Morris mano.
Anand Kumar

Digital Logic Design

* Number System

Normally we use decimal system :-

$N = d_n d_{n-1} d_{n-1} \cdots d_0 d_{-1} d_{-2})_b$

d ⇒ digit
b ⇒ base / radix.
d_n ⇒ Most significant digit
d_{n-2} ⇒ Least significant digit

In decimal System
b ⇒ 10.

$(325)_{10} \Rightarrow 5 \times 10^0 + 2 \times 10 + 3 \times 10^2$

$(1010)_2 \Rightarrow 0 \times 2^0 + 1 \times 2^1 + 0 \times 2^2 + 1 \times 2^3$
$= 0 + 2 + 0 + 8$
$= 10.$

$(127)_8 = 7 \times 8^0 + 2 \times 8^1 + 1 \times 8^2$
$= 87.$

Hexadecimal → 16 digits
0 1 2 3 4 5 6 7 8 9
normal till 9.

10	A
11	B
12	C
13	D
14	E
15	F

$(AF)_{16} = 15 \times 16^0 + 10 \times 16^1$
$= 175$

Q. $(12.23)_8 \longrightarrow (\quad)_{10}$
(+ve . −ve)

$\Rightarrow 2 \times 8^0 + 1 \times 8^1 + 2 \times 8^{-1} + 3 \times 8^{-2}$

* BINARY
We use binary in digital as it comprises of 0 & 1 i.e. it follows on-off logic.

Decimal to Binary

$(17)_{10} \longrightarrow (\quad)_2$

Method 1: Identify highest power of 2 $\leq N$.

$2^4 = 16 \leq 17$
$4 \rightarrow 5^{th}$ bit.

1 _ _ _ _

Do $17 - 2^4 = 1$
& repeat for 1
$2^0 = 1 \leq 1$

$\Rightarrow$ 1 0 0 0 1

Q. $(127)_{10} \longrightarrow (\quad)_2$

16 2^4
64 2^6

127.
$2^6 \Rightarrow 64$. 7th.
63.
$32 \Rightarrow 2^5$ 6th.
31
$2^4 \Rightarrow 16$ 5th.
15.
$8 \Rightarrow 2^3$ 4th.
7
$2^2 \Rightarrow 4$ 3rd.
3
$2 \Rightarrow$ 2nd
1 1st.

$\Rightarrow 1\ 1\ 1\ 1\ 1\ 1\ 1$

Method 2: dividend remainder

2 | 17		
2 | 8	1	
2 | 4	0	10001 $_2$
2 | 2	0	
0 | 1	0	
	1	

MORRIS MANO
ANAND KUMAR

Q.) $(49)_{10} \longrightarrow (\quad)_2$

2 | 49		
2 | 24	1	
2 | 12	0	
2 | 6	0	110001
2 | 3	0	
2 | 1	1	
2 | 1	1	
1		

Q.) Conversion

i) $(10.12)_{10} \longrightarrow (\quad)_2$

for 10 its clear.

for 0.12

0.12×2	$= 0.24$	0
0.24×2	$= 0.48$	0
0.48×2	$= 0.96$	0
0.96×2	$= 1.92$	1

So till it forms a whole number else 4-5 digits.

$\Rightarrow (1010.0001)_2$

Q. $(32.1)_{10} \longrightarrow (\quad)_8$

8 | 32	0	
8 | 4	4	
0		

$\Rightarrow 40$

0.1×8	$= 0.8$	$\longrightarrow 0$
0.8×8	$= 6.4$	$\longrightarrow 6$
0.4×8	$= 3.2$	$\longrightarrow 3$
0.2×8	$= 1.6$	$\longrightarrow 1$
0.6×8	$= 4.8$	$\longrightarrow 4$

$\Rightarrow (40.06314)_8$

Q. ~~BIN~~

* BINARY to OCTAL/HEXA.

Q.) $(010\ 110\ 101 \cdot 001\ 10)_2 \longrightarrow (\quad)_8$

$2^3 \Rightarrow 8$.

so take 3bits togethers

$(010\ 110\ 101 \cdot 001\ 100)_2$

$\longrightarrow (2\ 6\ 5 \cdot 1\ 4)_8$

$2^4 = 16$.

take 4 bits together.

$(0000\ 1011\ 0101 \cdot 0011\ 0000)_2$

$0\ B.\ 5 \cdot 3\ 0.$

$\Rightarrow (B5 \cdot 30)_{16}$.

