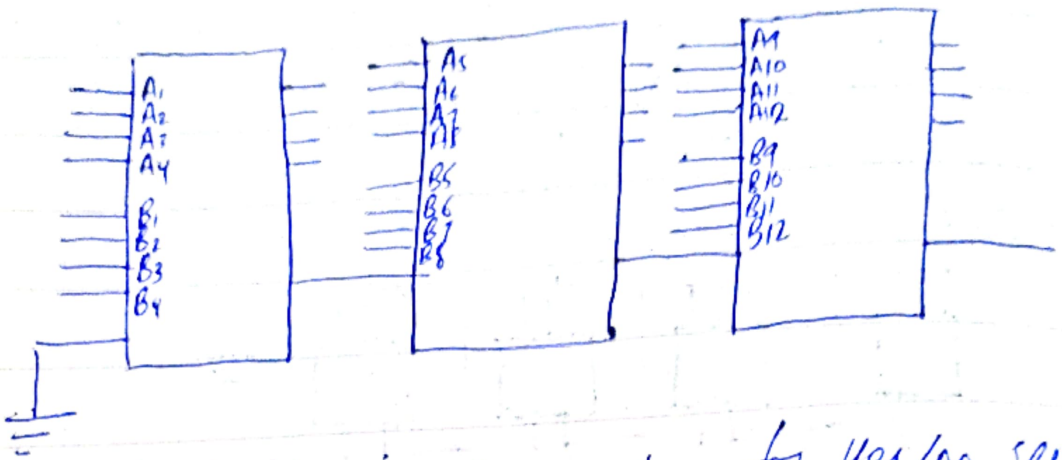
- COUT of 1 adder being input in higher order adder is called internal carries.

- 6·2 checkup:



Related Problem

3 FA to implement 12 bit adder



- Application is voting system for yes/no see fig 6-13 in book

* Ripple Carry & look Ahead

- The delay in carry propagation from Cout of previous adder to Cin of next adder cause time delay
- In worst case, there is such delay in each adder making it quite significant in last adder output
- When no carry produced (best case) addition time = Propogation time

- The concept of look ahead is predicting Cout without waiting for input from Cin.
- We predict Cout is 1 when ?

A	B	Cin	Cout
0	0	0	
0	0	1	
0	1	0	
0	1	1	1
1	0	0	
1	0	1	1
1	1	0	
1	1	1	1

only 1 when A⊕B & Cin=1 → $(A \oplus B) \cdot C_{in}$ — Carry Propagator (P)

only 1 when AB both 1 → AB — Carry generator (G)

$$C_{OUT} = AB + (A \oplus B)C_{in}$$
$$= G + PC_{in}$$

- by derivation of parallel look ahead adder, C_{OUT} is dependant only on C_{in1} and C_g & C_p

Exercise 6.3 Checkup:

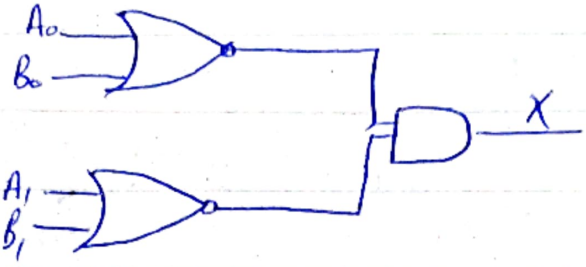
1) A=1, B=0, C_g=?, C_p=?
- $C_g = AB \rightarrow (1)(0) = 0$
- $C_p = (1) \oplus (0) = 1$

2) C_{OUT}=? when C_{in}=1, C_g=0, C_p=1
$C_{OUT} = (0) + (1)(1) = 1$

* Comparator:

① Equality (0 → when diff, 1 → when same)
- Use XNOR gate
- for one bit two num, use XNOR gate

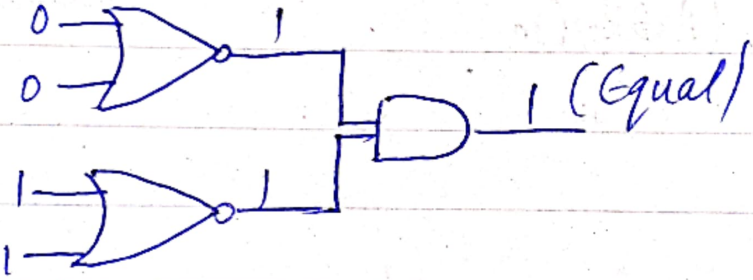
Two, 2 bit number



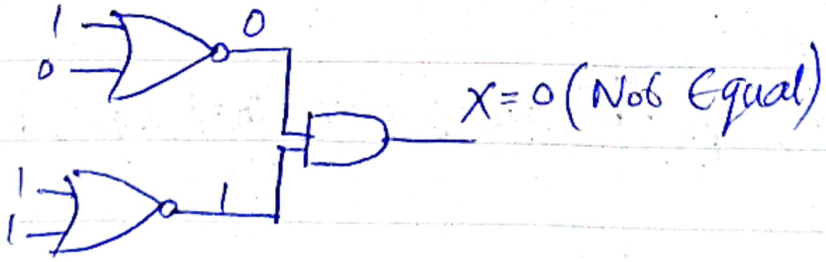
∴ 1 when 1 from both ~~XOR~~ XNOR, 1 from XNOR only when same input

Example : 6.5 :

a) 10 and 10



b) 11 & 10

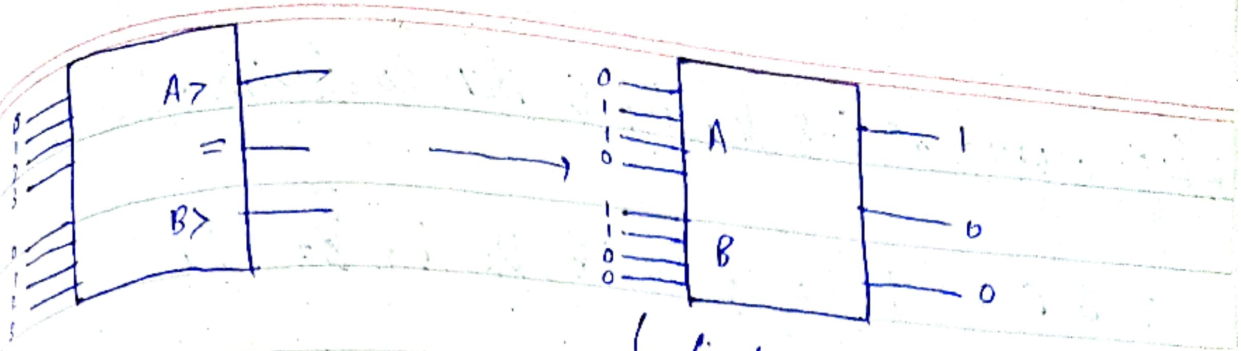


→ Inequality :

- Provide additional info $A > B$ or $A < B$
- Compare MSB A_3 and B_3

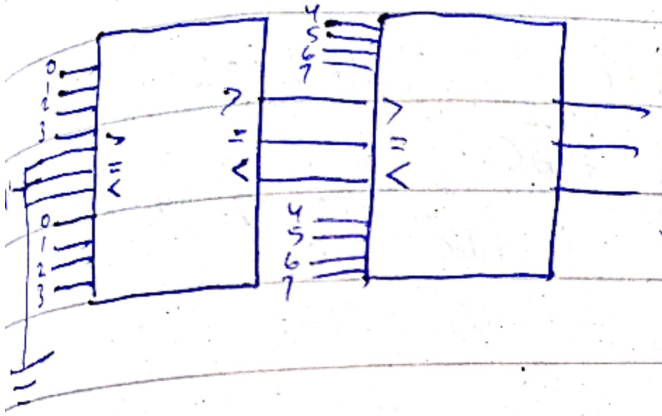
If different
$A > B$ or $A < B$

If same, check
A_2 & B_2

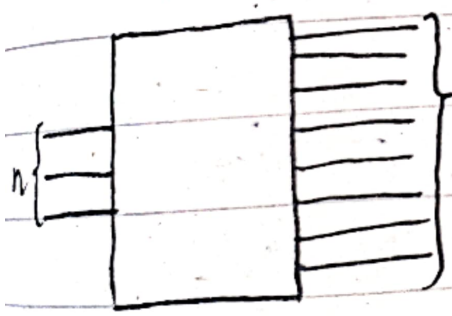


↳ first compare A_3, B_3, same
Now A_2, B_2, $A_2 > A_0$, $A > B$

Example: Cascade comparators for 8 bit



• Decoders:

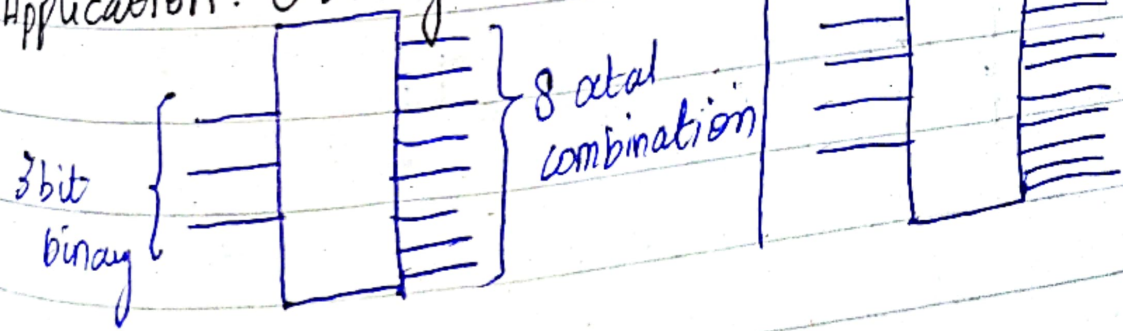


$2^3 = 8$ max can be $\leq 2^n$

• 1 – 2 decoder, 2-4 decoder,
3-8 decoder, 4-16 decoder ....