Q). $(611\ 001 \cdot 001\ 101)_2$

$(3\ 1 \cdot 1\ 5)_8$.

$0001\ 1001 \cdot 0011\ 0100$

$(1\ 9 \cdot 3\ 4)_{16}$

Octal to Hexa.

Q). $(321.23)_8 \rightarrow (\quad)_{16}$

first convert to binary.

$3\ 2\ 1 \cdot 2\ 3$

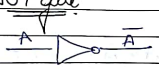
$(011\ 010\ 001 \cdot 010\ 011)_2$

$0\ D\ 1 \cdot 4C$

$(D1 \cdot 4C)_{16}$.

* Logic Gates

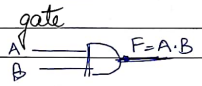
1) NOT gate



$F = \bar{A}$

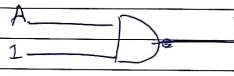
A	F
0	1
1	0

2) AND gate



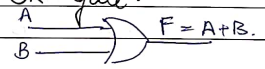
$F = A \cdot B$.

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

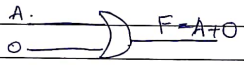


buffer.

3). OR gate.

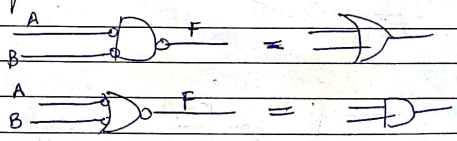


A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

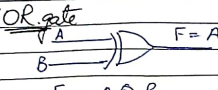


buffer.

4) NOR gate.



4) XOR gate



$F = A \oplus B$

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

also called inequality detector.

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

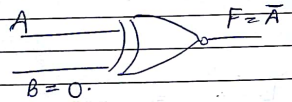
5) XNOR gate.

$F = A \odot B$

$= \overline{A \oplus B}$

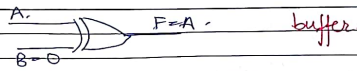
$= \bar{A}\bar{B} + A \cdot B$

A	B	F
0	0	1
0	1	0
1	0	0
1	1	1

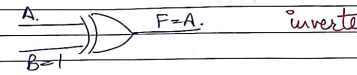


inverter

buffer

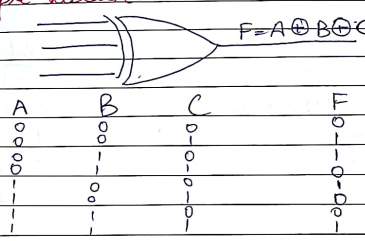


buffer.



inverter.

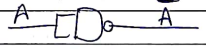
⇒ Triple Variable.



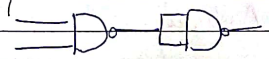
⇒ for XOR → we need odd number of '1's.

Q.) Make the following from following.

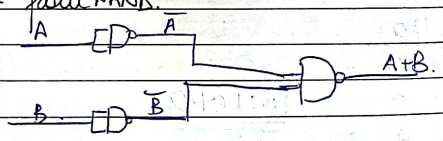
i) NOT from NAND.



ii) AND from NAND.



iii) OR from NAND.



* Binary Algebra

Addition / subtraction.

Q.
$$\begin{array}{r} 101.11 \\ +\ 101.01 \\ \hline 1011.00 \end{array}$$

Add

0+0=0.
1+1=10 → 0 carry 1
0+1=1
1+0=1

Sub

0−0=0.
1−0=1
1−1=0.
0−1=1
1 ⇒ borrow

Q.
$$\begin{array}{r} +01.+1 \\ -\ +01.-01 \\ \hline \end{array} \qquad \begin{array}{r} 10001 \\ -\ 01001 \\ \hline 00110 \end{array}$$

Q.
$$\begin{array}{r} 1 \rightarrow 00101 \\ -\ 01010 \\ \hline (-11011). \end{array} \qquad \begin{array}{r} 11.01 \\ -\ 0.10 \\ \hline 10.11 \end{array}$$

Multiplication & Division

Q.
$$\begin{array}{r} 111.1 \\ \times\ 110 \\ \hline 1000.0 \\ 1111\ 0 \\ \hline 101101.0 \\ 111100 \end{array}$$

Q.
$$\begin{array}{r} 110\ \overline{)\ 101101}\ (111.1 \\ -\ 110 \\ \hline 1010 \\ -\ 110 \\ \hline 1001 \\ -\ 110 \\ \hline 0110. \\ -\ 110 \\ \hline \times \end{array}$$

see first 3 digits. If it is smaller take 4.