• In given conditions, only 1 output is high at a time

→ Application: ① Binary to Octal / BCD to decimal

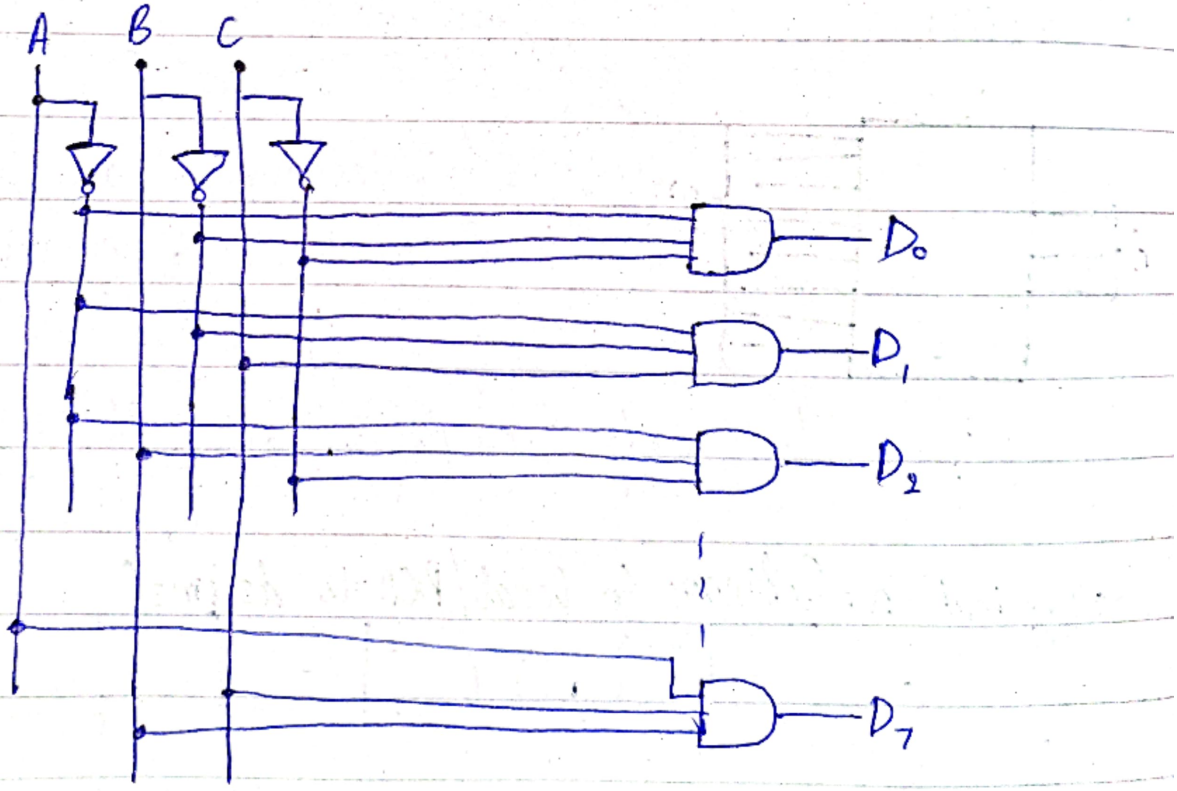


* Logic diagram of 3-8 decoder & BCD to decimal

A	B	C	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	$0 = \bar{A}\bar{B}\bar{C}$							
0	0	1		$0 = \bar{A}\bar{B}C$						
0	1	0			$0 = \bar{A}B\bar{C}$					
0	1	1				$0 = \bar{A}BC$				
1	0	0					$0 = A\bar{B}\bar{C}$			
1	0	1						$0 = A\bar{B}C$		
1	1	0							$0 = AB\bar{C}$	
1	1	1								$0 = A \cdot BC$

∴ Active low

Use high active for 1 on appropriate output



- Enable pin is also there but we assume it as high always

* BCD to decimal decoder

- Use 4-16 decoder (only 10 outputs as BCD is 0-9

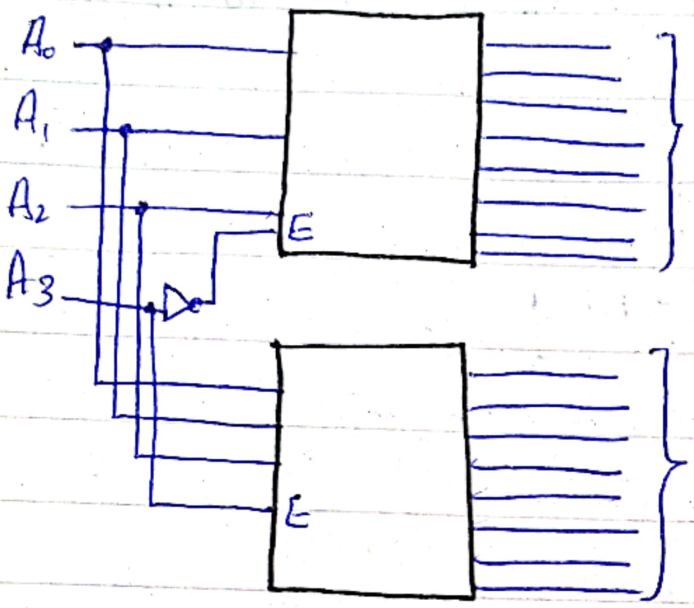
A	B	C	D	D_1 D_2 D_3 D_4 D_5 D_6 D_7 D_8 D_9 ~~D_{10} D_{11} D_{12} D_{13}~~
0	0	0	0	$D_1 = \bar{A}\bar{B}\bar{C}\bar{D}$
0	0	0	1	$1 = \bar{A}\bar{B}\bar{C}D$
0	0	1	0	$1 = \bar{A}\bar{B}C\bar{D}$
0	0	1	1	$1 = \bar{A}\bar{B}CD$
0	1	0	0	$1 = \bar{A}B\bar{C}\bar{D}$
0	1	0	1	$1 = \bar{A}B\bar{C}D$
0	1	1	0	$1 = \bar{A}BC\bar{D}$
0	1	1	1	$1 = \bar{A}BCD$
1	0	0	0	$1 = A\bar{B}\bar{C}\bar{D}$
1	0	0	1	$1 = A\bar{B}\bar{C}D$
1	0	1	0	BCD not greater that 9
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

14/5/16

- For higher order decoder we will use multiple low order decoder to make it cost effective & less complex

* 4-16 decoder using 3-8 decoder

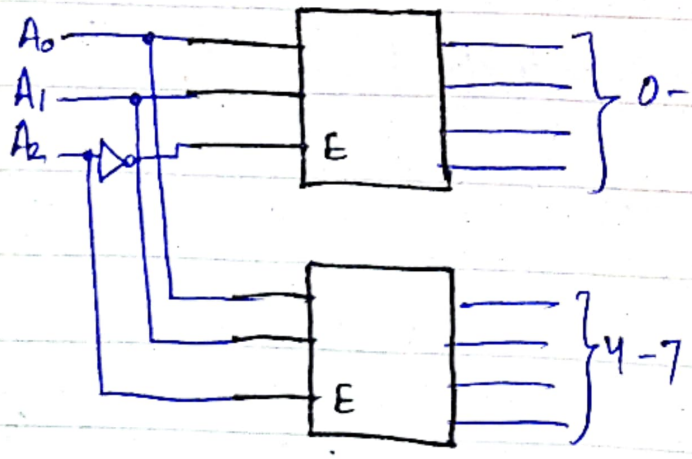
↳ 8 output, 16 needed, use 2 decoder



0-7 → 0xxx (2 disabled)
8-15 → 1xxx (1 disabled)

* 3-8 decoder using 2-4 decoder

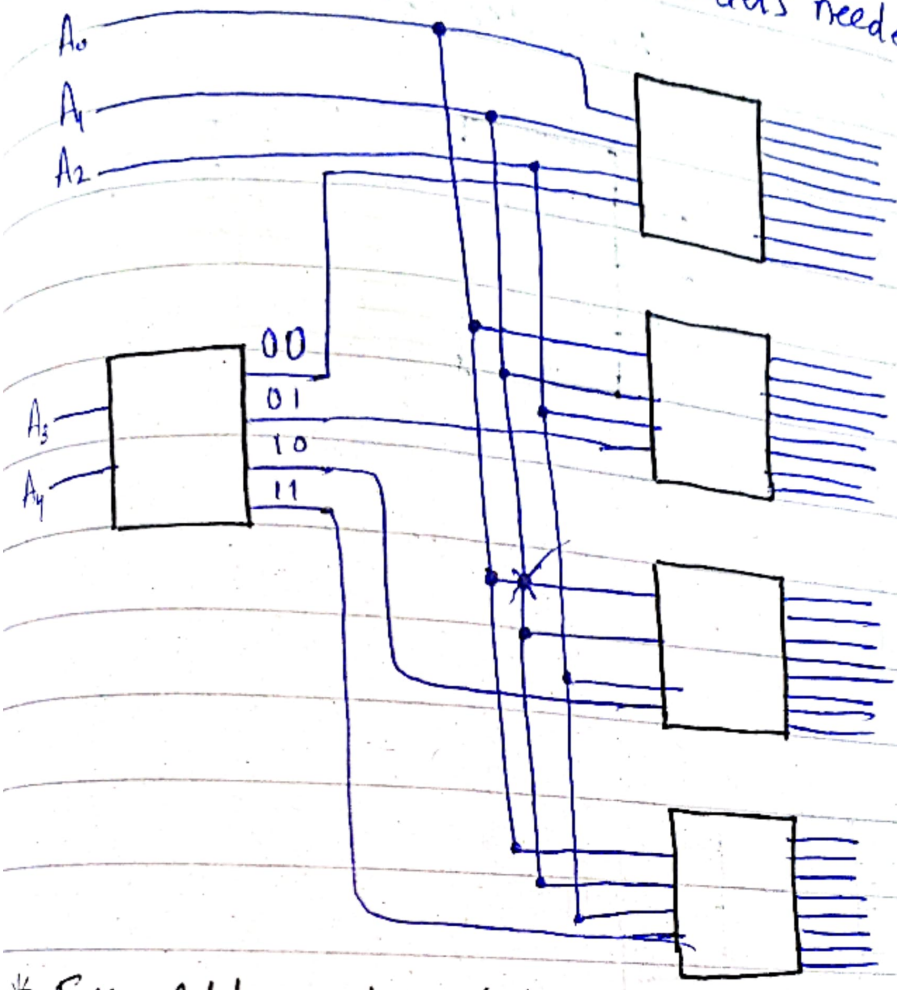
↳ 4 output, 8 needed, 2 required



0	0	0	1st activate, 2nd disable
0	0	1	
0	1	0	
0	1	1	
1	0	0	2nd activate 1 disabled
1	0	1	
1	1	0	
1	1	1	