* r's complement and (r−1)'s complement.

Write the magnitude of the number but the first bit is reserved for sign.

1's complement.

$$\begin{array}{r} 11111 \\ -\ 10101 \\ \hline 01010 \end{array}$$

Either subtract from 1 or
Do 0 → 1, 1 → 0 those changes.

Q) $(746)_{10}$ → 9's complement.

$$\begin{array}{r} 999 \\ -\ 746 \\ \hline 253 \end{array}$$

10's complement → ?
9's complement + 1

(746) → 253 in 9's complement
+1
254.

Q. $(10101\,11)_2$ → 1's & 2's complement

$(010100)_2$ → 1's comple.
+1
010101 → 2's complement.

	Signed	1's complement	2's comp.
+5	0101	0101	0101
-5	1101	1010	1011

No change in 1's comp & 2's compl. for +ve number.

-9	11001	10110	10111
-16	110000	101111	110000

For 2^n, we can drop sign bit.
i.e. 10000 is -16 in 2's compliment.

⇒ We can write copy sign bit to LHS for any number.

eg 11111001 ✓

So how to recognise?

11111111 → -ve & 2's complement
↓ find 2's compliment.
00010000
+1
00010001 → 9.

Ans = -9.

Q. 11000111 → ? 2's complement.

1 10001101 → -ve.
↳ 01110010
+1
01110011 = 1+2+16+32+64.
= -115

* Overflow
When the output cannot be represented in our bit memory, it is called overflow.

2's compliment arithmetics

1) 13 - 12

13 - 12 ⇒ 13 + (-12)

13 = 01101

(2's comp) -12 ⇒ 11100 → 00011
+1
10100.

0 1101
1 0100
(1) 0 0001
carry

00001

Rule 1: Discard the carry

Q2. -13 + 12

(-13) + 12.

-13 1 1101

0010

1 1 0011

12 → 110

1 0011
+ 0 0 1 0 0
1 1 1 1 1 1.

Ans = 1 1 1 1

Q3. -3 - 4

-3 + (-4).

(-3) → 1 1 0 1

-4 100
011
1 1 100

1 1 0 1
1 1 0 0
1 0 0 1
carry

Ans = 1001

But 1 0001 ~~-1~~.

alternate

in 2's compliment, sign bit can be copied.

so, -3 = 101
-4 = 100.
(1) 001

001 = +1 which is wrong.

⇒ Take atleast 1 extra bit.

Q. 1) 127 - 54
2) 17 - 32
3) -16 - 128.

* 1's compliment Arithmetics

1) 13 - 12

13 + (-12)

13 = 1101
-12 ⇒ 1100
1 0011

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

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

end around.
i.e. add this carry.

000000
+1
000001 ⇒ +1

Q. 12 − 13 (8 bit)

12 + (−13).

13 → 0000 0000 1100

−12 → 1111 1111 0010

1111 1110

1 0000 0001

Q −3 − 4 (8 bit)

−3 ⇒ 1111 1100.

−4 ⇒ 1111 1011

0000 0111

+1

1111 1000

1111 1000 → −7.

1000 0111

* BCD Codes. (Binary coded decimals)

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

But for

10 → 1 0 → 0001 0000

11 → 1 1 → 0001 0001

12 → 1 2 → 0001 0010.

* Excess 3 Code.

		Add 3
0	0000	0010
1	0001	0100
2	0010	0101
3	0011	↓
4	0100	↓
5	0101	↓
6	0110	↓
7	0111	↓
8	1000	→ 1011
9	1001	→ 1100

eg 238.

→ 2 3 8

⇒ 0101 0110 1011

9's complement of 238 → 7 6 1

1010 1001 0100

↓ ↓ ↓ 1's comp

0101 0110 1011

i.e. 238.

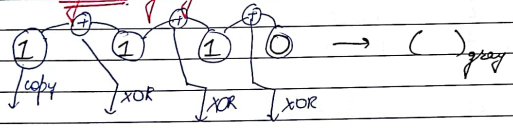
* Gray Code

→ Used in communication and error detection.

→ 2 consecutive numbers differ at exactly 1 bit.
i.e. we can change only 1 bit.