* 5-32 using 3-8 decoder

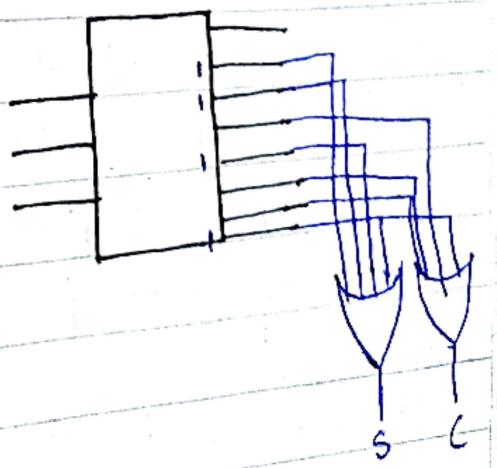
↳ 8 output, 4 decoders needed



* Full Adder using decoder

A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

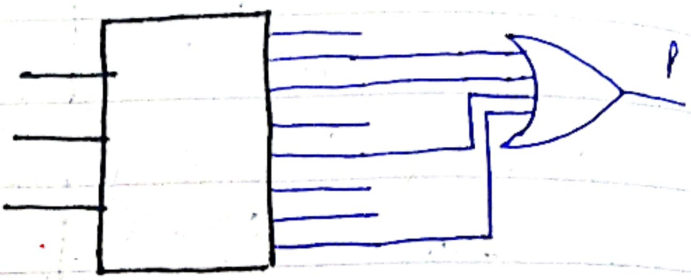
∴ Use 3-8 decoder



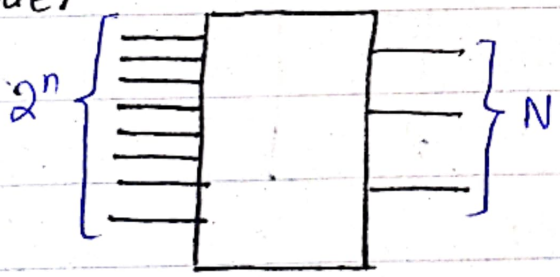
* 3 bit even Parity checker using decoder

A	B	C	P
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

∴ 3-8 decoder

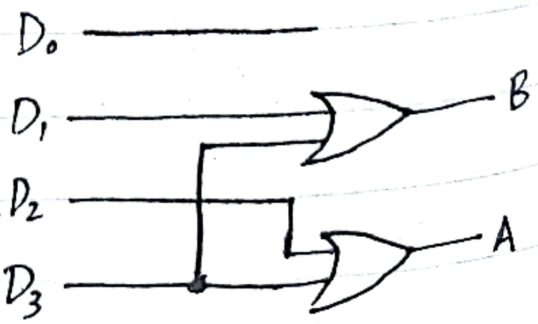


* Encoder



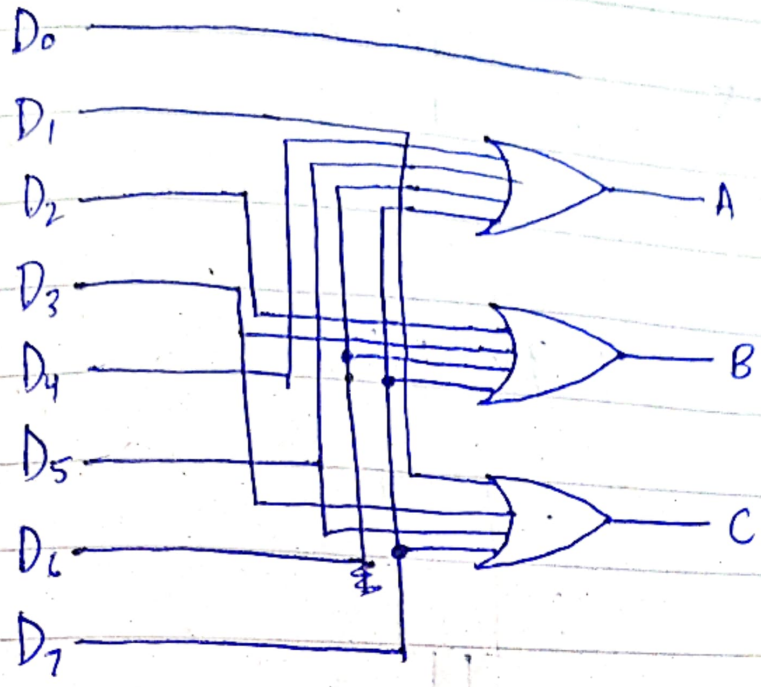
D_0	D_1	D_2	D_3	A	B
1				0	0
	1			0	1
		1		1	0
			1	1	1