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

Binary to Gray

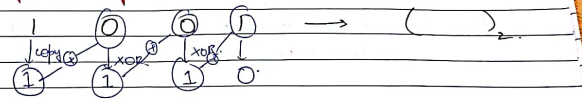
Q. 

1 0 0 0

Q. 1 0 1 1 0 0 1 0

⇒ 1 1 1 0 1 0 1 1.

Gray to binary

Q. 

⇒ 1 1 1 0.

* Sum of Product and Product of sum. and Truth Table.

Q. Write a Truth table for all numbers 0 to 7 divisible by 3.

0 →	A	B	C	⇒ atleast 3 bit -
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	0

Now we can express F in 2 ways:-

Minterms / SOP (output = 1)

Maxterms / POS. (output = 0)

$\Sigma(m_0, m_3, m_6)$

$\Pi(M_1, M_2, M_4, M_5, M_7)$

$F = \Sigma(0,3,6)$

$F = m_0 + m_3 + m_6$

$F = \bar{A}\cdot\bar{B}\cdot\bar{C} + \bar{A}\cdot B C + A\cdot B\cdot\bar{C}$

(complement for '0')

$F = \Pi(M_1, M_2, M_4, M_5, M_7)$

$F = \Pi(1,2,4,5,7)$

$F = M_1 \cdot M_2 \cdot M_4 \cdot M_5 \cdot M_7$

$F = (A+B+\bar{C})\cdot(A+\bar{B}+C)\cdot(\bar{A}+B+C)\cdot(\bar{A}+B+\bar{C})\cdot(\bar{A}+\bar{B}+\bar{C})$

(complement for '1')

* K-MAP

* 3 variable

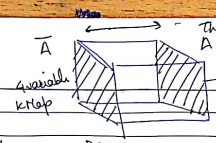
A	B	C
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

A \ BC	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	000 (0)	001 (1)	011 (3)	010 (2)
A	100 (4)	101 (5)	111 (7)	110 (6)

* 4 variable

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

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0 (0000)	1 (0001)	3 (0011)	2 (0010)
$\bar{A}B$	4 (0100)	5 (0101)	7 (0111)	6 (0110)
AB	12 (1100)	13 (1101)	15 (1111)	14 (1110)
$A\bar{B}$	8 (1000)	9 (1001)	11 (1011)	10 (1010)



* 5 variable

Draw 2 kMap, one with $X = 0$ and 1 with $X = 1$.

$X = 0$:

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	1	3	2
$\bar{A}B$	4	5	7	6
AB	12	13	15	14
$A\bar{B}$	8	9	11	10

$X = 1$:

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	16	17	19	18
$\bar{A}B$	20	21	23	22
AB	28	29	31	30
$A\bar{B}$	24	25	27	26

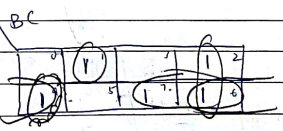
NOTE.

for connection b/w 2 tables if there is just a difference of 1 bit, they can be clubbed.

$0 \rightarrow 16$, $8 \rightarrow 24$.

Q Draw a kMap for 3 bit ~~number~~ truth table T.T.

A	B	C	F
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	1
1	1	1	1



$= \bar{A}\bar{B}C + B\bar{C} + AB + A\bar{C}$

Q. $\Sigma = (0,1,2,3,4,6,7) = ?$

Full Subtractor

A	B	B_{in}	Diff	B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

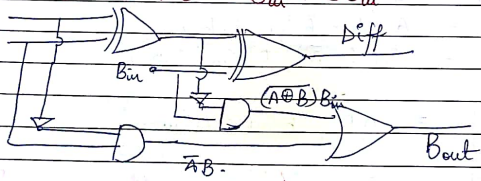
for Diff

$\Rightarrow \bar{A}\bar{B}B_{in} + \bar{A}B\bar{B}_{in} + A\bar{B}\bar{B}_{in} + ABB_{in}$

$Diff = A \oplus B \oplus B_{in}$

for Borrow

$Borrow = \bar{A}B + \bar{A}B_{in} + BB_{in}$



$B_{out} = (A \oplus B)B_{in} + \bar{A}B.$

$= \bar{A}BB_{in} + \bar{A}\bar{B}B_{in} + \bar{A}B.$

$= ABB_{in} + \bar{A}(B + \bar{B}B_{in})$

$= ABB_{in} + \bar{A}(B+\bar{B})(B + B_{in})$ ($B+\bar{B} = 1$)