∴ 4-2 encoder



- A=1, when $D_2, D_3 = 1$
- B=1, when $D_1, D_3 = 1$

-8-3 line encoder



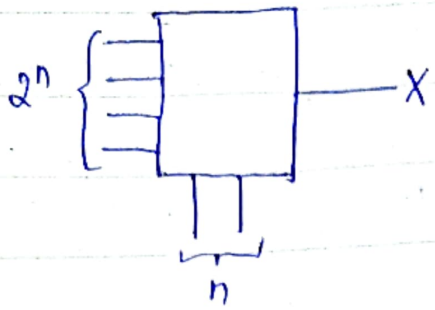
- limitation is D_0 is not connected to any and if 2D are high simultaniously it will make incorrect output for example "011" when D_1 & D_2 high simultaniously making D_3 as high, which is wrong.

- Now we use priority encoder, so input with higher subscript have higher priority
- In this case when D_1 & D_2 high simultaniously. D_2 will get go priority making 010 (correct)

→ ~~Example 6.11~~ Check 6.6

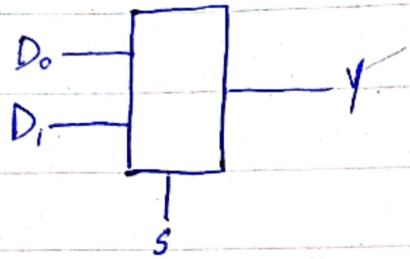
a) 2 = 0010, 9 = 1001 ⇒ 1011 (Not a valid BCD)

b) 1 = 0001, 5 = 0101 ⇒



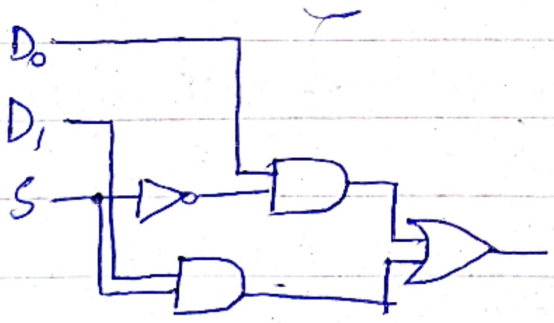
0	0
0	1
1	0
1	1

* 2x1 MUX

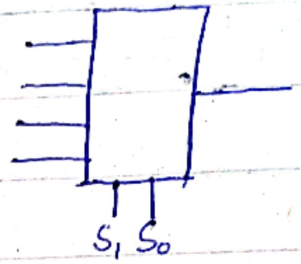


S	Y
0	D_0
1	D_1

$Y = \bar{S} D_0 + S D_1$



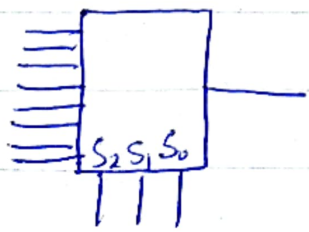
* 4x1 MUX



S_{e1}	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

$\therefore Y = \bar{S}_1 \bar{S}_0 D_0 + \bar{S}_1 S_0 D_1 + S_1 \bar{S}_0 D_2 + S_1 S_0 D_3$

* 8x1 MUX



S_2	S_1	S_0	Y
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7

$Y = \bar{S}_2 \bar{S}_1 \bar{S}_0 D_0 + \bar{S}_2 \bar{S}_1 S_0 D_1 + \bar{S}_2 S_1 \bar{S}_0 D_2 \ldots\ldots$

* Making 8x1 Mux using 4x1 MUX