$= ABB_{in} + \bar{A}B + \bar{A}B_{in}$

$= B(\bar{A}+A)(\bar{A}+B_{in}) + \bar{A}B_{in}.$

$= \bar{A}B + B_{in}B + \bar{A}B_{in}.$

$B_{out} = \bar{A}B + B_{in}B + \bar{A}B_{in}.$

Comparators

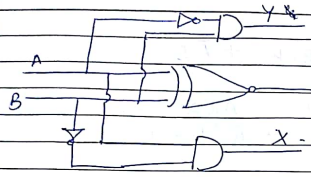
1 bit comp

A	B	x: A>B	y: A<B	z: A=B
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1

$x = m_2 = A\bar{B}$

$y = m_1 = \bar{A}B$

$z = m_0 + m_3 = \bar{A}\cdot\bar{B} + AB = A \odot B.$



4 bit comparator.

$A \Rightarrow a_3\ a_2\ a_1\ a_0$

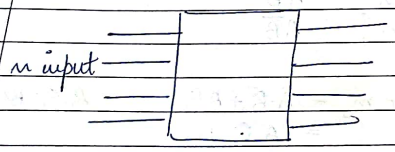
$B \Rightarrow b_3\ b_2\ b_1\ b_0$

$\Rightarrow F(A=B) = (a_3 \odot b_3)\cdot(a_2 \odot b_2)\cdot(a_1 \odot b_1)(a_0 \odot b_0)$

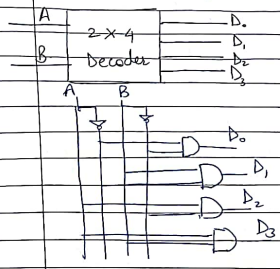
$\Rightarrow F(A>B) = a_3\cdot\overline{b}_3 + (a_3 \odot b_3)\cdot a_2\overline{b}_2 + (a_3 \odot b_3)\cdot(a_2 \odot b_2)\cdot a_1\overline{b}_1 + (a_3 \odot b_3)(a_2 \odot b_2)(a_1 \odot b_1)\,a_0\cdot\overline{b}_0$

$F(A<B) = \overline{a}_3\cdot b_3 + (a_3 \odot b_3)\cdot\overline{a}_2 b_2 + (a_3 \odot b_3)(a_2 \odot b_2)\cdot\overline{a}_1 b_1 + (a_3 \odot b_3)(a_2 \odot b_2)(a_1 \odot b_1)\cdot\overline{a}_0 b_0$

Decoders.

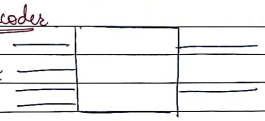


→ Used to detect input combination.



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

Encoder



	E_3	E_2	E_1	E_0	D_1	D_0	
m_8	1	0	0	0	1	1	$D_1 = E_3\overline{E}_2\overline{E}_1\overline{E}_0 + \overline{E}_3E_2\overline{E}_1\overline{E}_0$
m_4	0	1	0	0	1	0	
m_2	0	0	1	0	0	1	$D_0 = E_3\overline{E}_2\overline{E}_1\overline{E}_0 + \overline{E}_3\overline{E}_2E_1\overline{E}_0$
m_0	0	0	0	1	0	0	

eg if input sequence is $E_3E_2E_1E_0$ 1101 not allowed.

Priority Encoder

Works on priority.

Priority is directly related to significance of bits.

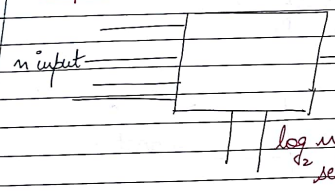
E_3	E_2	E_1	E_0	D_1	D_0
1	X	X	X	1	1
0	1	X	X	1	0
0	0	1	X	0	1
0	0	0	1	0	0

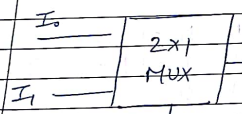
x ⇒ doesn't matter

$D_1 = E_3 + \overline{E_3} E_2$ $\quad D_0 = E_3 + \overline{E_3}\cdot\overline{E_2}\cdot E_1$

in truth table consider all possible case of x.

Multiplexers (Data Selectors)



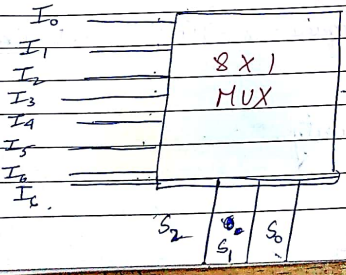


$F = \overline{S} I_0$ or $F_1 = S I_1$

$F = \overline{S} I_0 + S I_1$

S=0.

S=1



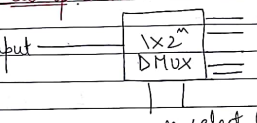
$S_2 S_1 S_0$ select input.

$F = I_0 \cdot \overline{S_2}\,\overline{S_1}\,\overline{S_0} + I_1 \cdot \overline{S_2}\,\overline{S_1} \cdot S_0 + I_2 \cdot \overline{S_2} S_1 \cdot \overline{S_0}$

$+ I_3 \cdot \overline{S_2} S_1 S_0 + I_4 S_2 \cdot \overline{S_1}\,\overline{S_0} + I_5 \cdot S_2 \cdot \overline{S_1} \cdot S_0$

$+ I_6 \cdot S_2 \cdot S_1 \cdot \overline{S_0} + I_7 \cdot S_2 S_1 S_0$.

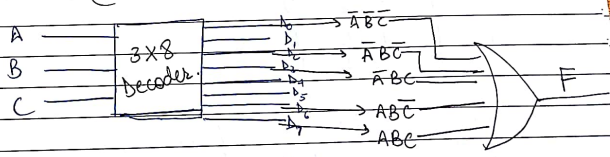
* De Multiplexer



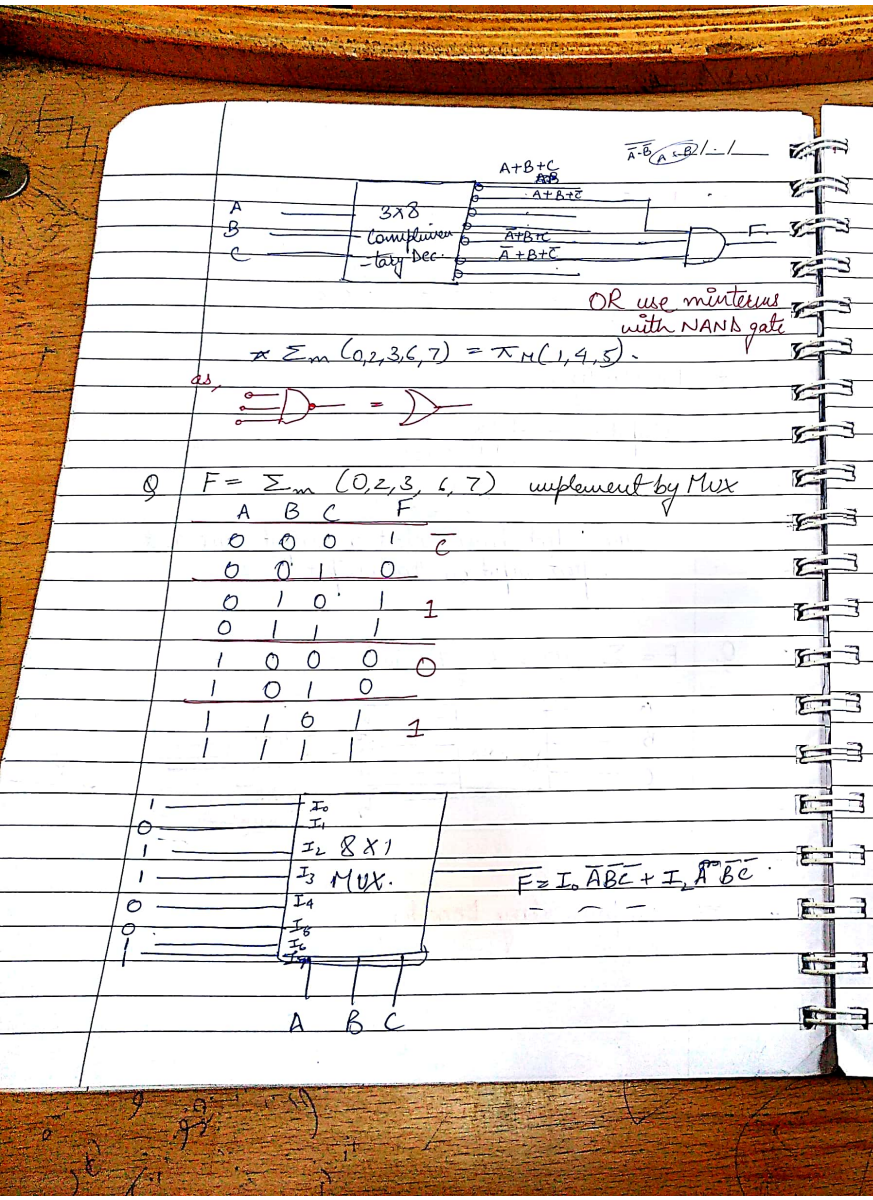
m select lines

m select lines select a output line and reflect input on that output line.

Q. $F = \Sigma_m (0,2,3,6,7)$ in Decoder.



⇒ Complementary Decoder.



OR use minterms with NAND gate

$\pi \Sigma_m(0,2,3,6,7) = \pi_M(1,4,5)$.

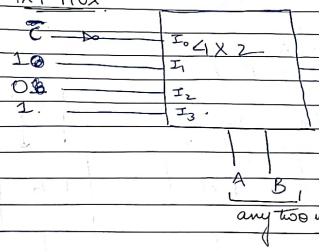
as,

Q $F = \Sigma_m(0,2,3,4,7)$ implement by MUX

A	B	C	F	
0	0	0	1	$\bar{C}$
0	0	1	0	
0	1	0	1	1
0	1	1	1	
1	0	0	0	0
1	0	1	0	
1	1	0	1	1
1	1	1	1	

$F = I_0\bar{A}\bar{B}\bar{C} + I_2\bar{A}\bar{B}C$.

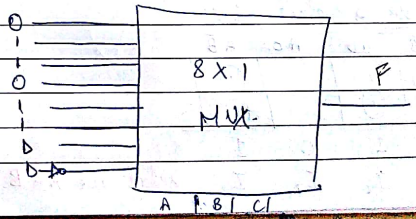
4X1 MUX



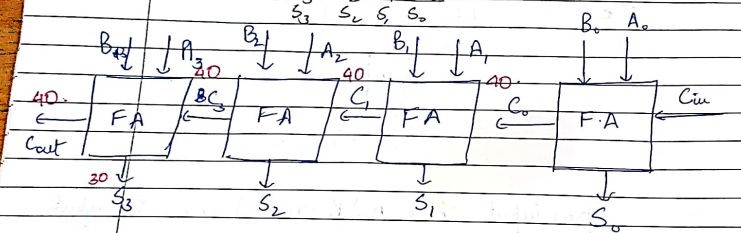
pair input where A & B same & represent output in terms of 0, 1, C

Q. $F = \pi_M(0,1,6,7,12,15)$. 8X1 MUX

A	B	C	D	F	
0	0	0	0	0	0
0	0	0	1	1	1
0	0	1	0	1	1
0	0	1	1	0	0
0	1	0	0	1	1
0	1	0	1	1	1
0	1	1	0	0	$\bar{D}$
0	1	1	1	0	$\bar{D}$



* 4 bit parallel adder.



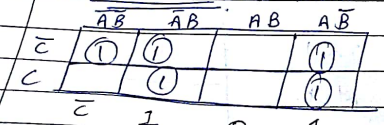
If sum takes 30 ns
carry takes 40 ns
Total time = ?

⇒ sum = 40 + 40 + 40 + 30
= 150 ns

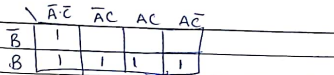
carry = 40 + 40 + 40 + 40
= 160 ns

NOTE

for a 4×1 MUX in Question
$\Sigma_m (0, 2, 3, 6, 7)$
To chose a select lines



∴ selct line A & B.



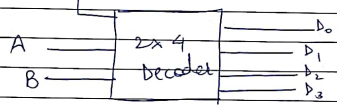
select line = AC

* Decoder using Enables

2 kinds of enabler — high (1), low (0).

decoder works — output always

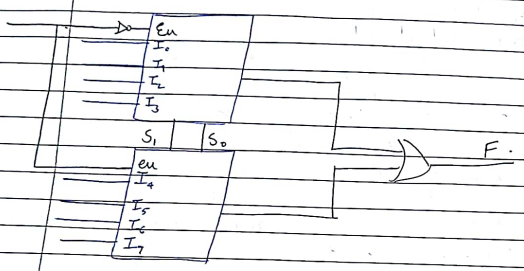
eg 2×4



$\bar{E}_m$	A	B	D_0	D_1	D_2	D_3
1	X	X	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	1	0	0
0	1	0	0	0	1	0
0	1	1	0	0	0	1

8×1 MUX by 4×1 MUX.

Now, in 8×1 MUX we need 3 select lines.
$S_2 S_1 S_0$
selecting $S_1 S_0$ for 4×1



but we need to select 1 number so we need 3 bit.

(for 0 & 4
0 00 1 00

8x1 using 2 4x1 MUX.

making n×1 MUX from m×1 MUX.

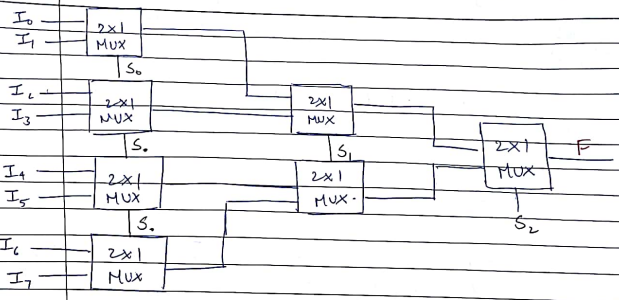
$= \left(\frac{n\times 1}{m\times 1}\right) \times \frac{1}{m} + \left(\frac{n\times 1}{m\times 1}\right)$ (m×1) MUX needed.

± 1 OR gate.

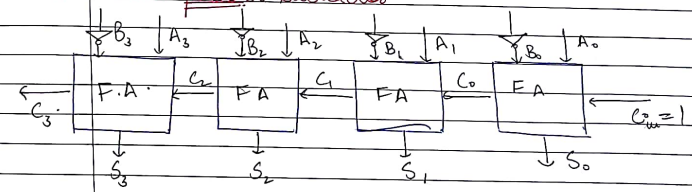
eg $\left(\frac{8\times 1}{2\times 1}\right)\times\frac{1}{2} + \left(\frac{8\times 1}{2\times 1}\right)$

$= 2 + 4$

$= 6$ 2×1 MUX.



4 bit parallel subtractor



$A - B \Rightarrow A + (-B)$.

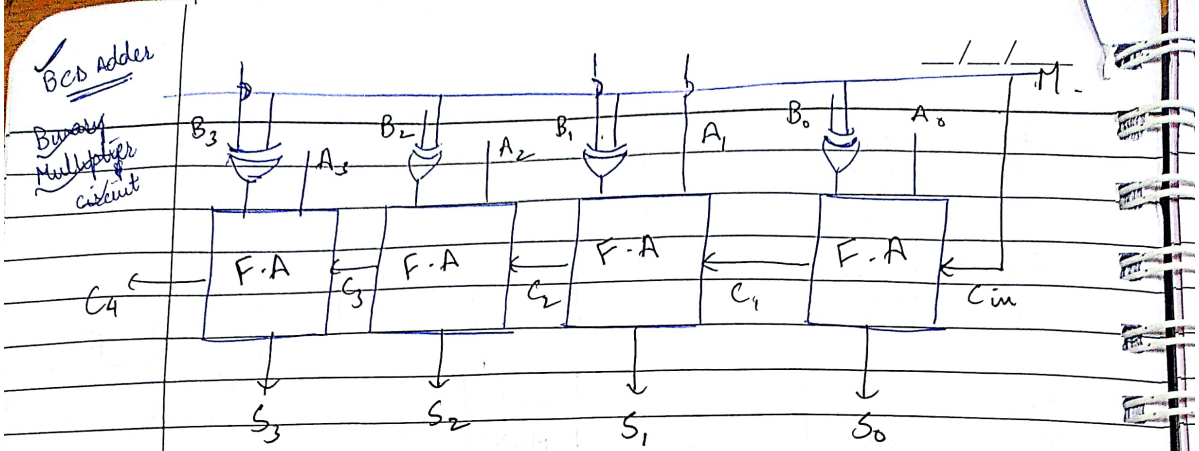
$\Rightarrow$ A + 2's complement of B.

$\bar{B} + 1$ (not gate) $\rightarrow C_{in} = 1$.

Adder subtractor combined

BCD adder

Binary Multiplier circuit



if $M = 1 \Rightarrow 1 \oplus B = \overline{B}$ i.e. subtractor

$M = 0 \Rightarrow 0 \oplus B = B$ i.e. Adder.